Черкаський національний університет імені Богдана Хмельницького

Кафедра програмного забезпечення автоматизованих систем

Ілля Порубльов

Елементи теорії ігор (частина 1)

Навчальний посібник

Черкаси, 2025

УДК 519.1/519.83:004.42(075.8)
П60

Порубльов, І. Елементи теорії ігор (част. 1) [Електронне видання] : навч. посіб. / І. Порубльов; Черкаський нац. ун-т ім. Б. Хмельницького. — Черкаси : ЧНУ ім. Б. Хмельницького, 2025. — 117 с.

Рекомендовано до видання Вченою радою Черкаського національного університету імені Богдана Хмельницького (протокол № *4* від "*25*" червня 2025 року).

Рецензенти:
Міца О. В., доктор технічних наук, професор, завідувач кафедри інформаційних управляючих систем і технологій ДВНЗ "Ужгородський національний університет".
Сидоров М. В., доктор фізико-математичних наук, професор, завідувач кафедри прикладної математики Харківського національного університету радіоелектроніки.
Онищенко Б. О., кандидат фізико-математичних наук, доцент, декан факультету обчислювальної техніки, інтелектуальних та управляючих систем Черкаського національного університету імені Богдана Хмельницького.

Поточну частину 1 цього навчального посібника зосереджено головним чином на комбінаторних іграх (виграшні та програшні позиції, функція та теорема Шпрага-Ґранді). Крім того, оглянуто загальну класифікацію ігор у математичній теорії ігор, застосування динамічного програмування та теорії ймовірності до ігор з числовими виграшами.

Особливістю посібника є те, що для більшості згаданих тем наведено, крім теорії, задачі, розв'язками яких є комп'ютерні програми, а перевірку організовано на сайті автоматичної перевірки https://dmoj.cdu.edu.ua. Це робить посібник орієнтованим переважно на студентів галузі знань F (12) «Інформаційні технології».

Зміст

Вступ

Це видання є 1-ю частиною посібника з дисципліни «Елементи теорії ігор»; 2-а частина ще перебуває у процесі розробки.

Деякі технічні особливості цього посібника. Наведені тут задачі 1.2:A, 1.2:B, 1.2:C і ще з пів сотні інших є задачами з програмування, призначеними й підготовленими для автоматичної перевірки. За задумом автора, задачі та можливість їх автоматичної перевірки є по суті складовою посібника. Наразі планується, що вони будуть доступними за посиланням `https://dmoj.cdu.edu.ua` [1], але, чесно кажучи, наразі ще не відомо, чи точно ця система автоматичної перевірки працюватиме досить довго й досить стабільно; чи буде доступною публічно, чи за запрошенням; чи довго адреса лишатиметься незмінною, тощо.

В цьому посібнику, що поширюється як `pdf`-файл, є чимало гіперпосилань у Інтернет, виділених синім кольором, а також посилань на інші місця цього ж `pdf`-файла, виділених червоним (наприклад: тут вводиться поняття «граф гри»). Якщо Ви не бачите в попередньому реченні виділень синім та/або червоним кольором (не будучи дальтоніком і переглядаючи `pdf`-файл на кольоровому екрані) або хоча б якесь із них не працює як відповідного типу посилання — спробуйте інший `pdf`-переглядач; зокрема, але не тільки, підходять Acrobat (включно з безкоштовною версією Acrobat Reader), SumatraPDF.

Які ігри розглядаються в цій навчальній дисципліні? Всі ігри, що розглядаються в цій дисципліні, належать до *математичної теорії ігор*, тобто, згідно з [2], до «теорії математичних моделей прийняття оптимальних рішень в умовах конфлікту». Поняття «конфлікт» тут зазвичай розглядають у «відносно лагідному» його вигляді, як взаємодію сторін, що мають різні інтереси; однак, «лагідність» тут не обов’язкова. Теорію ігор можуть застосовувати і як складову аналізу економічної взаємодії (де йдеться про реальні прибутки/збитки), і навіть як складову аналізу воєнних дій.

Взагалі, теорія ігор зазвичай спирається на приблизно такі аксіоми.

(Цей перелік *не* є цілком стандартним. Нормально навіть, коли в різних темах («підтеоріях» теорії ігор) розглядаються трохи різні списки аксіоми; втім, тоді це варто чітко окреслювати у відповідних темах.)

- ***Є ситуація, на яку можуть впливати кілька сторін. Цю ситуацію й називають «грою», а задіяні зацікавлені сторони «гравцями».***

  (Таким чином, слово «гра» тут про взаємодію сторін, і *не* має *зовсім нічого спільного* зі словом «гратися» у смислі «знову граєшся, коли нарешті займешся чимось корисним?».

  Теорія ігор переплетена з прийняттям рішень в умовах невизначеності: і там, і там розглядають питання «як діяти, коли потрібно щось вирішити, а вплив на результат мають не лише особа, яка приймає рішення, а й інші обставини?». Однак, є й істотна відмінність. Теорія прийняття рішень в умовах невизначеності зосереджується переважно на ситуаціях, коли хоч і є непідвдадні обставини, але вони не мають свободи волі, зазвичай можуть бути описані засобами теорії ймовірності, й зазвичай не знають, що вони впливають на са́ме цей результат (наприклад: природний дощ може зіпсувати деякі види господарчої діяльності, але він не йде спеціально для того, щоб зіпсувати

са́ме цю діяльність). А ***теорія ігор концентрується на тих взаємодіях, де всі гравці про цю взаємодію знають і діють свідомо.***

У більшості тем цієї дисципліни гравців рівно двоє; у деяких окремах темах допускається участь більшої кількості гравців (3-х, 4-х, …), але ситуації, коли гравців менше 2-х, теорія ігор зазвичай не розглядає: яка тоді взаємодія? де конфлікт інтересів?

Ще з іншого боку, «зазвичай не розглядає» означає, що іноді все ж розглядає. Так буває: інструменти іноді, як виключення, використовують не за прямим призначенням. В цьому посібнику є рівно одне таке виключення — задача 1.9:A, де «один свідомий гравець “грає проти random-у”».)

- ***У кожної гри є правила (у різних ігор різні). Гравці їх знають і дотримуються.***
  (На це можна заявити «Так це для слабаків; реального успіху досягають ті, хто знаходить нестандартні ходи, виходить за рамки чужих правил і запроваджує власні». Що ж, у цьому є значна частка правди: ця дисципліна, що лежить на стику математики й програмування, *не* навчить ні підкорювати інших, ні робити революції. Однак, деяка користь від дисципліни все ж є і для тих, хто хоче запровадити власні правила. Зокрема, вона може допомогти аналізувати питання «до чого призведуть такі-то правила взаємодії, якщо їх дотримуватися?» і відкидати на етапі продумування особливо невдалі проєкти правил, обходячись без натурних експериментів чи істотно зменшуючи обсяги їх використання.)
- ***Покращення результату деякого гравця може відбуватися за рахунок погіршення результат(у/ів) інш(ого/их) гравц(я/ів); у цьому й полягає конфлікт.***
  (Нерідко це «може» означає «неминуче буде»; але в тім-то й справа, що не завжди. *Бувають* ситуації, коли можна покращити чиїсь результати, не погіршивши результати інш(ого/их) гравц(я/ів). Звісно, так буває лише в деяких іграх, а не в усіх. Але тим важливіше вміти розрізняти одні взаємодії від інших.)
- ***Кожен гравець бажає виграти і прикладає (в межах, дозволених правилами гри) всіх зусиль для свого виграшу.***
  – ***Якщо виграш має числове вираження — кожен гравець бажає виграти якнайбільше і прикладає (в межах, дозволених правилами гри) всіх зусиль для збільшення свого виграшу.***
  (Ще раз наголошу, що метою є свій виграш (як факт чи як його збільшення); наскільки тісно це пов’язано з програшем/зменшенням виграшу суперника — залежить від гри.)
- ***Ідеальний гравець спроможний повністю розуміти й пам’ятати правила гри та всі обставини, які впливають на подальший перебіг гри, передбачати наслідки своїх дій згідно правил гри, тощо.***
  (Звісно, на практиці гравці можуть бути й неідеальними; однак, толку будувати математичну теорію оптимальної гри, якщо гравці будуть неспроможними її дотриматися?
  Як варіант, можна розглядати припущення, що ідеальним є лише один з гравців, і аналіз гри проводиться «з його боку».
  У теорії ігор взагалі (за межами цієї дисципліни), задачу «як грати, враховуючи неідеальність людської пам’яті та спостережливості?» іноді все ж розглядають. Зокрема, вигадують евристичні способи, знаючи про їх недостатню науковість.
  Ще можна спочатку знайти, як повинен грати ідеальний гравець і яких результатів може при цьому досягти, а потім дослідити щось схоже на «а тепер розглянемо такий-то спосіб грати, він значно простіший, і, за таких-то додаткових припущень, такими-то математичними перетвореннями можна показати, що він дає ймовірність/матсподівання/… виграшу лише на стільки-то відсотків менше, чим складний спосіб для ідеального гравця». Як бачимо, тут для аналізу гри неідеального гравця спочатку досліджують цю ж гру для ідеального.)

Хоча більшість бакалаврських дисциплін з теорії ігор передбачає концентрацію головним чином на *матричних іграх* з [3], у цій дисципліні планується приділити їм деяку увагу, але не основну, а поточна частина 1 посібника їх не згадує взагалі, бо присвячена кільком різновидам *послідовних ігор* з [4] (зокрема, *комбінато́рним іграм* з [5]). Для комбінаторних ігор виконуються також додаткові аксіоми, які не виконуються для інших ігор.

У розд. 1.6.3 розглянуто ще одну аксіому, яку часто додають до ігор із зациклюваннями.

1 Послідовні ігри

Прикладами відомих послідовних ігор можуть бути шахи, шашки, хрестики-нулики, го, доміно, нарди, більшість ігор з гральними картами, тощо.

(Втім, так виходить, що найдетальніше в цій дисципліні будуть розглянуті не широко відомі ігри, а специфічні.)

Як бачимо хоч за списком прикладів ігор, хоч за описом з [4], послідовні ігри передбачають, що гравці впорядковані й ходять поперемінно згідно з цим порядком (у стандартному випадку, коли гравців двоє — 1-й, 2-й, 1-й, 2-й, 1-й, 2-й, …), доки гра не закінчиться. Для переважної більшості послідовних ігор типово, що протягом гри гравці роблять по багато (чи, принаймні, по кілька) ходів. Чіткої заборони на те, щоб гра закінчилася швидко (наприклад, після першого ж ходу, чи, навіть, ще не почавшись) нема, але це розглядається як нестандартні особливі випадки, а не щось типове.

Найважливішою властивістю того, що ходи робляться послідовно, є те, що *гравці, які ходять пізніше, мають інформацію про всі раніше зроблені ходи* (чи то на попередніх «раундах», чи то попередніх гравців).

(В рамках цієї дисципліни, ми нехтуємо припущенням «ой, вони-то зроблені раніше, але гравець міг забути чи не помітити», бо завжди вважаємо, що хоча б один з гравців є ідеальним.)

Багато ходів означають, що протягом цих ходів змінюватимуться *позиції* гри. На жаль, деталі поняття «позиція» можуть виходити різними для різних типів ігор, тому це поняття водночас і важливе, і *не* виходить сформулювати його централізовано, чітко й єдинократно (натомість, доводиться формулювати кілька його варіантів для різних типів ігор). Важлива властивість позицій, яка виконується для багатьох, але не всіх, ігор, згадана тут.

1.1 Основні властивості та способи класифікації послідовних ігор

1.1.1 Кількість гравців

Це сказано раніше, але важливо, тому повторю. Найважливішим частковим випадком є ігри, які за означенням можуть мати лише двох гравців, не більше й не менше. Деякі розділи (зокрема: 1.2, 1.3, 1.4, 1.5, 1.6.4, 1.7.1; разом вони становлять переважну більшість цієї частини 1 посібника) дуже істотно спираються на факт, що гравців рівно двоє; абсолютно неможливо «трохи модифікувати» розглянуті там алгоритми під ігри з іншою кількістю гравців, відповідна теорія від такої зміни повністю втрачає сенс. У деяких розділах теорії ігор допускається участь більшої кількості гравців (3-х, 4-х, …); ми згадаємо це в деяких темах, такого буде небагато (розд. 1.7.4, частково 1.8). В цьому посібнику є тільки одне дрібне виключення (задача 1.9:A), де «один свідомий гравець “грає проти random-у”». Скрізь, крім цього виключення, ми розглядатимемо взаємодію гравців і конфлікт інтересів.

1.1.2 Неперервні й дискретні

У слова «неперервний», у застосуванні до ігор, є два смисли. Один — у багатьох спортивних-у-буквальному-смислі-слова іграх гравці бігають (змінюють своє положення відносно поля), не «чекаючи кінця ходу іншої сторони», бо ніякого «кінця ходу» взагалі нема. Таке геть не розглядається в цій дисципліні, та й у математичній теорії ігор взагалі (за межами нашої дисципліни) згадується лише іноді й поверхово.

Але є ще один смисл протиставлення неперервних і дискретних ігор. Бувають ігри, в яких і чітко визначено послідовність ходів різних гравців, і ці ходи чітко розділені (поки поточний гравець не закінчить свій хід, інш(ий/і) нічого не роб(и/ля)ть), але фізична взаємодія ігрових об’єктів реального світу (наприклад, спочатку кия і ку́льки, потім ку́льок між собою в більярді, шашок у грі «чапаєв», тощо) складнувата й погано передбачувана. В цих іграх украй важко відділити позиції одна від одної (навіть якщо ясно, що позицією є поточне розташування ку́льок/шашок/…, лишається питання, з якою точністю варто вимірювати ці розташування), і важливою практичною складовою

таких ігор є ризик неточно вдарити по ку́льці/шашці/… й «випадково походити не туди» (отримати не той результат, якого хотів).

Відповідно, послідовними дискретними є ті ігри, в яких і ходи відбуваються послідовно, і позиції чітко відокремлені одна від одної (ідеальний гравець вміє їх гарантовано розрізняти), і наслідки ходу залежать лише від прийнятого гравцем рішення, а не від точності його фізичного втілення.

В рамках нашої дисципліни, будуть розглянуті лише дискретні ігри.

1.1.3 Детерміновані й недетерміновані

Недетермінованість (невизначеність) означає існування зовнішніх випадкових чинників, які впливають на гру попри бажання сами́х гравців. Відповідно, детермінованість (визначеність) — відсутність таких впливів. Часто зовнішні випадкові чинники пов'язані із, наприклад, киданням грального кубика чи якогось аналога.

(Для кращого розуміння суті недермінованих ігор, наголосимо на істотній відмінності між (зазвичай, дитячими) іграми-«блукалками», як-то ця, де все визначається *лише* правилами+полем гри і кубиком, і тими іграми, де випадковість поєднується з рішеннями гравця. Наприклад, у вже згаданих нардах кубики визначають, на скільки позицій гравець може рухати свої ігрові фішки, але часто є вибір, які зі своїх фішок рухати; в іграх «свиня», «1000» та ще багатьох поточна кількість очок поточного ходу визначається кубик(ом/ами), але гравець може приймати рішення, чи завершити хід з цією кількістю очок, чи намагатися повторно кинути кубик(и), що може принести ще більше очок, але може й «згоріти» вже набране; тощо.

З іграми-«блукалками» ця дисципліна зовсім ніяк не пов'язана; такі ігри або взагалі нема смислу вивчати математично, або там потрібні лише засоби теорії ймовірності. А для ігор, де поєднуються випадковість і рішення гравців, можна поєднувати засоби теорії ймовірності, динамічного програмування та тієї самої теорії ігор, яку й вивчає ця дисципліна. Деяке початкове уявлення про це може дати розд. 1.9.)

«Якийсь аналог кубика» буває дуже віддаленим та/або неочевидним аналогом, часом навіть неясно, чи варто вважати гру детермінованою, чи ні. Наприклад, ігри, віднесені у минулому пункті до «послідовних, але неперервних», зазвичай ще й «не зовсім детерміновані», бо на фізичні взаємодії ігрових об'єктів реального світу можуть впливати дрібні нерівності поверхні, вітер та інші аналогічні фактори.

Наразі ця дисципліна зосереджується переважно на детермінованих іграх, лише дуже коротко згадуючи деякі з недетермінованих у розд. 1.9. Однак, досить несподівано, використання псевдовипадкових чисел *все'дно буде(!)* помічним у деяких ситуаціях, що стосуються цілком детермінованих ігор… (зокрема, одна така ситуація описана тут; інша — причому вельми відмінна! — стосується т. зв. «змішаних стратегій» матричних ігор; це входить до навчальної дисципліни, але поки що маємо частину 1 посібника, і в ній цього нема).

1.1.4 Повна і неповна інформація

Хоч щоб було цікаво грати, хоч щоб математично досліджувати гру, природньо, щоб кожен гравець на кожному своєму ході бачив інформацію про поточний стан послідовної гри. Однак, тут слід розрізняти, чи «хоч якусь інформацію», чи «повну інформацію» згідно [6].

Наприклад, доміно та більшість ігор із гральними картами передбачають, що гравець знає свої поточні кісточки доміно чи гральні карти, але не знає кісточок/карт суперник(а/ів).

На противагу іграм попереднього абзацу, шашки, хрестики-нулики, ґо та ще багато ігор надають гравцям повну інформацію. Кожен гравець бачить поточну позицію (як свої, так і чужі фігури/позначки/…), і тому може (знаючи правила гри) не лише визначити варіанти свого ходу безпосередньо тоді, коли вже треба ходити. Крім того, він може (принаймні, теоретично, будучи ідеальним гравцем) прогнозувати «якщо я піду так-то, то супернику дістанеться така-то позиція, в якій у нього будуть такий-то, такий-то і такий-то ходи».

Ця властивість повноти інформації досить хитро переплетена з попередньою властивістю детермінованості; однак, ми не будемо розглядати можливі поєднання, зосередившись головним чином на детермінованих іграх з повною інформацією. Точніше кажучи, дещо згадано в розд. 1.9, але

він і вельми поверховий, і відносно невеликий, а вся решта поточної частини 1 такого взагалі не згадуватиме.

Продовжимо аналіз, що (теоретично…) може робити ідеальний гравець, якщо гра детермінована з повною інформацією. Як вже сказано, він (теоретично…) може бачити варіанти відповіді суперника на кожен свій хід, і враховувати це, вибираючи, як походити. Але це можна продовжити й поглибити.

Звісно, навіть ідеальний гравець не може визначити гарантовано, як походить суперник; на те він і суперник, щоб ходити непідконтрольно хай би як завгодно ідеальному гравцю. Але у детермінованій грі з повною інформацією ідеальний гравець (теоретично…) може будувати різні варіанти розвитку подій: «якщо на такий-то мій хід суперник відповість так-то, то мені дістанеться така-то позиція, з неї в мене будуть такі-то варіанти ходів; звісно, суперник на той самий мій хід може відповісти й так-то, тоді мені дістанеться така-то позиція, з неї в мене будуть такі-то варіанти ходів; …, …, …». Доки йдеться лише про суто теоретичні міркування й ідеальний гравець не обмежений обсягами ні людської, ні комп'ютерної пам'яті, можна вважати, що у детермінованій грі з повною інформацією ідеальний гравець (теоретично…) може від будь-якої позиції «розкрутити» взагалі всі можливі варіанти подальшого розвитку гри.

По суті, це як ще дві аксіоми, які доповнюють наведені у вступі, але дійсні лише в межах послідовних детермінованих ігор з повною інформацією:

- ***У детермінованій грі з повною інформацією, гравці*** (всі; байдуже, ідеальні чи ні) ***вибирають ходи з зарані відомих варіантів, і ці варіанти залежать лише від поточної позиції, й ні від чого більше.***
- ***У детермінованій грі з повною інформацією, ідеальний гравець може від будь-якої позиції «розкрутити» взагалі всі можливі варіанти подальшого розвитку гри.***

Щоб перша зі щойно доданих аксіом була правильною, для досі згаданих ігор варто включати у позицію, крім якогось аналогу розміщення фігур, також інформацію, чий хід: маючи перед очима лише розміщення фігур, гарантовано сказати, чий хід, можна хіба що у хрестиках-нуликах, а також деяких позиціях, дуже близьких до стартової. Однак, у розд. 1.2 (за виключенням його частини 1.2.4) та всьому розд. 1.4 розглянемо такі ігри, для яких, навпаки, вигідно не включати інформацію «чий хід» до позиції; так і виходить, що дати чітке, однозначне й універсальне означення терміна «позиція», на жаль, малореально.

В будь-якому разі, для всіх детермінованих послідовних ігор з повною інформацією з'являється поняття *«граф гри»* у смилі теорії графів: позиції гри стають вершинами графа, ходи — ребрами орієнтованими, тому їх називають також ду́гами). Про саме́ поняття графа див., наприклад, [7], [8, с. 112] та інші джерела; про зв'язок ігор із графами згадано як у [14, с. 9], [15, с. 14], [6] та інших джерелах, так і далі в само́му цьому посібнику (багато де, найдетальніше — тут).

(**Чи є шахи грою з повною інформацією?** Звісно, ніхто не сподівається в рамках цієї дисципліни написати програму, яка справді грала б у шахи (це нереально), тому це питання не має прямих практичних наслідків. Але і цей, і подальший аналогічний аналіз корисні тим, що пояснюють (з точки зору теорії послідовних ігор, а не шахів), чому позиції гри буває треба *розширювати* (доповнювати додатковими параметрами).

У шахах є такий специфічний хід, як рокірування, яке можливе, лише якщо клітинки між королем і відповідною турою вільні (що можна побачити, подивившись на дошку), а також ні король, ні відповідна тура ще не ходили (а це вже видно не завжди: у нехай рідкісному, але можливому випадку, коли король/тура/обоє ходи(в/ла/ли), але поверну(в/ла/ли)ся на початков(е/і) місц(е/я), цього не видно. Тому, для гарантування принципу «позиція визначає перелік ходів із неї» опис позиції шахів повинен включати і розміщення фігур, і, наприклад, ще 7 логічних (`bool`) значень: одне — «чи зараз хід білих?», ще три — «чи ходив білий король?», «чи ходила ліва біла тура?», «чи ходила права біла тура?», і ще три аналогічні для відповідних чорних фігур. Звісно, можуть бути й інші правильні способи задати позицію (наприклад: скоротити з $1+3+3=7$ додаткових `bool`-ів до $1+2+2=5$, зберігаючи замість відповідей про короля, туру й іншу туру відповіді «чи можливе коротке/довге рокірування?»).

А найгірше, що навіть це ще не кінець відповіді на питання «чи є шахи грою з повною інформацією?». Далі буде.)

1.1.5 Щодо скінче́нності ігор; цикли та ациклічні ігри

Є кілька різних ознак, за якими гра може бути скінче́нною чи нескінче́нною. «Найнескінче́ннішою» є гра, що має нескінче́нну кількість позицій, хоча б для деяких позицій має нескінче́нну кількість варіантів ходу з цієї однієї позиції, й може тривати нескінче́нну кількість ходів. Звісно, це — найжахливіший випадок, якщо збираємося програмувати. А найприємніший — коли всі ці характеристики скінче́нні, ще й досить малі́, щоб аналіз усіх варіантів зайняв небагато пам'яті й часу.

Можливі й проміжні ситуації. Наприклад, гра може мати потенційно нескінче́нну кількість позицій, але з кожної з них лише скінче́нну кількість ходів. При бажанні, можна вважати са́ме такою гру Баше з розд. 1.2.1 (одну з найпростіших із усіх розглянутих у цій дисципліні, й потенційна нескінче́нність не заважає їй бути такою). Однак, оскільки наша дисципліна орієнтована на алгоритмічно-програмістський підхід, розглядаємо переважно ігри, скінче́нні в усіх смислах.

Наприклад, шахи скінче́нні як за кількістю позицій, так і за кількістю ходів із позиції, але шахова партія в принципі може тривати вічно, багатократно повторюючи ті са́мі позиції — від «білі походили, чорні походили, білі вернулися на попереднє місце, чорні теж» до значно довших і заплутаніших повторень (також див. тут).

(Якщо говорити ще практичніше, то варто також розрізняти скінче́нні й досить малі (прийнаймні, для комп'ютера) кількості від кількостей формально скінче́нних, але нереально-величезних. Скажімо, для тих са́мих шахів варіантів ходу з конкретної позиції відносно мало (з початкової позиції їх 20, і, наскільки розумію, не буває позицій, з яких більше 100 варіантів ходу), але кількість можливих позицій настільки величезна, що для перегляду їх усих що пам'яті, що швидкодії комп'ютерів не вистачає настільки кардинально, що мабуть не вистачатиме взагалі нико́ли в майбутньому.

Здебільшого, ми розглядатимемо ігри, в яких скінче́нні складові не лише формально скінче́нні, а й практично досить малі (прийнаймні, для комп'ютера). Хоча, будуть також окремі приклади (у різних розд. 1.2:A, 1.2.4, 1.6.4, але всі так чи інак пов'язані з грою Баше), де у грі є певні повторювані шаблони, тож можна легко розглянути, наприклад, кількість позицій 10^{17}–10^{18} (на що ніяк не вистачило би ні пам'яті, ні швидкодії комп'ютерів, якби, як у більшості ігор, розглядали кожну окремо). Також, головною темою розд. 1.3–1.4 є неочевидна теорія, яка дозволяє для певного класу ігор (не всіх!) дуже сильно скоротити кількість позицій, які справді треба розглядати.)

Вже згадано, що з точки зору ідеального гравця кожна детермінована гра з повною інформацією пов'язана з деяким графом, тож доречним є термін теорії графів *«цикл»* (маршрут, що повертається до свого початку; див. [8, с. 120], [9] та інші джерела). Слова́ «початок маршруту» *не* є вимогою повторювати са́ме початкову позицію гри; назва *«гра з циклами»* означає можливість хоча б десь хоча б якось повторно потрапляти протягом гри в позицію, де вже були. Такі ігри мають право на існування й заслуговують на математичний апарат для їх розв'язування. Частково такі засоби розглянуті в розд. 1.6, частково взагалі-то існують у теорії комбінаторних ігор, але поза рамками цієї дисципліни.

Тобто, більша частина матеріалу цієї дисципліни зосереджена на випадку *ациклічних ігор*, графи яких є ациклічними (див. [10], [8, с. 136], …), бо самі правила гри забороняють повторення позицій. Наприклад, якісь палички/камінці/картки забирають, але не додають, чи фішку рухають донизу та/або праворуч, але не вгору й не ліворуч, тощо.

Ситуації, коли ациклічність є, зручні, зокрема, тим, що можна застосовувати деякі прийоми вираження «більших через менші», подібні до зворотної індукції (вона ж обернена індукція чи індукція назад; див. [11]), динамічного програмування та аналогічних табличних технік. Далі по тексту це буде використовуватися дуже багато разів, причому в дещо різних ситуаціях і в поєднаннях з дещо різними іншими засобами аналізу ігор (зокрема, але не тільки: тут, тут, тут, тут, тут, тут, тут, тут), а тут описано прийом, який для багатьох ігор пришвидшує табличну техніку, спираючись на особливості означень виграшності/програшності з розд. 1.2.

(Раз раніше згадували про шахи, продовжимо також і про них. Невже ця популярна гра, яку деякі люди роблять своєю професією, справді може тривати нескінче́нно?

Суто теоретично — так. Практично, са́ме заради боротьби з затягуванням (а отже, і з нескінче́нністю) шахових партій придумали «правило 50 ходів» та «правило потрійного повторення позиції»… Але обидва ці правила придумували люди-шахісти для людей-шахістів, а не математики, що спеціалізуються на теорії ігор. З суто математичної точки зору обидва ці правила жахливі: вони не гарантують завершення гри, а лише надають право-але-не-обов'язок оголошувати закінчення гри внічию, навіть якщо суперник не згоден. По суті, створюють для деяких ситуацій *хід* «завершити гру нічиєю», а гравець сам вирішує, скористатися ним чи ні.

Тому, для гарантування все того ж принципу «позиція визначає перелік ходів із неї» доводиться включати в «розширену позицію», крім «розміщення фігур, “чий хід?” та інформації про можливості рокірування» ще й інформацію про те, скільки було ходів без бою та без руху пішаків, та скільки разів уже повторювалося таке ж поєднання «розміщення фігур, “чий хід?” та інформація про можливості рокірування». **Щоб вважати шахи грою з повною інформацією, доводиться якось аж занадто роздути поняття «розширеної позиції»…**

Власне, тому й доводиться відточувати апарат теорії послідовних ігор зовсім не на класичних загальновідомих шахах, а на значно менш відомих широкому загалу інших іграх.)

1.1.6 Можливі результати ігор

Навіть якщо розглянути лише ігри, які колись та закінчуються (а не тривають вічно), та відкинути найнестандартніші варіанти, лишаються, щонайменше, такі варіанти завершення гри:

(а) за підсумками гри рахують числові виграші кожного з гравців (програші можна вважати від’ємними виграшами);

(б) гра завершується тим, що один з гравців виграв, інший (чи решта) програ(в/ли);

(в) гра завершується або тим, що один з гравців виграв, інший (чи решта) програ(в/ли), або нічиєю.

(Цей перелік не претендує на абсолютну вичерпність. Буває, що розглядають якийсь інший перелік. Однак, це більш-менш вживаний варіант, і в нашій навчальній дисципліні буде так.)

У деяких розділах теорії ігор, не розглянутих у поточній частині 1 посібника, всі ігри мають числові виграші. Але в поточній частині 1 вельми значна частина (розд. 1.2–1.4) розглядає виключно ситуацію «гравців двоє, результат гри — хтось один виграв, інший програв», розд. 1.5 та 1.6 розглядають (по-різному) деякі ігри з нічиїми, а розд. 1.7–1.9 (які разом становлять помітну частину, але все-таки меншість, поточної частини 1) присвячені деяким засобам аналізу послідовних ігор з числовими виграшами.

Виникає питання: а чому так? Очевидно ж, що ігри з числовими виграшами загальніші, бо теоретично правильним є міркування «Навіщо взагалі окремі методи розв’язування ігор з результатом “хтось виграв, інший програв”? Будь-яку таку гру можна переформулювати, перетворивши факт виграшу в числовий виграш (+1), факт програшу в числовий виграш (–1), і розв’язати цю ж задачу як задачу з числовими виграшами…». Однак, тут для вужчої задачі можливі кращі способи розв’язування. Зокрема, але не тільки, засоби розд. 1.4 взагалі ніяк не узагальнити на ігри з числовими виграшами, а засоби розд. 1.2 хоч у деякому смислі й узагальнюються (див. розд. 1.7.1), але при узагальненні змінюються, ускладнюються та стають менш ефективними.

1.1.7 «Нормальна умова» завершення гри

Ця умова стосується лише послідовних детермінованих ациклічних ігор двох гравців з повною інформацією, в яких один гравець виграє й інший програє, без нічиїх і без числового підсумку гри, й передбачає, що *гравець, який не може зробити хід, програє (відповідно, інший виграє)*. Ця умова найактивніше використовується у розд. 1.4 (де вона обов’язкова), але згадується і в багатьох інших місцях.

(Якщо не користуватися засобами розд. 1.4, цю умову можна й порушувати, тобто розглядати ігри, де гравець, який не може зробити хід, виграє, або неможливість зробити хід означає лише завершення гри, а хто виграв залежить від якихось додаткових властивостей «тупикової» позиції, з якої нема ходів. І деякі засоби (не розд. 1.4, а інші) можуть під це підлаштовуватися. Такі приклади розглянуті, зокрема, в розд. 1.2.3 та 1.3.3.)

1.1.8 Неупереджені (impartial, безсторонні, непартійні) та упереджені (partisan, партизанські, партійні) ігри

Згідно [12], гра *неупереджена* (*безсторонняя*, *impartial*), коли ***і*** *заміна в поточній позиції самої лише властивості «чий хід?» ніколи не змінює сукупність можливих ходів,* ***і*** *виграші симетричні*. Відповідно, гра *упереджена* (*partisan*, *партизанська*, *партійна*), коли *хоча б для деяких позицій*

*заміна самої лише властивості «чий хід?» змінює сукупність можливих ходів, **та/або** виграші різних гравців рахуються по-різному*.

(Сумно, що як у джерелі [13], так і в багатьох офіційніших джерелах упередженість описують як «гра упереджена, коли хоча б для деяких позицій заміна самої лише властивості “чий хід?” змінює сукупність можливих ходів», хоча взагалі-то з означення неупередженої гри, твердження «кожна гра або неупереджена, або упереджена» та законів математичної логіки випливає також і твердження «коли єдиною несиметричністю є різні функції виграшів, це теж робить гру упередженою». Прикладом ситуації, де це справді важливо, є задача 1.7:D та ще деякі пов’язані з нею задачі розд. 1.7.3. Водночас, у більшості випадків причиною упередженості ігор справді є різні ходи, а не різні функції виграшу, й це *частково* пояснює цю невідповідність. Ще одне пояснення цієї невідповідності — різні функції виграшу просто неможливі, коли гра сформульована так, що неминуче один гравець виграє, інший програє; аналогічно й коли гра може завершуватися внічию, але, якщо не внічию, то один гравець виграє, інший програє. Багато в яких джерелах просто розглядають лише такі ігри, а для них цієї невідповідності нема.)

Більшість реально поширених ігор упереджені: в хрестиках-нуликах 1-й гравець може ставити лише “X”, 2-й лише “O”; у шашках чи шахах 1-й гравець може рухати лише білі фігури, 2-й лише чорні.

(Гравець може бити чужу фігуру, але навіть цей випадок, хоч і включає фізичне переміщення чужої фігури за межі дошки, все ж вважається ходом своєю фігурою. А ходи, де гравець лише переміщує чужу фігуру, заборонені.)

Неупереджених ігор багато в рамках цього посібника, зокрема у розд. 1.2 (за виключенням його частини 1.2.4), розд. 1.7.1 та всьому розд. 1.4.

(Однак, більшість ігор з цих розділів якісь специфічні; чи є приклади значно відоміших широкому загалу неупереджених ігор? Важко сказати, і тому, що справді мало таких ігор, і тому, що неясно, що таке «відома широкому загалу». Я не зміг пригадати жодної дискретної детермінованої неупередженої гри, про яку взнав не при вивченні теорії ігор як математичної дисципліни, а, наприклад, у дитинстві. Однак, серед послідовних неперервних (які потребують точної/акуратної/… фізичної взаємодії з ігровими об’єктами в реальному світі), дещо таке пригадати вдалося (втім, не факт, чи «широкий загал» визнає їх відомими, тим паче, що знайти їх у вікіпедії не вдалося).

Наприклад, (зовсім дитяча) гра «Туалет» (де зазвичай використовують картонні гральні карти, але це могли б бути будь-які майже плоскі фігури зі схожими фізичними властивостями, зокрема співмірними тертям і жорсткістю щодо згинів) має такий сюжет: спочатку більшість карт викладають деяким безсистемним чином, у межах, дуже приблизно кажучи, круга діаметром 30–40 см, у кілька шарів (слоёв), щоб деякі верхні частково(!) накривали собою деякі нижні, потім поверх цього останні 2–4 карти ставляться так, щоб стояли, спираючись одна на о́дну (ото і є «туалет», тільки не питайте, чому; це цитата з дитячої народної творчості, а не моя думка). Це «початкова позиція» (яка в різних партіях реально виходить трошки різною). Далі гравці почергово повинні витягувати однією рукою по одній карті (з тих, що безсистемно викладені в кілька шарів) за хід, причому гравець сам вирішує, яку карту тягти, й обмежень, подібних до «тягти зі свого боку», *нема*. Програє той гравець, на чиєму ході завалюється «туалет» або витягується більше однієї карти. Можна довго сміятися над цією грою, але це все-таки приклад послідовної неупередженої гри: чий би не був хід, перед поточним гравцем стоїть та са́ма задача, і, маючи конкретний поточний стан гри («позицію»), чий би не був хід, вибирати, яку карту і як тягти, можна з одних і тих самих варіантів.

Ще смутно пригадується одна гра, де на одному з етапів треба ударами добиватися перевертання монет: поки гравцю це вдається, він забирає перевернуту монету собі й може далі вдаряти по іншим монетам; коли не вдається — хід переходить наступному гравцю, й він має ту саму мету й може вибирати з тих самих ходів, що й робить цю гру (точніше, цей її етап і в тому вигляді, як я це пам’ятаю) неупередженою. Від детальнішого опису утримаюся і тому, що деяких деталей не пам’ятаю, і щоб не наводити розлогий опис азартної гри.)

Властивість неупередженості (як і гри са́ме двох гравців, дискретності, скінче́нності, ациклічності, детермінованості та повної інформації, а також «нормального завершення») украй важлива для засобів, розглянутих у розд. 1.4. Й ці засоби справді досить цікаві, потужні й неочевидні, щоб заради них вводити аж стільки обмежень на властивості гри.

1.2 Виграшні та програшні позиції

В межах цього розд. 1.2, всі ігри будуть іграми двох гравців, послідовними, дискретними, детермінованими, скінче́нними&ациклічними, з повною інформацією, закінчуватися виграшем одного з гравців і програшем іншого (без нічиїх і без числового вираження виграшу), але не всі ігри цього розд. 1.2 будуть неупередженими та не всі матимуть «нормальну умову завершення».

Враховуючи ці обмеження, можна запровадити означення виграшних та програшних позицій.

- ***Позиція гри програшна, коли або тим фактом, що гравцю дісталася ця позиція, він вже програв, або абсолютно кожен хід з неї веде до виграшної позиції (яка дістанеться суперникові).***
- ***Позиція гри виграшна, коли або тим фактом, що гравцю дісталася ця позиція, він вже виграв, або існує хоча б один (можна й більше) хід з неї, що веде до програшної позиції (яка дістанеться суперникові).***

Важливо, що в означенні програшної позиції говориться про *всі* ходи, а в означенні виграшної про *хоча б один* хід. Якщо дісталася програшна позиція, то навіть ідеальний гравець не зуміє нічого з цим вдіяти, щоб і гарантовано, і не порушивши правил гри. Водночас, отримання виграшної позиції гарантує виграш — близький чи далекий — ідеальному гравцю, але нічого не гарантує невмілому, який може як вгадувати, ходити вдало й кінець кінцем виграти, так і програти внаслідок невдалого свого ходу (помилково зробив не «виграшний хід», після якого супернику дісталась би програшна позиція, а інший хід, де супернику дісталася виграшна позиція, й суперник — ідеальний чи ні — зумів цим скористатись).

Неважко переконатися (але ми це залишимо без доведення), що за умов, перелічених у найпершому абзаці цього розд. 1.2, (а також стандартного для всієї теорії ігор припущення, що аналіз відбувається з точки зору ідеального гравця) кожна позиція буде або виграшною, або програшною (не може бути ні водночас і виграшної, і програшної, ні такої, яка не є ні виграшною, ні програшною).

У деякій літературі (наприклад, але не тільки, у [14], тобто тут) наші виграшні позиції називають *N-позиціями*, а наші програшні — *P-позиціями*, з тих міркувань, що в N-позиції свій виграш теоретично забезпечує (будучи ідеальним) наступний (**N**ext) гравець, який ходитиме із цієї позиції, а в P-позиції — попередній (**P**revious) гравець, який походи(в/ть) у цю позицію. Однак, з не дуже зрозумілих мені причин, надто багато джерел (зокрема й щойно згадане [14]) штучно звужують апарат використання виграшних та програшних позицій, безпідставно вимагаючи, крім вказаних на початку цього розд. 1.2, ще й «нормального закінчення» та неупередженості. Тому варто розуміти, що засоби цього розділу давно описані в літературі (а не є оригінальним дослідженням автора посібника), але є проблема трохи різних підходів до цього питання, трохи різних варіантів термінології, тощо, і в цих питаннях іноді доводилося вибирати один з варіантів самостійним волюнтаристським рішенням.

1.2.1 Гра Башé — теорія

(Цю гру часто ніяк не називають (чи називають «Палички» або «Камінці»); це пов'язано з різними думками, чи вважати істотною роль Клода Гаспара Баше де Мезір'яка в аналізі цієї гри. Але суть гри від того не змінюється. Не змінюється вона й від того, чи в купці палички, чи камінці, чи ще якісь предмети, тож нехай це будуть палички.)

Самá гра така: «Є одна купка, яка спочатку містить N паличок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з купки або 1, або 2, або 3 палички (але, звісно, не більше, чим їх є в купці). Інших варіантів ходу нема. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом із ще однією або ще двома)».

Як бачимо, ця гра неупереджена: гравці мають однакову (але кожен щодо себе, тому вони суперники, а не спільники) кінцеву мету забрати останню паличку, й однаково можуть забирати з купки або 1, або 2, або 3 палички (якщо їх стільки є). Це означає, що в цій грі невигідно (надлишково, створює ускладнення без усякої користі) включати «чий хід?» у позицію, краще вважати позицією самý лише кількість паличок N.

Гравець, якому дісталася будь-яка з позицій $N=1$, $N=2$ або $N=3$, може одним ходом забрати всі палички й цим відразу виграти. Отже, всі ці позиції виграшні.

Гравець, якому дісталася позиція $N=4$, може походити лише або в позицію $N=3$ (забравши 1 паличку), або в $N=2$ (забравши 2), або в $N=1$ (забравши 3). Кожна з трьох позицій (3, 2 та 1) виграшна. Тобто, абсолютно кожен хід з позиції $N=4$ веде до виграшної позиції (яка дістається супернику). Тобто, позиція $N=4$ програшна.

Міркування попереднього абзаца можна переформулювати «якщо я піду 1, суперник може відповісти 3; якщо я 2, суперник 2; якщо я 3, суперник 1; інших варіантів нема; отже, якщо моя

позиція 4, суперник завжди може забрати все й виграти»; попервах це навіть може здаватися природнішим формулюванням. Однак, оскільки як теорія послідовних комбінаторних ігор взагалі, так і конкретні означення виграшних та програшних позицій передбачають погляд з точки зору ідеальних гравців (а отже — можливість дивитися на *багато* ходів уперед), «правильніше» дивитися суто на «виграшність»/«програшність», бо так легше перейти до аналізу позицій, від яких до кінця гри багато ходів.

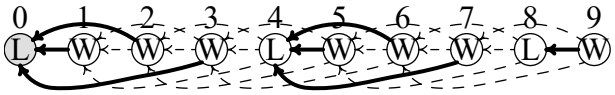
Зокрема, при аналізі позицій $N=5$, $N=6$ та $N=7$ можна й варто не розкручувати послідовності ходів аж до кінця гри, а сказати, що кожна з них виграшна, бо з кожної з них існує хід у програшну позицію $N=4$. Потім, аналізуючи позицію $N=8$, можна дійти висновку про її програшність на підставі самого лише факту, що з неї можна піти лише в $N=7$, $N=6$ або $N=5$, а вони всі виграшні. Пояснення, чому вони виграшні, для висновку про програшність $N=8$ неважливе. І так далі.

Чи варто вважати $N=0$ позицією, і якщо вважати, то програшною чи виграшною? Мабуть, варто, хоч би тому, що остання одна чи останні дві чи останні три палички за правилами гри все ж забираються, отже їх забирання зручно важати ходом, а хід переводить гру з позиції в позицію. А що якщо цю позицію розглядати, то слід вважати її програшною — тут вже однозначно (без усяких «мабуть»): по-перше, «кому дістається ця позиція (з якої нема ходів), той програ́є» якраз і є «нормальною умовою» завершення (для поточного розд. 1.2 дотримуватися «нормальної умови» не обов'язково, але ж і не заборонено); по-друге, якраз виходить, що з виграшних позицій $N=1$, $N=2$ та $N=3$ існує хід у програшну позицію $N=0$.

Таким чином, за результатами аналізу гри Баше вийшло, що програшними є всі ті й лише ті позиції, де $N \vdots 4$ (де "$\vdots$" — «кратне», воно ж «ділиться націло на»), а решта позицій є виграшними. Більш того: для кожної виграшної позиції відомий той хід (з N наявних у купці паличок, забрати $N \bmod 4$, де $N \bmod 4$, відоме також як $N\%4$, є залишком від ділення N на 4), який приводить у програшну позицію, яка дістається супернику. ■

Однак, це *лише* для гри Баше вийшли конкретні простенькі формули; так буде далеко не завжди.

Зобразимо також граф цієї гри, на якому відзначимо виграшні (`W`, від **W**in) та програшні (`L`, від **L**oose) позиції, та стрілки, що відображають ходи; жирним позначимо «виграшні ходи» (які ведуть з виграшної позиції до програшної, що дістанеться супернику), пунктиром — решту ходів. Сірим кольором виділено вершину, де гра закінчується (в цій грі вона єдина, але в інших іграх їх може бути кілька).



Чому цей граф зображено са́ме для $N=9$? Просто для прикладу. І взагалі, такі графи зазвичай не є якось остато́чною відповіддю. Користь таких графів зазвичай або в тому, щоб, подивившись на нього, помітити якусь специфічну для са́ме цієї задачі закономірність і завдяки ній придумати специфічне для са́ме цієї задачі розв'язання (що са́ме в цій задачі вже зроблено), або в тому, щоб зручно співставляти з ним код, який по суті аналіз такого графу й реалізовуватиме (що са́ме в цій задачі не потрібно). Однак, краще навести його зараз для цієї простої гри, чим щоб цей інструмент потім уперше з'являвся у складнішій ситуації.

Задача 1.2:A. «Гра Баше–1»

(Правила гри див. на початку розд. 1.2.1.)

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Єдине ціле число N ($1 \leqslant N \leqslant 12345$) — початкова кількість паличок у купці.

Результати. Єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо дру́гий).

Приклади:

Вхідні дані	Результати
2	1
25	1
256	2

Розбір задачі. По суті все вже розібрано на початку розд. 1.2.1: у програмі слід лише прочитати N, перевірити, чи $N \bmod 4 = 0$, і якщо так, то вивести 2 (раз початкова позиція програшна, то 1-й гравець не може забезпечити собі виграш, а 2-й (будучи ідеальним) може), інакше 1 (раз початкова позиція виграшна, то 1-й гравець (будучи ідеальним) може забезпечити собі виграш).

Задача 1.2:B. «Гра Баше — інтерактив»

(Правила гри див. на початку розд. 1.2.1.)

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Ця задача є інтерактивною: Ваша програма не отримає всіх вхідних даних на початку, а отримуватиме по мірі виконання доуточнення, що залежатимуть від попередніх дій Вашої програми. Тим не менш, *її перевірка буде автоматичною*. Тому, слід чітко дотримуватися формату спілкування з програмою, яка грає роль суперника.

Протокол взаємодії. На початку, один раз, Ваша програма повинна прочитати одне ціле число в окремому рядку — початкову кількість паличок N ($1 \leqslant N \leqslant 12345$). Потім вона повинна повторювати такий цикл:

1. Вивести єдине число в окремому рядку — свій хід, тобто кількість паличок, які вона зараз забирає з купки. Це повинно бути ціле число від 1 до 3, причому не більше за поточну кількість паличок у купці.
2. Якщо після цього купка стає порожньою, вивести окремим рядком фразу "`I won!`" (без лапок, символ-у-символ згідно зразку) й завершити.
3. Інакше, прочитати хід програми-суперниці, тобто кількість паличок, які вона зараз забирає з купки (єдине ціле число, в окремому рядку). Гарантовано, що хід допустимий (є цілим числом від 1 до 3 і не перевищує поточного залишку паличок у купці). Само собою, ця гарантія дійсна лише за умови, що Ваша програма правильно визначила, що гра ще не закінчилася.
4. Якщо після цього купка стає порожньою, вивести фразу "`You won...`" (без лапок, символ-у-символ згідно зразку) й завершити роботу.

Все це слід повторювати, доки не будуть забрані всі палички (тобто, доки якась із програм-гравців не ви́грає). Програма-суперниця не виводить фраз "`I won!`" / "`You won...`" чи якихось їх аналогів.

Приклад:

Вхід, суперник	Ваша програма
5 2	 1 2 I won!

Примітки. Ніби порожні рядки між різними ходами у прикладі зроблені, щоб було видно, хто коли ходить; вводити/виводити їх не слід.

Загальний хід гри з прикладу можна прокоментувати так. У купці спочатку 5 паличок. Ваша програма забирає одну, лишається 4; програма-суперниця забирає дві, лишається 2; Ваша програма забирає обидві, повідомляє про свій виграш і завершує роботу.

Оцінювання. У приблизно половині тестів Ваша програма матиме справу з ідеальною програмою-суперницею, яка не робить помилок. У іншій приблизно половині — з різними програмами-суперницями, які грати не вміють — тобто, роблять лише ходи, які дотримуються формальних вимог «забирати лише від 1 до 3 паличок» та «забирати не більше паличок, чим реально є у купці», але можуть вибирати не найкращий з допустимих ходів, дотримуючись кожна своїх власних

уявлень про те, як варто грати в цю гру. Буде оцінюватися як уміння Вашої програми виграти там, де це гарантовано можливо, так і вміння Вашої програми гідно, дотримуючись правил гри, програти, так і вміння Вашої програми скористатися (теж згідно правил) помилками чи іншими неадекватностями програми-суперниці, якщо такі будуть.

Тобто, щоб отримати абсолютно повні бали, Ваша програма має врахувати все щойно перелічене.

За будь-яке порушення правил гри з боку Вашої програми, відповідний тест оцінюватиметься як не пройдений.

Розбір задачі. Гра розібрана в розд. 1.2.1 і запрограмована в попередній задачі, тож ніби природно, щоб лишалися тільки суто технічні моменти: підраховувати, скільки паличок лишилося; організувати цикл, що контролює їхню кількість; виводити свої ходи й читати ходи програми-суперниці…

(До речі, із введенням/виведенням *можуть* бути проблеми: важливо виводити символ переведення рядка після свого ходу, читати хід програми-суперниці разом з переведенням рядка, вводити з консолі й виводити на консоль без використання файлів. Для мови C# усе досить добре: очевидні `Console.ReadLine(...)` та `Console.WriteLine(...)` працюють правильно; однак, з деякими іншими мовами програмування ситуація гірша. Якщо порушувати вимоги цього абзацу, може виникнути, наприклад, ситуація, коли система автоматичної перевірки не передає ходи Вашої програми до програми-суперниці. Причому, система перевірки навряд чи скаже щось корисне, найімовірніше буде вердикт «Перевищено ліміт часу роботи», що може спонукати до зовсім неправильних дій. ***Дуже важливо не(!) пробувати пришвидшити введення/виведення якою б не було буферизацією***, яка прискорює введення/виведення в багатьох інших ситуаціях, але тут лише збільшує ймовірність, що ходи не передаються між програмами, «застряючи» якраз-таки в буфері.)

Але тут можуть бути й ідейні складнощі! Питання «що робити, якщо дісталася програшна позиція?» досі взагалі не розглядалося. Само собою, треба не забути написати обидві гілки відповідного розгалуження (якщо позиція виграшна, тобто `n%4!=0`, то забрати `n%4` паличок, інакше, коли позиція програшна, просто якось походити). Але треба ще вирішити, *як са́ме «якось»* ходити. Якби програма-суперниця була завжди ідеальною, з цим було б легше, бо було б відомо, що раз ми були у програшній позиції й мусили віддати програмі-суперниці виграшну, то вона ви́грає, а наша програма, що не роби, програ́є… Але в задачі сказано зокрема й перехоплювати ініціативу в неідеальних програм-суперниць!

Може здатися природньою ідея «у програшній позиції завжди забирати 1 паличку, бо це відтягує кінець гри й тому збільшує ймовірність помилки неідеальної програми-суперниці»… Але це не так. Одна з «неідеальних програм-суперниць» тупо намагається «забрати якнайбільше паличок (якщо їх кількість $\leqslant 3$, то всі; інакше, 3)». І наша програма, замість «відтягти кінець гри», програє аж настільки тупій суперниці.

Очевидно, кожен з варіантів «будучи у програшній позиції, завжди забирати 2 палички» та «будучи у програшній позиції, завжди забирати 3 палички» має аналогічну ваду: наша програма програватиме іншим, але теж дуже тупим програмам-суперницям «намагається завжди брати 2 палички (але якщо лишається 1, то забирає 1)» та «абсолютно завжди бере рівно 1 паличку» відповідно.

То що ж робити, якщо розкритикували і варіант «бра́ти 1 паличку», і варіант «бра́ти 2 палички», і варіант «бра́ти 3 палички», а інших варіантів нема?

Доречність випадкових чисел у детермінованій грі. На питання, яким закінчили попередній пункт, може бути чимало різних відповідей, але тут наведемо одну — водночас і напрочуд просту, і дещо несподівану: *варто використати випадкові чи́сла*. Звісно, random слід використовувати *лише у програшних* позиціях; у виграшних слід діяти згідно з основним аналізом гри, тобто забирати `n%4` паличок і цим віддавати супернику програшну позицію. Звісно, random не допоможе, коли програма-суперниця є ідеальним гравцем. Але random — цілком гідний засіб не потрапити в пастку несвідомого підігрування конкретній реалізації неідеальної програми-суперниці, та й проти підвладних помилкам розумних гравців (зокрема й людей, але не тільки) random теж непогано працює.

(Ймовірність, що навіть випадкові числа все’дно підіграють дуже тупій програмі-суперниці, теоретично існує, але практично нереально-низька: якщо хоч один раз «не підіграли», то наша програма перехопить виграшну позицію й,

дотримуючись правильної стратегії «забирати `n%4` паличок», виграє; щоб цього не сталося, треба кожнісінького разу підіграти; якщо для випадкового вибору й дуже тупої програми-суперниці ймовірність підіграти становить $\frac{1}{3}$, то за $\frac{n}{4}$ ходів ймовірність підіграти щоразу становить $(\frac{1}{3})^{n/4}$, що, наприклад, при $n=20$ становить $\frac{1}{3^5} \approx 4 \cdot 10^{-3}$, а при $n=100$ становить $\frac{1}{3^{25}} \approx 10^{-12}$.

Звісно, сумнівним місцем цих міркувань є «а звідки ми знаємо, що ймовірність випадково підіграти тупій програмі становить са́ме $\frac{1}{3}$?». Поки невідомо, що таке «тупа програма-суперниця», то ми цього й не знаємо. Але конкретно для вищезгаданих тупих програм-суперниць (одна завжди бере 1, інша намагається 2, третя намагається 3) там справді $\frac{1}{3}$ (якщо тільки не спіймаєте описаний далі гайзенбаг чи щось схоже).

Навіть на це все ще можна відповісти «ну а раптом са́ме мені аж так не пощастить, що мала ймовірність все ж реалізується? тим паче, при $n=4$ чи $n=8$ ймовірності $\frac{1}{3} = 0{,}(3)$ чи $\frac{1}{9} = 0{,}(1)$ не такі й малі?». Що ж, це практично повністю вирішується тим, що «перехоплення ініціативи» фактично перевіряється лише на відносно великих розмірах $n \geqslant 40$, де ймовірність випадково підіграти вже не перевищує $\frac{1}{3^{10}} \approx 1{,}7 \cdot 10^{-5}$. (Це ніяк не протирічить, наприклад, тесту з умови $n=5$, бо попереднє речення було лише про «перехоплення ініціативи», і для *таких* тестів $n \geqslant 40$ дотримано; а для тестів, де треба вигравати, стартуючи з виграшної позиції, чи гідно програвати, стартуючи з програшної, n може бути малим, там так навіть легше.) Крім того, є можливість здати програму ще раз (і ще 2, 3, … рази), а рахується ж найкращий результат.)

Просто прикольний гайзенбаг. (Деякими мовами програмування[1] виявляється важливим ініціалізувати псевдовипадковість один раз, а не безпосередньо перед генерацією кожного числа. Як видно на `ideone.com/o0hJKF`, якщо багато разів підряд ініціалізувати псевдовипадковість і генерувати псевдовипадкове число, результати виходять, як правило, *однакові(!)* підряд. Що, насправді, логічно, якщо розуміти, як це працює: `srand(time(NULL))` задає ініціалізацію генератора псевдовипадкових чисел залежно від поточного часу, що зазвичай означає, що при різних запусках час буде різним, а тому різними будуть і псевдовипадкові послідовності. Але виклики багатьох підряд ініціалізацій, коли програма виконується автоматично без debug'а і без звернень до користувача-людини виявляються досить швидкими, отже досить близькими за часом, щоб із величезною ймовірністю траплялося, що генератор ініціалізується щоразу однаково, й тому дає те са́ме число.

«Найвеселіше», що, не знаючи, досить важко визначити/локалізувати/… цю проблему: коли пробувати все це дебажити, то, найімовірніше, проміжки часу стануть великими, генератор переініціалізовуватиметься щоразу інакше, і даватиме, як правило, інше число (випадкові збіги можливі, але низькоймовірні), а якщо знову запустити автоматично — проблема знову вертається. Таку ситуацію називають також гайзенбаг, від слів баг (як помилка програми) та Гайзенберг (автор принципу невизначеності, з яким пов'язаний також ефект спостерігача). Зв'язок між цією ситуацією й Гайзенбергом у тому, що поява цього багу в автоматичному виконанні програми й зникнення при debug'у якраз і є ефектом спостерігача.

Можете взяти згаданий код `ideone.com/o0hJKF` і спробувати взагалі прибрати `srand` (масових повторів не буде, але послідовності будуть однакові при різних запусках; конкретно для цієї задачі це прийнятно, для багатьох інших застосувань псевдовипадкових чисел неприйнятно) та/або перемістити його туди, де йому й місце — на початок функції `main`.)

1.2.2 Фішка на мінному полі

Задача 1.2:C. «Фішка на мінному полі — 1»

На прямокутному полі $N \times M$ клітинок у лівому верхньому кутку стоїть фішка. Ця кутова клітинка гарантовано вільна (не замінована); абсолютно кожна інша клітинка по́ля може бути хоч вільною, хоч замінованою. Гра полягає в тому, що два гравці поперемінно рухають згадану фішку на якусь кількість клітинок або праворуч, або донизу (кожен гравець сам вирішує, в якому з цих двох напрямків і на скільки клітинок рухати; не можна ні лишати фішку на місці, ні ставати нею в замінування клітинку, ні перестрибувати нею через замінування клітинку). Програє той, хто не може нікуди походити (і знизу, і праворуч або край по́ля, або міна). Відповідно, його суперник виграє.

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Перший рядок містить два цілі числа́ N та M, розділені одним пропуском (пробілом) — спочатку кількість рядків, потім кількість стовпчиків. Обидва ці значення у межах від 1 до 12.

[1] зокрема, але не тільки, C++; цієї проблеми начебто нема ні для C#, ні для Java, але ситуація досить цікава, щоб вивчити її для загального розвитку, навіть якщо вона не становить для Вас безпосередньої технічної проблеми

Далі йдуть N рядків, що задають мінне поле. Кожен з них містить рівно по M символів `.` (позначає вільну клітинку) та/або `*` (позначає заміновану клітинку). Ці символи йдуть без роздільників, і кожен з цих N рядків містить лише ці символи та переведення рядка наприкінці.

Результати. Єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо дру́гий).

Приклади:

Вхідні дані	Результати
2 4 .... .**.	1
1 1 .	2

Примітка. У першому прикладі, перший гравець може, наприклад, піти на одну клітинку вниз, після чого дру́гому не буде куди йти, і він програ́є. Якби перший гравець сильно помили́вся і пішов на першому кроці на три клітинки праворуч, то дру́гий пішов би на одну вниз і виграв. Але це значило б, що перший грає неправильно, а питають про ситуацію правильної гри обох гравців.

У дру́гому прикладі, першому гравцю відразу ж нема куди йти, і він негайно програ́є (відповідно, виграє́ дру́гий гравець). Звісно, з точки зору загальнолюдських уявлень це якась дуже нечесна гра… Але вже яка є, описані обмеження формально дотримані.

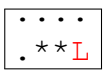
Розбір задачі. Оскільки гра неупереджена, а гравці можуть лише переміщувати одну на двох фішку, позиції гри відрізняються одна від іншої лише тим, де (в якій клітинці) розміщено фішку. Завдяки чому, в цій задачі позиціями можна і варто вважати клітинки по́ля, причому не геть усі, а лише незаміновані, куди дозволено ставити фішку.

Потрібно реалізувати поєднання означення виграшних і програшних позицій з початку поточного розд. 1.2 та деякої табличної техніки, подібної до зворотної індукції та динамічного програмування.

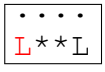
Щоб при знаходженні значень (ви/про)грашності ще не відомих клітинок-позицій завжди спиратися на вже відомі, можна (враховуючи, що ходи робляться лише униз та праворуч) почати з правого-нижнього кутка по́ля, й іти рядок за рядком «назустріч ходам».

(Можлива й інша реалізація — рекурсія з запам'ятовуваннями (memoized recursion), яка починається зі стартової позиції, далі все відбувається завдяки рекурсії. Про memoized recursion та причини, чому потрібні запам'ятовування (а «проста» рекурсія працюватиме жахливо довго) можна прочитати, наприклад, у матеріалах дисципліни «Алгоритми та структури даних» ([16, с. 52–54]), або, наприклад, це обговорення на stackoverflow.com. Для більшості ситуацій, зокрема й для цієї гри, спосіб memoized recursion складніший і тому недоречний. І далі зосереджуємося не на рекурсії, а на способі «йти "назустріч ходам"». Але про ситуації, де рекурсія із запам'ятовуванням може бути доречною, скажемо. По-перше, бувають заплутані серії підзадач, де важко розібратись, у якому порядку їх розв'язувати, щоб завжди спиратися лише на вже розв'язані. Рекурсія із запам'ятовуваннями дозволяє не думати над цим, під час виконання само вийде, що для вже розв'язаних підзадач будуть взяті готові відповіді, а ще не розв'язані будуть розв'язані рекурсивно. По-друге, бувають задачі, при рекурсивному розв'язуванні яких можна бачити, що деякі підзадачі насправді не потрібні (гарантовано гірші за інші вже розв'язані), і взагалі не розв'язувати їх, а при ітеративній реалізації цього не видно. В цьому (й лише цьому) випадку рекурсія з запам'ятовуваннями може бути навіть швидшою, чим ітеративний спосіб.)

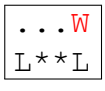
Розглянемо спосіб зворотної індукції (на циклах «назустріч ходам», без рекурсії) на прикладі.



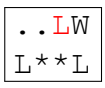
Ходів із правої-нижньої клітинки не існує; згідно правил гри та означення програшної позиції, це означає, що відповідна (позначена червоним) клітинка-позиція є програшною.



Для лівої-нижньої клітинки ситуація аналогічна, але тепер враховуючи і те, що знизу межа по́ля, і те, що праворуч міна.



З правої-верхньої клітинки існує хід униз у програшну позицію, значить сама́ права-верхня клітинка-позиція є виграшною.



Із виділеної кольором клітинки існує лише хід на 1 праворуч, і цей хід веде до виграшної позиції. Отже, всі ходи (цей хід і є всіма, раз інших нема) ведуть у виграшні позиції (фактично одну, але це неважливо), що й означає, що поточна клітинка-позиція програшна.

```
.WLW
L**L
```

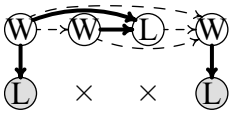
Із виділеної кольором клітинки є два можливі ходи (на 1 праворуч і на 2 праворуч), один з цих ходів веде до програшної — отже, поточна клітинка-позиція виграшна.

Якщо потрібні лише висновки про (ви/про)грашність позицій (а повні переліки ходів не потрібні) — ***можна обривати цикл, знайшовши першу програшну позицію***, куди є хід: цього досить, щоб оголосити цю позицію виграшною, й це вже не зміниться.

```
WWLW
L**L
```

Із лівої-верхньої клітинки є чотири можливі ходи (на 1 праворуч, на 2 праворуч, на 3 праворуч, на 1 униз), серед них є два ходи (на 2 праворуч та на 1 униз), що ведуть до програшних позицій; отже, сама́ поточна клітинка-позиція виграшна.

Що ходів до програшних позицій аж два — приємно, а також збільшує ймовірність виграшу неідеального першого гравця. Але наразі, з точки зору формального аналізу для ідеальних гравців, це не важливо: досить якогось одного з цих ходів, дру́гий ніц не змінює.



А ось відповідний граф гри.

Тут і далі, я вважаю, що читачі володіють достатніми навичками алгоритмізації, щоб після досі поясненого самостійно доформулювати алгоритм і запрограмувати його.

Задача 1.2:D. «Фішка на мінному полі — 2»

Правила гри та формат вхідних даних відповідають задачі 1.2:C «Фішка на мінному полі—1», але максимальні розміри кожного з вимірів становлять 1234.

Результати. Перший рядок має містити єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо дру́гий). Якщо відповідь з першого рядка `2`, то на цьому виведення слід припинити. А якщо відповідь з першого рядка `1`, то далі треба вивести також перелік всіх можливих перших ходів першого гравця, після яких дру́гий (при правильній грі першого) вже ніяк не зможе виграти. Цей перелік виводити в такому форматі: спочатку, якщо існує виграшний хід униз, то вивести його як велику латинську літеру `D`, одинарний пропуск (пробіл) та довжину ходу (на скільки клітинок іти вниз); потім, якщо існує виграшний хід направо, то аналогічно, але на початку велику латинську `R`. (Заодно *доведіть*, що в цій задачі не може бути багатьох виграшних ходів униз чи багатьох виграшних ходів направо.)

Приклади:

Вхідні дані	Результати
2 4 .... .**.	1 D 1 R 2
1 1 .	2

Примітка. Чому виграшними ходами для першого прикладу є са́ме ці два й ніякі більше, випливає з цього графу гри та пояснень перед ним.

Розбір задачі. Майже все вже є в розборі задачі 1.2:C; і потребу запам'ятати й вивести ті програшні позиції, ходи в які роблять початкову позицію виграшною (якщо, звісно, такі ходи є), і додаткове завдання довести, що в цій задачі не може бути різних виграшних ходів донизу та/або різних виграшних ходів праворуч, оголошую простими й виношу на самостійне виконання читачами.

Але збільшення максимального розміру по́ля підвищує вимоги до ефективності.

Яку сумарно кількість варіантів треба розглянути, щоб для кожної клітинки подивитися всі клітинки, куди з неї можна піти? Якби гра була такою самою, але без мін, то існувало б у середньому $\frac{M+N-2}{2} \approx \frac{M+N}{2}$ ходів з клітинки (з лівої-верхньої $(M-1)+(N-1)$, з правої-нижньої 0, сумарно $M+N-2$, і так для кожної пари клітинок, симетричних відносно центру по́ля). Звідси, загальна кількість ходів, які (ніби…) треба передивитися, становить $\approx {}^1\!/_2 \times MN \times (M+N)$. При $M=N=$ $=1234$ це становить $\approx 1234^3 \approx 1{,}9 \cdot 10^9$, що, навіть для комп'ютера, дещо забагато.

На це ніби й можна казати «Коли мін нема, задачу можна розв'язати простішим способом, а коли міни є, кількість ходів зменшується, бо через міни не можна перестрибувати»… Але в цій задачі такі

міркування ніц не дають. Розглянемо знову поле максимальних розмірів, тепер з *кількома* (наприклад, 5-ма чи 10-ма) мінами. Тут водночас і міни є (тож міркування для випадку без мін незастосовні) і лишаються майже всі рядки та майже всі стовпчики, *в межах яких* мін усе-таки нема, тож для них лишаються правильними майже ті самі міркування, й загальна кількість ходів зменшується хіба на якісь 0,1–1%, тобто майже така ж.

Міркування «Знайшовши перший хід у програшну позицію, можна обривати перебір ходів» дає трохи більше, але не набагато.

(Аналітично в мене вийшло, що скорочення утричі — оптимістична оцінка (фактичне скорочення може бути меншим, як-то «удвічі» чи «на третину»); однак, впевненості у правильності цієї оцінки нема. Водночас, крім міркувань в мене є результати вимірювань часу роботи програм, і вони теж свідчать, що сама лише оптимізація «обривати цикл» не рятує ситуації по-справжньому на вхідних даних вигляду «розміри великі, міни є, але мало».)

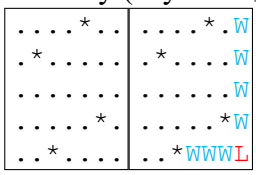
Тож, *потрібно придумати, як визначати (ви/про)грашність швидше*. Наступний пункт са́ме про це.

Оптимізаційний прийом, що прискорює визначення виграшності/програшності за рахунок розгляду в іншому порядку. Важко сказати, чи цей прийом суто алгоритмічний, чи має цінність для розуміння теорії послідовних ігор. Також важко сказати, наскільки він масовий: дуже корисний для гри «Фішки на мінному полі—2» (1.2:D, са́ме на ній ми й розглядаємо прийом), може бути корисним і для багатьох інших послідовних ігор, але є й ігри, для яких він некорисний, а для деяких навіть шкідливий.

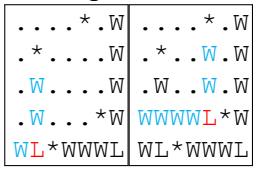
Цей прийом «вмонтовується» у спосіб «іти рядок за рядком “назустріч ходам”», а *з рекурсивним способом несумісний*.

Цей прийом ні в якому разі не змінює поняття виграшності/програшності, означені на самому початку поточного розд. 1.2; мета — отримати для всіх позицій *такі самі* значення, але швидше.

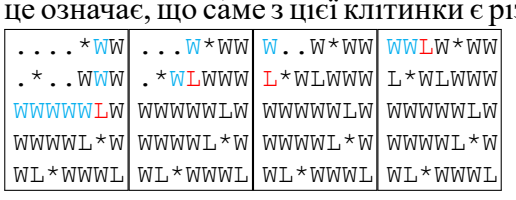
Згадані означення природні, щоб задати математичну суть, але вони спонукають спочатку вибирати, для якої клітинки-позиції ми хочемо визначити (ви/про)грашність, потім перебирати можливі ходи з вибраної позиції й шукати серед них хід у програшну позицію. *Замість цього*, можна щоразу, знайшовши програшну позицію, відразу позначати як виграшні всі позиції, звідки є хід у поточну (яку ми щойно визначили як програшну).



На лівішому з рисунків зображено поле, як воно задається у вхідних даних. На правішому — повний результат аналізу того, що права-нижня клітинка визначена програшною, а ті, звідки існує хід у неї, виграшними. Чимало клітинок вже позначені виграшними, і вже не треба запускати перебір ходів із них: вони ж не просто так позначені, а тому, що з них вже знайшли хід у програшну, й цього досить.



На цих чергових двох рисунках зображено спочатку (лівіше) повні результати аналізу ще однієї клітинки, з якої взагалі нема ходів, потім (правіше) повні результати аналізу клітинки, з якої ходи є, досі вона не була ніяк позначена, при перевірці ходів *із* неї виявилося, що вона програшна, після чого при перевірці ходів *до* неї вдалося позначити ще кілька клітинок виграшними. (Одну з них позначили виграшною повторно; це означає, що са́ме з цієї клітинки є різні «виграшні ходи», що ведуть до різних програшних позицій).



А на цих рисунках зображено ще 4 результати аналізу клітинок, кожна з яких досі не була позначена й виявилася програшною. На цьому все: (ви/про)-грашності всіх $5 \times 7 - 4 = 31$ незамінованих клітинок були визначені за 7 аналізів-у-обидва боки, кожен з яких не надто складніший за аналіз у один бік, що з’ясовує стан лише однієї клітинки-позиції.

Дії на кожній ітерації головн(ого/их) цикл(у/ів) змінюються, але ***сам(і) головн(ий/і) цикл(и) лиша(є/ю)ться без змін*** відносно ранішого способу. Зокрема, погляньмо ще раз на щойно завершену

серію рисунків-табличок, зважаючи лише на червоні позначки: вони, як і раніше, проходять рядок за рядком справа наліво, знизу догори.

Бачимо, що в *усіх* випадках досі не позначена клітинка виявлялася програшною. Тому виникає питання: це збіг обставин, чи вони *гарантовано* програшні й можна було навіть не перевіряти?

Переформулюймо питання: чи можливо, щоб досі не позначена клітинка виявлялася не програшною, а виграшною? Згідно початкового означення, позиція виграшна, коли «або факт її отримання вже означає виграш, або існує хоча б один хід з неї у програшну». У грі, яку зáраз розглядаємо (задачі 1.2:C, 1.2:D, 1.2:E), перший випадок (виграти самúм лише отриманням позиції, без подальших ходів) неможливий,[2] і лишається розглянути тільки випадок «існує хоча б один хід у програшну».

Як організован(ий/і) головн(ий/і) цикл(и) перебору позицій? (Т(ой/і) сáм(ий/і), про як(ий/і) вже казали, що без змін відносно ранішого способу.) Так, щоб розглянути дальші від старту позиції раніше, щоб щоразу, коли виникає потреба з'ясувати, в які позиції ведуть ходи з поточної позиції, для тих дальших позицій вже було відомо, виграшні вони чи програшні. Якщо проблеми вже з цим — цей оптимізаційний прийом нема до чóго застосовувати, бо нема тієї менш оптимальної версії алгоритму, в яку його слід «вмонтувати».

Але це означає, що намір «щоразу, знайшовши програшну позицію, відразу позначати як виграшні всі ті позиції, звідки є хід у цю програшну» *не поєднується* з «із теперішньої (для головн(ого/их) цикл(у/ів)) клітинки (позначимо її A) є хід у деяку програшну клітинку-позицію (позначимо її B)»: враховуючи попередній абзац, існування такої B означало б, що B вже була розглянута раніше, причому протягом розгляду B всі клітинки, звідки є ходи в B, були позначені виграшними, а до цих «всіх клітинок» належить зокрема й A. Тобто, припущення, що досі не позначена клітинка-позиція виявляється виграшною, утворює протиріччя (клітинка не позначена, хоча повинна бути позначеною); тобто, це неможливо. А припущення, що досі не відвідана клітинка-позиція виявляється програшною, має не лише підтвердження прикладами, а й теоретично теж усе якраз виходить: з програшної можна піти лише у виграшні, й ми ніяк не позначали ті, звідки можна прийти у виграшні.

Наскільки великий виграш швидкодії дає цей прийом? На жаль, це *сильно* залежить від правил конкретної гри, дати універсальну відповідь просто неможливо. Конкретно щодо 1.2:D «Фішки на мінному полі—2» вище було сказано, що без цього прийому треба розглядати $\approx {}^1\!/_2 \times MN \times (M+N)$ (за припущення $M \approx N$ це по суті $\approx N^3$) ходів. А з цим прийомом виходить $\Theta(MN)$ дій,[3] що багато кому може здаватися контрінтуїтивним, бо просто йти через усі клітинки, пропускаючи міни та вже позначені виграшними, вже потребýє $\Theta(MN)$, й інтуїція може (помилково!) схилити до думки, ніби якщо позначення клітинок виграшними відбувається у ще вкладеніших циклах, то й асимптотична оцінка мусить бути більшою; однак, це не так: кожна ітерація кожного з цих найвкладеніших циклів позначає якусь клітинку-позицію як виграшну, а клітинка-позиція може бути позначена виграшною не більше разу по вертикалі й не більше разу по горизонталі, тож, незважаючи на велику вкладеність, сумарний час роботи цього етапу теж складає $\Theta(MN)$, звідки й увесь час аналізу всього пóля складає $\Theta(MN)$.

*А все-таки **чому** цей прийом дає виграш у швидкодії?* Головним чином тому, що для оголошення позиції виграшною досить знайти хоча б одну програшну, куди можна походити з неї, і при очевидній реалізації перевірки (ви/про)грашності для натрапляння на такий хід у програшну зазвичай доводиться перебирати чимало некорисних ходів у виграшні. А цей прийом дозволяє концентрувати розгляд корисних ходів.

А як це може поєднуватися з тим, що для деяких ігор цей прийом взагалі нічого не дає, а іноді навіть шкідливий? Головних причин дві. (1) Для цього прийому важливо «розвертати» ходи, тобто шукати не «наступні» позиції (куди можна піти з поточної), а «попередні» (звідки можна прийти у поточну). Для деяких ігор це важко/незручно/… (2) Хоч у попередньому абзаці й пояснено, як

[2] І не лише сáме в цій грі, а й у всіх, що мають умову нормального завершення у смислі розд. 1.1.7, й можна переконатися, що подальші міркування правильні для всіх таких ігор. Однак, якщо гра (як, наприклад, у задачі 1.2:F), *не* задовольняє цю умову, то ситуація дещо інша; її аналіз відкладемо до розд. 1.2.3.

[3] тут і далі, Θ, O та Ω — асимптотичні позначення, що вивчалися в дисципліні «Алгоритми та структури даних»

цей прийом концентрує корисні ходи у програшні позиції й відкидає некорисні ходи у виграшні, є й протилежна сторона. Діючи без цього прийому суто за означенням, можна обривати цикл(и) перебору наступних позицій після того, як знайдено хоча б одну програшну наступну позицію й поточну позицію вже можна оголосити виграшною. А коли діємо згідно з цим прийомом, щоб точно не пропустити нічого важливого, слід перебирати *всі* ходи́, що ведуть у *кожну* програшну позицію, й тут нема (принаймні, поки говоримо про взагалі будь-яку гру, а не якусь конкретну) гарантованої причини обривати такі перебори. (Зокрема, на цих рисунках-табличках можна бачити ситуації, коли *потрібно* продовжити позначування клітинок-позицій виграшними навіть після того, як деякі клітинки-позиції позначають виграшними *повторно*.) І в принципі можна придумати таку гру, щоб таких повторів було багато.

Задача 1.2:E. «Фішка на мінному полі — інтерактив»

Правила гри відповідають задачам 1.2:C «Фішка на мінному полі—1» та 1.2:D «Фішка…—2». Кожен з розмірів по́ля в межах від 1 до 123.

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Протокол взаємодії. На початку, один раз, Ваша програма повинна прочитати поле гри (у форматі, відповідному задачам 1.2:C та 1.2:D).

Потім вона повинна повторювати такий цикл:

1. Якщо йти нема куди, вивести фразу “`I lost...`” (без лапок, символ-у-символ згідно зразку) й завершити роботу.
2. Вивести в окремому рядку свій хід, тобто спочатку або велику латинську `D` (якщо хід униз), або велику латинську `R` (якщо направо), потім пробіл, потім єдине ціле число — на скільки клітинок переміститися. Ця кількість повинна бути цілою, строго додатною, не виводити за ме́жі по́ля, не приводити у клітинку з міною і не перестрибувати ні через одну клітинку з міною.
3. Якщо після цього програмі-суперниці нема куди йти, вивести фразу “`I won!`” (без лапок, символ-у-символ згідно зразку) й завершити роботу.
4. Прочитати хід програми-суперниці, у форматі в точності як у позаминулому пункті. Гарантовано, що цей хід відповідає всім вимогам, сформульованим у позаминулому пункті. Само собою, ця гарантія дійсна лише якщо Ваша програма правильно визначила, що гра ще не закінчилася.

Все це слід повторювати, доки фішка не потрапить у клітинку, з якої за правилами не можна вийти (тобто, доки якась із програм-гравців не програ́є). Програма-суперниця не виводить фраз “`I won!`” / “`I lost...`” чи якихось їх аналогів.

Приклади:

вхід/суперник	Ваша програма
2 4 .... .**. R 1	 R 2 D 1 I won!

вхід/суперник	Ваша програма
2 4 .... .**. 	 D 1 I won!

Примітки. Нібито порожні рядки між різними ходами у прикладі зроблені, щоб краще було видно, хто коли ходить; вводити/виводити їх не слід.

Обидві наведені послідовності ходів є прикладами правильної гри. Ваша програма не зобов’язана при різних запусках для одного поля робити різні ходи. Але вона має таке право. При цьому автоматична перевірка не шукатиме кращий чи гірший результат, а просто оцінюватиме перший.

Оцінювання. У приблизно половині тестів Ваша програма матиме справу з ідеальною програмою-суперницею, яка не робить помилок. У іншій приблизно половині — з різними програмами-суперницями, які грати не вміють — тобто, роблять лише ходи, які дотримуються формальних вимог гри, але можуть вибирати не найкращий з допустимих ходів, дотримуючись кожна

своїх уявлень про те, як варто грати в цю гру. Буде оцінюватися як уміння Вашої програми виграти там, де це гарантовано можливо, так і вміння Вашої програми гідно, дотримуючись правил гри, програти, так і вміння Вашої програми скористатися (теж згідно правил) помилками чи іншими неадекватностями програми-суперниці, якщо такі будуть.

За будь-яке порушення правил гри з боку Вашої програми, відповідний тест оцінюватиметься як не пройдений.

Розбір задачі. Тут цілком достатньо поєднати ідеї розв'язань задач 1.2:C («Фішка…—1») та 1.2:B («Баше–інтерактивна»). Навіть не обов'язково використовувати оптимізаційний прийом, як у задачі 1.2:D («Фішка…—2»), бо розміри по́ля все-таки не настільки великі. (Хоча, дуже вже неоптимальні розв'язки можуть і не пройти за часом.)

Тут *не* потрібні випадкові числа. Точніше кажучи, на час написання цього тексту мені не вдалося придумати, які в цій грі можуть бути тупі програми-суперниці, щоб і «не вміли грати», і для уникнення несвідомого підігрування їм корисні були б випадкові числа; якщо вдасться, такі тести будуть додані, тому про всяк випадок можна все-таки зробити випадковий вибір ходу (звісно, лише з програшних позицій і лише якщо є більше одного варіанту).

Чи варто, виконуючи не-перші ходи, заново запускати увесь пошук виграшних та програшних клітинок-позицій? Загальний життєвий досвід може схилити до думки, ніби варто, бо «а може, завдяки виконаним ходам з'явилась якась нова інформація, яку варто врахувати?» та/або «а може, вже знаючи, що потрапили са́ме в цю позицію, слід якогось ретельніше розглянути варіанти, пов'язані са́ме з нею?». Але це той випадок, коли життєвий досвід і математична модель не відповідають одне́ о́дному. Ця гра детермінована з повною інформацією, а призначення клітинкам-позиціям позначок "`W`"/"`L`" відбувається з точки зору ідеального гравця, враховуючи всі варіанти розвитку гри; тож, ні нової інформації, ні ще ретельнішого розгляду не може бути; значить, і заново щось розглядати недоцільно.

(Попередній абзац цілком правильний, доки позначки "`W`"/"`L`" розставляються цикл(ами/ом), що перебира(ють/є) «головну поточну позицію» «назустріч ходам». При використанні ж рекурсії із запам'ятовуваннями стає можливим випадок, коли прийшли у позицію, яку досі не розглядали. (Як це стає можливим? Наприклад, так: аналізуючи деяку позицію, ми швидко знайшли хід з неї у програшну, тим з'ясували, що позиція виграшна, й не стали розглядати інші ходи з неї; але протягом гри ця виграшна позиція дісталася суперни́ку, й він походив з неї способом, який ми не розглядали.) Однак, і для випадку рекурсії із запам'ятовуваннями теж вигідно продовжувати користуватися вже знайденими розв'язаними позиціями (навіть якщо нема гарантії, що завжди потраплятимемо в уже розв'язані позиції, все'дно лишається хоча б велика ймовірність цього), а вичищати все запам'ятоване й починати все геть заново абсолютно недоцільно.)

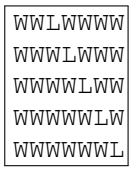
Повернімось до способу, де позначки "`W`"/"`L`" розставляють у циклах, і поміркуймо, як зручно реалізувати підхід «розставити один раз, користуватися на всіх ходах». Очевидними є такі два варіанти.

(а) Будувати й зберігати лише позначки "`W`"/"`L`" (мабуть, у тій же табличці, де й "`*`", але це деталі реалізації, які можна зробити по-різному); тоді на кожному ході треба передивлятися як поточну позицію, так і позиції, куди є ходи з неї; зокрема, якщо поточна клітинка-позиція виграшна, треба (знову) перебрати ходи з неї, щоб знайти «виграшний хід», який веде до програшної.

(б) Щоразу, позначаючи клітинку-позицію як "`W`", тут же запам'ятовувати (у ще одній табличці, наприклад, типу `(char,short)[,]`), який хід ("`D`" чи "`R`", і на скільки клітинок) дав право позначити цю позицію як "`W`". Відповідно, під час «само́ї гри» (виконання ходів) це можна буде бра́ти в готовому вигляді. Треба тільки визначитися, що зберігати в тих комірках, якщо «виграшних ходів» більше одного, і що, якщо їх нема (позиція програшна, чи клітинка не позиція, а міна).

(У кожного з цих варіантів свої переваги й свої недоліки; на мою думку, варіант (б) зручніший, *якщо* для нього достатньо пам'яті. Також варто розуміти, що варіант (б) *не* є чимось унікальним; це, радше, пристосування під цю задачу загальної ідеї так званого ***масиву виборів*** зі зворотного ходу для динамічного програмування, пошуку в ширину й багатьох інших алгоритмів. Ця ідея описана в дуже багатьох джерелах, зокрема [16, с. 54–55] (тільки там ***масив попередників***, де зберігається, з якої платформи вигідно прибути, а тут са́ме масив виборів, бо зберігаємо не «клітинку, куди прийдемо», а «куди й на скільки клітинок хід»; перетворити одне в інше геть не складно, але це різні речі, й слід розуміти, що зберігаємо).)

У що перетворюється «Фішка …» при відсутності мін? У *дуже просту* задачу.



На рисунку зображено приклад поля без мін, де вже визначили виграшні та програшні позиції. Очевидно, що без мін *завжди* (за будь-яких розмірів поля) буде аналогічна картина: програшна права-нижня клітинка-позиція — єдина програшна на весь нижній рядок і весь правий стовпчик; клітинка-позиція, сусідня з правою-нижньою по діагоналі ліворуч-угору, програшна (з неї є ходи лише у виграшні); далі міркування постійно повторюються як щодо відповідних рядка і стовпчика, так і щодо «діагоналі» ліворуч-нагору від правої-нижньої клітинки-позиції: ця «діагональ» складається з програшних позицій, вся решта позицій виграшні.

1.2.3 Що зміниться, якщо умова завершення гри не є «нормальною»?

Йдеться про «нормальну» згідно розд. 1.1.7: хто зробив останній хід — виграє́, хто не може походити — програє́. Відповідно, за́раз розглянемо чи то найпростішу зміну «на протилежне», тобто «хто зробив останній хід — *про*програє́, хто не може походити — *ви*грає́», чи то ще якийсь варіант.

(Напридумувати таких варіантів можна багато; наприклад, взявши за основу «Фішку на мінному полі», можна змінити правила, наприклад, на «якщо гра закінчилася на такій тупиковій клітинці, що і праворуч, і знизу однаково або край по́ля, або міна, то хто зробив останній хід — виграє́, хто не може походити — програє́; однак, якщо з одного з цих боків (праворуч/унизу) край по́ля, а з іншого міна, тоді, навпаки, хто зробив останній хід — програє́, хто не може походити — виграє́»; або, наприклад: на «розфарбуємо всі клітинки мінного по́ля в чорний і білий кольори за принципом шахівниці, причому лівий верхній кут білий; якщо гра закінчилася на білій тупиковій клітинці, то хто зробив останній хід — виграє́, хто не може походити — програє́; однак, якщо гра закінчується на чорній тупиковій клітинці, тоді, навпаки, хто зробив останній хід — програє́, хто не може походити — виграє́»; і так далі. Звісно, ці варіанти якісь заплутані… але в тім-то й справа, що правила визначення переможця можуть бути різними, а техніка виграшних і програшних позицій все’дно може з цим упоратися. Звісно, на кожен такий варіант треба по-своєму переписати шматочок коду… але в тім-то й справа, що шматочок, а решта коду при цьому незмінна.)

Вважаємо, що правила визначення (ви/про)грашності позицій, з яких нема ходів, можуть бути якими завгодно, але якщо ходи є (хоча б один) — діють давно відомі «позиція програшна, якщо всі ходи ведуть у виграшні» та «позиція виграшна, якщо хоча б один хід веде у програшну».

(Нема смислу розглядати ігри, де якісь позиції водночас і (ви/про)грашні само́ю фактом (а не подальшими ходами), і з цих же позицій є подальші ходи. Якщо така позиція виграшна, то навіщо ще кудись ходити, коли можна вже за́раз оголосити себе переможцем гри? А якщо програшна, то чи це значить, що вже програв і гра скінчилася? (А що тоді дають подальші ходи?) Чи що можна продовжувати гру? (А в чому тоді полягає програшність-за-фактом такої позиції?)

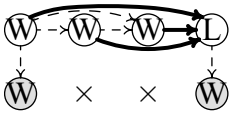
Все це стає трохи осмисленішим, якщо вважати «на цьому кроці програв, але потім можливо відіграється»; таку гру можливо вдасться (а можливо й не вдасться) проаналізувати засобами якогось іншого розділу (можливо, 1.7), але це точно не підходить під вимоги поточного розд. 1.2, особливо в частині «всі ігри будуть… закінчуватися виграшем одного з гравців і програшем іншого (без можливості нічиїх і без числового вираження виграшу)».)

Згідно розглянутих у вступі аксіом теорії ігор, гравці знають правила гри, і прикладають усіх (дозволених правилами гри) зусиль для свого виграшу. Тому́, ***якщо правило визначення переможця змінилося на протилежне — вважаємо, що обидва гравці знають про це, і прикладають зусиль для свого виграшу вже за зміненими правилами***.

(Життя складне́, й поза всяким сумнівом траплялося, що хтось колись не домовився із суперником щодо правил, і один гравець прикладав зусиль, щоб виграти в одному смислі, а інший — у іншому… Але такі ситуації зазвичай не є предметом наукового аналізу, за деяких обставин можуть бути причиною звинувачення в шахрайстві, в окремих випадках можуть бути предметом вивчення психіатрії, тощо.

Предметом дослідження теорією ігор такі ситуації можуть стати хіба що коли гравці мають різні уявлення про виграш — так, що це порушує згадану на початку розд. 1.2 вимогу «закінчуватися виграшем одного з гравців і програшем іншого» — і *знають* про це. Але, звісно, тоді доводиться категорично відмовитися від засобів розд. 1.2, і в найкращому випадку знайти спосіб, як обоє досягнуть кожен своєї мети, в найгіршому — потонути у проблемах розд. 1.7.3 та/або 1.8 і нічого не з’ясувати.)

Тому, *докорінно неправильно* думати, ніби «Якщо правила визначення, хто виграв, змінити на рівно протилежні (“хто не має ходів, виграє́” замість “хто не має ходів, програє́”), то всі програшні позиції стануть виграшними, а всі виграшні програшними». Так може траплятися в деяких окремих випадках, але зазвичай буде інакше. Знаючи зміни правил гри, гравець може вибрати інший варіант ходу з тієї ж позиції, що і перешкодить тому, щоб «виграшні стали програшними й навпаки».



Наприклад, розглянемо варіант «Фішки на мінному полі», де гравець, який не може походити, виграє, на такому ж, як тут (міни розставлені так само) полі. Клітинки нижнього рядка, з яких нема ходів, стають виграшними. Верхня-права клітинка стає програшною, бо єдиний хід з неї тепер веде до виграшної, а не програшної, позиції, яка дістанеться супернику. Решта позицій верхнього рядка виграшні, бо з кожної існує хід у програшну верхню-крайню-праву.

Зокрема, початкова (ліва-верхня) позиція як була виграшною за першого формулювання правил, так і лишилася виграшною тепер; однак, перелік ходів, які ведуть до програшних позицій і тим забезпечують виграш, змінився докорінно.

Коли розв'язуємо задачу, спираючись на поняття виграшних та програшних позицій — конкретні відповіді зазвичай змінюються, але в алгоритмі змінюються, по суті, *лише* початкові значення виграшності/програшності тупикових позицій, а далі процес відбувається за цілком тими ж правилами, якщо говорити про «просте» використання означень.

Якщо ж говорити про оптимізаційний прийом, змін трохи більше, але теж небагато. Коли головний цикл перебору позицій натрапляє на ще не позначену позицію, тепер не можна казати, що вона точно програшна: варіант, що вона згідно правил поточної гри (відмінних від «нормальних умов завершення») виграшна, тепер цілком можливий. Відповідно, для ще не позначених позицій тепер *потрібна* перевірка виграшності/програшності. Але якщо позиція *не* означає *негайного* завершення гри виграшем того, кому вона дісталася, й виграшність позиції може досягатися лише за властивістю «існує хоча б один хід у програшну позицію», то *в цьому випадку* цілком працюють раніші міркування, які доводили, що виграшність ще не позначених позицій призводить до протиріччя, отже неможлива.

Повторимо змінений оптимізаційний прийом іншими словами, виділивши цианови м кольором спільне з раніше описаним варіантом для ігор з нормальною умовою завершення, а жовтогарячим відмінне.

Перебираємо позиції основним(и) цикл(ом/ами) «назустріч ходам».

- Натрапивши цим(и) основним(и) цикл(ом/ами) перебору на вже оцінену (обов'язково виграшну!) позицію, просто переходимо далі, нічого не роблячи.
- Натрапивши цим(и) основним(и) цикл(ом/ами) перебору ще не розглянуту позицію, треба перш за все перевірити, чи існують з неї якісь ходи:
  - якщо ходів з неї нема, то (ви/про)грашність слід визначати за специфічними для цієї задачі й цієї позиції правилами;
  - якщо ходи є (хоча б один), це програшна позиція, бо виграшна-з-причини-існування-ходу-до-виграшної вже була би позначеною.
- Для всіх програшних позицій (байдуже, за яким з цих пунктів була визначена програшність) слід запускати цикл(и) позначання виграшними всіх позицій, звідки є ходи в цю програшну.

Задача 1.2:F. «Фішка на мінному полі — навпаки»

Єдина відмінність цієї задачі від задачі 1.2:C — той, кому нема куди ходити, виграє, а не програє. Це єдина з усіх задач серії «Фішка на мінному полі», де вжите таке (протилежне «нормальному») правило визначення виграшу.

Решта основної частини умови, формати вхідних даних (включно з обмеженням розмірів «від 1 до 12») та результатів (не самі результати, а формат) цілком відповідають задачі 1.2:C.

Приклади:

Вхідні дані	Результати
2 4 .... .**.	1
1 1 .	1

Примітка. У першому прикладі, єдиний спосіб, яким перший гравець може виграти — піти три клітинки праворуч, після чого дру́гий буде змушений піти на одну клітинку вниз (це буде єдиний можливий його хід), після чого першому не буде куди йти, і він ви́грає. Таким чином, зміна правила визначення переможця на протилежне не гарантує зміни переможця на протилежного. Водночас, дру́гий приклад показує, що зміна правила визначення переможця на протилежне в деяких випадках може змінити переможця на протилежного.

Розбір задачі. Все вже пояснено раніше.

1.2.4 Що зміниться, якщо гра упереджена?

Тим, хто уважно читав розд. 1.1 (класифікацію ігор) та початок розд. 1.2.1 «Гра Баше», це очевидно: *просто питання «чий хід?» слід включати у позицію*. Щоб позиція була, наприклад, не кількістю паличок, а парою (кіль-ть паличок; хто ходить?), або, наприклад, не клітинкою прямокутного по́ля, а трійкою (номер рядка; номер стовпчика; хто ходить?). Далі застосовуємо щодо ***таких*** позицій ту саму табличну техніку.

Наприклад, розглянемо гру, аналогічну грі Баше, теж «нормальну» («хто не може походити, програє»), але нехай 1-й гравець може бра́ти на кожному своєму ході або 1, або 7 паличок, а 2-й — або 3, або 4 палички.

Найдурніше, що можна зробити — лементувати «ой лихо, тепер же навіть не ясно, чи позиція “1 паличка” виграшна, чи програшна, бо тепер 1-й гравець може забрати її одним ходом і тим виграти, а 2-й не може». В позаминулому абзаці чітко сказано, що «1 паличка» *не є позицією*! У «звичайній» грі Баше було позицією, а тепер не є. Тепер є окремо позиція (1; 1) і окремо (2; 1) (будемо позначати так позиції «ходить 1-й гравець, лишилася 1 паличка» і «ходить 2-й гравець, лишилася 1 паличка»), і позиція (1; 1) виграшна, а (2; 1) програшна, са́ме зі згаданих на початку цього абзацу причин.

(Чому я позначив «лишилась 1 паличка, ходить 2-й гравець» за (2; 1), а не (1; 2)? Поважних причин нема; якщо хочете, можете скрізь попереставляти ці чи́сла. Однак, є дві другорядні причини: (1) якщо робити масив зубчастим, то накладні витрати на розміри $2 \times (n+1)$ значно менші, чим на $(n+1) \times 2$; (2) далі наведено кілька табличок із процесом визначення (ви/про)грашності, й низькі таблички на всю ширину колонки значно легше вверстати у текст, чим високі вузькі таблички. Втім, є й недолік: щоб перебрати позиції «назустріч ходам», доводиться зробити цикли по стовпчикам, а не по рядкам.)

Заповнимо табличку таких позицій (до 10 паличок включно) позначками (W/L) ((ви/про)грашна). Спочатку табличка «порожня» (всі значення поки що невідомі):

	0	1	2	3	4	5	6	7	8	9	10
1	?	?	?	?	?	?	?	?	?	?	?
2	?	?	?	?	?	?	?	?	?	?	?

Використаємо (це не обов’язково, але ми використаємо) оптимізаційний прийом «для кожної програшної позиції позначати виграшними всі позиції, з яких є ходи до цієї програшної», включно з властивістю «всі досі не позначені позиції програшні».

(Якщо хочете, можете проконтролювати отримані значення (ви/про)грашності, знайшовши їх іншим способом. Але реальної потреби в тому нема. І сам оптимізаційний прийом, і доведення додаткової властивості спиралися на ациклічність гри, «нормальну умову завершення» та припущення, що головн(ий/і) цикл(и) перебору позицій вдалося організувати «назустріч ходам». А на особливості «Фішки на мінному полі» воно не спиралося. Отже, воно стосується також і цієї задачі.)

Позиція (1; 0) досі не позначена, отже програшна, тому позначимо її програшною, а позиції (2; 3) та (2; 4), ***з*** яких існують ходи ***у*** позицію (1; 0) — виграшними.

	0	1	2	3	4	5	6	7	8	9	10
1	L	?	?	?	?	?	?	?	?	?	?
2	?	?	?	W	W	?	?	?	?	?	?

(Хід однієї зі сторін включає в себе зміну складової «чий хід?». Через це, будь-який правильний спосіб аналізу такої гри (хоч із використанням оптимізаційного прийому, хоч без) постійно визначатиме (ви/про)грашність позицій кожного гравця через (ви/про)грашність позицій іншого гравця. Щоб усе це можна було робити «назустріч ходам» і постійно спиратися лише на вже розв'язані позиції, неминуче треба зробити зовнішній цикл за стовпчиками (кількостями паличок), внутрішній за рядками (гравцями).

Водночас, оптимізаційний прийом перебирає ходи ***в*** поточну позицію — отже, постійно використовує ходи (в цій грі, кількості паличок, які можна забрати за один раз) *не* поточного гравця, а іншого. Бо, щоб настала ситуація «зараз ходитиме 1-й гравець», безпосередньо перед тим має походити 2-й, і навпаки.

А аналогічний алгоритм без оптимізаційного прийому перебирає ходи ***із*** поточної позиції — отже, постійно використовує ходи само́го поточного гравця.)

Далі, з аналогічних причин, досі не позначену позицію (2; 0) слід позначити програшною, а позиції (1; 1) та (1; 7), звідки 1-й може прийти у (2; 0) одним своїм ходом — виграшними:

	0	1	2	3	4	5	6	7	8	9	10
1	L	W	?	?	?	?	?	W	?	?	?
2	L	?	?	W	W	?	?	?	?	?	?

Позиція (1; 1) вже позначена як виграшна, пропускаємо. Далі, з програшною (2; 1) і виграшними (1; 2), (1; 8) все аналогічно (2; 0) і (1; 1), (1; 7):

	0	1	2	3	4	5	6	7	8	9	10
1	L	W	W	?	?	?	?	W	W	?	?
2	L	L	?	W	W	?	?	?	?	?	?

Далі, позиція (1; 2) вже позначена як виграшна, пропускаємо, а з програшною (2; 2) і виграшними (1; 3), (1; 9) все аналогічно (2; 0) і (1; 1), (1; 7):

	0	1	2	3	4	5	6	7	8	9	10
1	L	W	W	W	?	?	?	W	W	W	?
2	L	L	L	W	W	?	?	?	?	?	?

Далі, позиції (1; 3) та (2; 3) вже позначені як виграшні, пропускаємо, а з програшною (1; 4) і виграшними (2; 7), (2; 8) все аналогічно (1; 0) і (2; 3), (2; 4):

	0	1	2	3	4	5	6	7	8	9	10
1	L	W	W	W	L	?	?	W	W	W	?
2	L	L	L	W	W	?	?	W	W	?	?

Далі, позиція (2; 4) вже позначена як виграшна, пропускаємо, а з програшною (1; 5) і виграшними (2; 8), (2; 9) все аналогічно (1; 0) і (2; 3), (2; 4):

	0	1	2	3	4	5	6	7	8	9	10
1	L	W	W	W	L	L	?	W	W	W	?
2	L	L	L	W	W	?	?	W	W	W	?

(при цьому, позиція (2; 8) позначена виграшною вдруге, але це неважливо).

Далі, позиція (2; 5) досі не позначена, отже її слід позначити програшною, а (1; 6) та (1; 12) виграшними; однак, ми будуємо табличку лише до 10, тому (й лише тому) позицію (1; 12) ігноруємо:

	0	1	2	3	4	5	6	7	8	9	10
1	L	W	W	W	L	L	W	W	W	W	?
2	L	L	L	W	W	L	?	W	W	W	?

Дальші позначення позицій — (2; 6) програшною, (1; 7) виграшною (повторно), (1; 10) та (2; 10) програшними — аналогічні ранішим крокам, поєднаємо їх:

	0	1	2	3	4	5	6	7	8	9	10
1	L	W	W	W	L	L	W	W	W	W	L
2	L	L	L	W	W	L	L	W	W	W	L

(Проговорю ще пару питань, хоч глобально й неважливих, але природніх при прочитанні поточного розд. 1.2.4.

(1) «А чому такі дивні конкретні чи́сла, як "1-й гравець — або 3, або 4 палички, а 2-й — або 1, або 7"»?

Відповідаю. Дещо несподівано, але якщо взяти довільні різні набори ходів випадково, то з великою ймовірністю два рядки з позначками (ви/про)грашності скоро виродяться: в одного з гравців усі позиції, починаючи з деякого $n^\star$ і до нескінче́нності, виграшні, а в іншого програшні. Хто хоче на це подивитися — може скопіювати цю гуглотабличку, позмінювати значення у комірках G8–G9 та G18–G19 аркушу «по 2 варіанти» чи в комірках G8–G12 та G18–G22 аркушу «по 5 варіантів» (вони відповідають за варіанти, скільки паличок може брати 1-й гравець і скільки 2-й) і дивитися результати в рядках 2–3.

Не те щоб мені було шкода гравця, якому дістануться програшні позиції… Мені шкода тих студентів, які могли б спробувати написати відповідну програму, побачити такі результати, подумати, ніби це неправда, й шукати помилку, якої нема.

(2) «А в цій модифікації гри Баше хіба не можна, як і в оригінальній грі Баше, знайти якийсь цикл — не у смислі циклу мови програмування, а в смислі “повторюваного шаблону”? Аналогічно тому, як в оригінальній грі Баше до нескінче́нності повторюються LWWW…LWWW…, так що визначення (ви/про)грашності позиції зводиться взагалі до $n\%4$…»?

Коротка відповідь — можна, але з цього значно менше толку, чим у випадку класичної гри Баше.

Наприклад, візьмемо конкретно вищезгадані ходи «1-й гравець може забирати або 3, або 4 палички, 2-й — або 1, або 7» і продовжити покинутий у стовпчику 10 аналіз, або зробити його заново іншими засобами (хоч би й щойно згаданою гуглотабличкою), можна побачити, що са́ме для цих чисел $\boldsymbol{\epsilon}$ цикл (повторюваний шаблон), довжиною 10. Але як оце 10 вивести із «1-й або 3, або 4, а 2-й або 1, або 7»? Чи, наприклад, як вивести довжину циклу 15 із правил, в цілому аналогічних, але «1-й може забирати або 1, або 3, або 5, або 6, або 7, а 2-й або 1, або 2, або 8, або 12»? Більш того: навіть якщо якось вдалося взнати довжину повторюваного шаблону, все’дно неясно, які з позицій цього шаблону виграшні та які програшні.

Тобто, практична користь від пошуку таких циклів-шаблонів обмежена, через що не будемо розглядати їх справді детально. Але коротко скажу. Бо це теоретична можливість розв’язувати нескінче́нні упереджені модифікації гри Баше (причому, не лише вигляду «1-й може забирати або 3, або 4, а 2-й або 1, або 7», а й для інших наборів таких ходів, аби вони були скінче́нними й чітко заданими сукупностями конкретних чисел). Або, що ближче до програмістської практики, не буквально нескінче́нні, але для дуже великої початкової кількості паличок, як-то 10^{17}–10^{18}: все-таки знаходимо повторюваний шаблон і за одну операцію `mod` (`%`) переходимо до того, що задачу досить розв’язати для значно меншої кількості паличок, а для величезної відповідь така сама, бо повторюється однаковий шаблон. Однак, щоб знайти довжину цього шаблона, мені відомий лише один спосіб: будувати табличку від 0 до скількох виявиться потрібно, «по дорозі» шукаючи, коли вже настане ситуація, що є однаковісінькі підряд результати позначок (ви/про)грашності довжиною $\geqslant$ максимуму з кількостей паличок, які можна забирати за один хід. Коли таке вдасться знайти — ото і є цикл, і цілі кількості його повторів можна «вирізати». Але нема ніякої гарантії, чи такий цикл буде починатися від самого початку (0 паличок), чи колись далі. Наприклад, в уже згаданій гуглотабличці на аркуші «по 5 варіантів» можна побачити, що для ходів «1-й може забирати або 1, або 3, або 5, або 6, або 7, а 2-й або 1, або 2, або 8, або 12» є цикл, довжини 15 (результати для 16 паличок такі ж, як для 1 палички, для 17 такі ж, як для 2, для 18 такі ж, як для 3, і так до нескінче́нності), але при цьому результати для 15 паличок не такі, як для 0).)

1.2.5 Ще один приклад неочевидних виграшних/програшних позицій

Задача 1.2:G. «Палички з ходами, залежними від попереднього; хто переможе?»

Є одна купка, яка спочатку містить N паличок. Двоє грають у таку гру. Спочатку перший може забрати з купки або 1, або 2 палички. На кожному подальшому ході кожен з гравців може забрати будь-яку кількість паличок від 1 до подвоєної кількості, щойно забраної суперником (обидві межі включно). Інших варіантів ходу нема. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом з деякими іншими).

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто з гравців може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Єдине ціле число N — початкова кількість паличок у купці ($2 \leqslant N \leqslant 1234567$).

Результати. Єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш, як би не грав дру́гий), або `2` (якщо дру́гий, як би не грав перший).

Приклади:

Вхідні дані	Результати
2	1
3	2
4	1
5	2
1024	2

Примітка. Ці приклади частково пояснені також у прикладах до наступної задачі.

Оцінювання. 1-й блок (тести 1–5) містить тести з умови, перевіряється завжди, але безпосередньо не оцінюється. 2-й блок (тести 6–10) містить усі значення з діапазону $6 \leqslant N \leqslant 10$ і оцінюється у 30% балів; перевіряється й оцінюється завжди. 3-й блок (тести 11–20) має обмеження $11 \leqslant N \leqslant 100$ й оцінюється у 20% балів; перевіряється й оцінюється завжди. 4-й блок (тести 21–30) має обмеження $101 \leqslant N \leqslant 1234$ й оцінюється у 20% балів; перевіряється й оцінюється завжди. 5-й блок (тести 31–40) має обмеження $12345 \leqslant N \leqslant 43210$ і оцінюється у 15% балів; перевіряється й оцінюється, лише якщо успішно пройдено всі попередні блоки. 6-й блок (тести 41–50) має обмеження $123456 \leqslant N \leqslant 1234567$ і оцінюється у 15% балів; перевіряється й оцінюється, лише якщо успішно пройдено всі попередні блоки. Для кожного окремо взятого блоку, бали нараховуються, лише якщо успішно пройдено *всі* тести блоку.

Задача 1.2:H. «Палички з ходами, залежними від попереднього; інтерактив»

Гра та сама, що у попередній задачі 1.2:G.

Напишіть програму, яка інтерактивно гратиме за першого гравця.

На початку, один раз, Ваша програма повинна прочитати одне ціле число в окремому рядку — початкову кількість паличок N ($2 \leqslant N \leqslant 12345$).

Потім слід повторювати такий цикл:

1. Вивести єдине число в окремому рядку — свій хід, тобто кількість паличок, які вона зараз забирає з купки. На першому ході це повинно бути 1 або 2, на подальших — ціле число від 1 до подвоєної кількості, щойно забраної програмою-суперницею, причому не більше за поточну кількість паличок у купці.
2. Якщо після цього купка стає порожньою, вивести окремим рядком фразу “`I won!`” (без лапок, символ-у-символ згідно зразку) і завершити.
3. Прочитати хід програми-суперниці, тобто кількість паличок, які вона зараз забирає з купки (єдине число, в окремому рядку). Якщо Ваша програма правильно визначила, що гра не закінчилася і цей хід відбудеться, то гарантовано, що він допустимий (число є цілим від 1 до подвоєної кількості, щойно забраної Вашою програмою, і не перевищує поточну кількість паличок у купці).
4. Якщо після цього купка стає порожньою, вивести окремим рядком фразу “`You won...`” (без лапок, символ-у-символ згідно зразку) і завершити.

Це слід повторювати, доки якась із програм-гравців не ви́грає. Програма-суперниця не виводить фраз “`I won!`” / “`You won...`” чи якихось їх аналогів.

Оцінювання. Тести оцінюються кожен окремо (без блоків). У 1-му тесті $N = 3$, у 2-му $N = 5$, і ці тести не приносять балів. Решта тестів приносять однакові бали. У 20% тестів $2 \leqslant N \leqslant 25$ ($N \neq 3$, $N \neq 5$), програма-суперниця ідеальна (не робить помилок). Ще у 20%, $100 < N \leqslant 1234$, суперниця ідеальна. Ще у 20%, $1234 < N \leqslant 12345$, суперниця ідеальна. Ще у 20%, $100 < N \leqslant 1234$, суперниці інші. Ще у 20%, $1234 < N \leqslant 12345$, суперниці інші. Ці інші програми-суперниці (їх кілька різних) роблять ходи, де гарантовано дотримані вимоги «забирати лише від 1 палички до подвоєної щойно забраної кількості» та «забирати не більше паличок, чим є у купці», але дотримуються кожна власних уявлень, як треба грати, частенько вибираючи не найкращий з допустимих ходів.

Буде оцінюватися і вміння Вашої програми виграти там, де це можливо, і вміння Вашої програми гідно, дотримуючись правил гри, програти, де виграш неможливий, і вміння Вашої програми скористатися (теж згідно правил) помилками чи іншими неадекватностями програми-суперниці, якщо такі будуть. За будь-яке порушення правил гри з боку Вашої програми, відповідний тест буде оцінено на 0 балів.

Приклади:

Вхід, суперник	Ваша програма
3 2	 1 You won...

У купці спочатку 3 палички. Ваша програма забирає одну, лишається дві; програма-суперниця забирає обидві й виграє.

Вхід, суперник	Ваша програма
3 1	 2 You won...

Спробуємо забрати не одну, а дві з трьох паличок, тобто лишити одну; програма-суперниця забирає її і теж виграє.

Вхід, суперник	Ваша програма
5 1 2	 1 1 You won...

У купці спочатку п'ять паличок. Ваша програма забирає одну, лишається чотири; програма-суперниця забирає одну, Вашій програмі дістається три палички, й вона ніяк не може виграти з вищеописаних причин.

Вхід, суперник	Ваша програма
5 3	 2 You won...

Спробуємо забрати дві з п'яти паличок (лишається три); програма-суперниця забирає всі три (має право, бо Ваша програма щойно взяла дві) й теж виграє.

Примітки. (1) Всі наведені послідовності ходів є прикладами правильної гри. Ваша програма не зобов'язана при різних запусках для однієї початкової кількості паличок робити різні ходи. Але вона має таке право. Якщо Ваша програма при різних запусках грає по-різному, система автоматичної перевірки не шукатиме ні найкращий, ні найгірший з результатів, а просто оцінюватиме перший. (2) Вводити/виводити порожні рядки не треба; додаткові вертикальні відступи у прúкладах зроблені умовно, щоб краще було видно, хто коли ходить.

Розбір задач 1.2:G, 1.2:H. Хоч гра, якій присвячені обидві ці задачі, схожа на гру Баше (розд. 1.2.1) за формулюванням, її аналіз *значно* складніший. Зі стандартних засобів аналізу послідовних ігор беруться, по суті, лише все ті самі означення виграшної і програшної позиції, яким і присвячений увесь поточний розд. 1.2, але потім потрібно чимало оптимізацій, для яких застосовуються зразу кілька спостережень, специфічних виключно для цієї задачі.

Ця гра має «нормальну умову завершення» та неупереджена; це дозволяє уникнути додаткових складнощів з розд. 1.2.3 та 1.2.4.

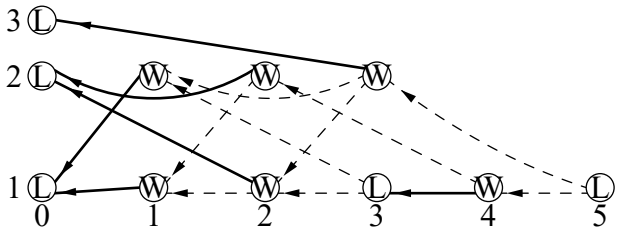
Може захотітися сказати «Раз гра неупереджена, то позицією повинна бути просто кількість паличок». Але це неправда, що можна бачити навіть за прикладами з умови задачі 1.2:H: в перших двох прикладах показано неможливість виграти, ходячи першим при $N = 3$, а в останньому прикладі суперник забирає всі три палички одним ходом і виграє́. Це суперечить згаданій тут аксіомі «Гравці вибирають ходи з зарані відомих варіантів, і ці варіанти залежать лише від поточної позиції…».

Але са́ме це й дає підказку! *Позицією цієї гри можна і варто вважати* ***пару*** **«кількість паличок, що лишилися у купці; кількість паличок, забраних на наразі останньому ході»**. Причому, хоча на

початку гри «наразі останнього ходу» не було, дозволені ходи (забирати або одну, або дві палички) якраз такі са́мі, які були б, якби щойно забрали одну паличку й лишилося N. Тож початковою є позиція $(N;1)$; якщо 1-й гравець забирає одну паличку, то 2-му дістається позиція $(N-1;1)$, а якщо дві, то позиція $(N-2;2)$. І так далі.

(А якби хтось вирішив усе зіпсувати, домігшись єдиної дрібної зміни у правилах гри «Спочатку 1-й гравець можна взяти або 1, або 2, або 3 палички» — чи зробило б це увесь подальший розв'язок недійсним? Відповідаю. *Деяку додаткову мороку* це справді принесло б: довелось би вважати, що стартова позиція не належить до «основних пар», а є «особливою», й аналізувати переходи з цієї особливої позиції у стандартні пари окремо. Звісно, змінились би також конкретні результати для деяких конкретних N. Але ніяких істотних і докорінних змін, які зробили б увесь ро́зв'язок непридатним, *не* відбулося б.)

Зобразимо граф цієї гри при $N=5$ (прямо тут) та при $N=12$ (внизу стор. 31).



Висновок «позицією слід вважати пару …» украй погано поєднується з обмеженнями задачі: якщо $N\approx 1234567$, а позицією є пара, то кількість позицій перевищує $10^{12}/4$, що не вкладеться в обмеження (ні пам'яті, ні часу). Що ж, це справді показує, що лише запрограмувати стандартний аналіз виграшності/програшності для позицій-пар не буде повним розв'язком задачі. Але програмувати все'дно варто. Накрайняк, навіть неефективну правильну програму можна здати, щоб отримати бали хоча б за деякі з блоків. Є й важливіша причина: буває, що ручний аналіз результатів, згенерованих неефективним розв'язком, дозволяє помітити особливості, на основі яких можна придумати ефективніший розв'язок. Так що дивимося на виграшні/програшні позиції, зображені внизу стор. 31.

Легко помітити, що позиції $(0;s)$ програшні при всіх s, бо $n=0$ означає, що позиція дістається після того, як суперник вже забрав останню паличку і виграв. Також легко помітити, що у випадку $n>0$ всі позиції $(n;s)$ при $s\geqslant\frac{n}{2}$ виграшні (бо при $s\geqslant\frac{n}{2}$ можна забрати всі палички за один хід)

А ще можна помітити таке: *якщо деяка позиція* $(n^\star;s^\star)$ *виграшна, то виграшні й усі позиції* $(n^\star;s)$ *при тому самому* $n^\star$ *та* $s\geqslant s^\star$ (на діаграмах і в таблиці — кожен стовпчик (крім $n=0$) або складається з самúх лише "W", або має внизу скінче́нну кількість "L", а вище самі́ лише "W").

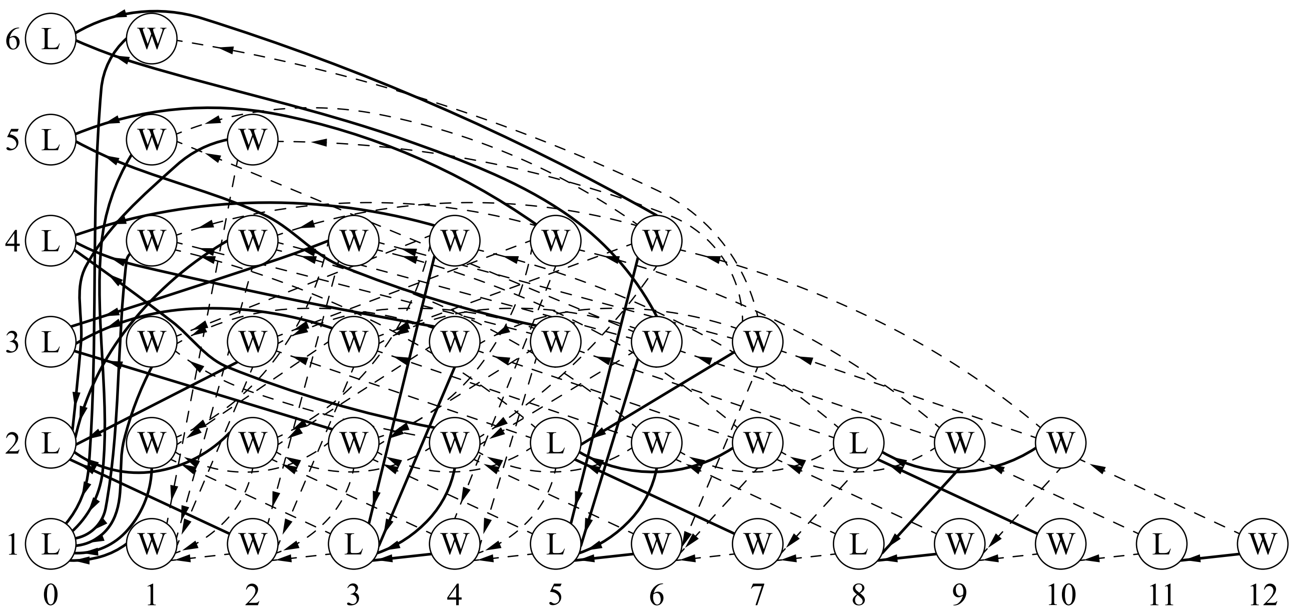
Доведемо, що це не збіг обставин для са́ме цих позицій, а так і є для всіх $n>0$. Куди можна піти з позиції $(n;1)$? Лише в $(n-1;\,1)$ (при всіх $n>0$) та в $(n-2;\,2)$ (при $n\geqslant 2$). А куди з $(n;2)$? При $n\leqslant 2$, в ті са́мі позиції; при $n\geqslant 3$ до них додається $(n-3;\,3)$, а при $n\geqslant 4$ також $(n-4;\,4)$. Порівнюючи ходи з $(n;2)$ та з $(n;3)$, знов бачимо, що всі ходи з $(n;2)$ лишаються допустимими, і до них додаються (якщо n досить велике) нові $(n-5;\,5)$ та $(n-6;\,6)$. І так для всіх збільшень s при сталому n: всі старі ходи лишаються, і до них, можливо, додаються нові. Ця гра має «нормальну умову завершення», тому позиція може бути виграшною лише за рахунок того, що з неї існує хід у деяку програшну позицію. Отже, раз цей самий хід можливий і при всіх більших s, всі позиції з більшим s за того ж n теж виграшні.

```
minWinS[0]:=round(2e9); // +∞
minWinS[1]:=1;
for i:=2 to N do
 for s:=1 to (i+1) div 2 do
  if (minWinS[i-(2*s-1)] > 2*s-1) or
     (minWinS[i-2*s] > 2*s) then
  begin // при цьому s вперше знайшли хід
   minWinS[i] := s; // у програшну позицію
   break // обриваємо внутр.-ій цикл for s…
  end // зовн.-ій цикл for i… продовжується
```

Це дає ключ до такого розв'язку. Реалізуємо стандартний аналіз «позиція виграшна, коли є хоча б один хід у програшну», але не зберігаючи кожну пару $(n;s)$, а подаючи інформацію в одновимірному масиві `minWinS` з діапазоном індексів `[0..N]` (обидві межі включно), де `minWinS[i]=k` означає,

що всі позиції $(i;1)$, $(i;2)$, …, $(i;\,k-1)$ (якщо вони взагалі існують, тобто $k>1$) програшні, а всі позиції $(i;k)$, $(i;\,k+1)$, … виграшні. По суті, останній рядок стор. 31 і є таким масивом. Остато́чну відповідь можна визначити так: якщо `minWinS[N]` $=1$, то позиція $(N;1)$ виграшна (виграє 1-й гравець), інакше програшна (2-й).

Визначити асимптотичну складність цього розв'язку важко. Дивитися на верхню межу внутрішнього циклу `for s:=1 to (i+1) div 2 do...` і робити висновок про «завелику для $N\approx10^6$ складність $O(N^2)$» нема толку, бо хоч цю межу й підібрано з осмисленого міркування «при $2\cdot s\geqslant i$ вже можна забрати всі i паличок одним ходом, тож ще більші s не потрібні», фактично цей



Далі зображено самі позначки W/L (без стрі́лок) для ще більшої кількості позицій, та вказано мінімальні s, починаючи з яких позиції стають виграшними.

s/n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
17	L	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
16	L	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
15	L	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
14	L	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
13	L	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
12	L	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
11	L	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
10	L	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
9	L	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
8	L	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
7	L	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
6	L	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W	W	W	W	W	W	L	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
5	L	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W	W	W	W	W	W	L	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
4	L	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W	W	W	W	W	W	L	W	W	W	W	W	W	W	W	W	W	W	W	L	W	W
3	L	W	W	W	W	W	W	W	L	W	W	W	W	L	W	W	W	W	W	W	W	L	W	W	W	W	W	W	W	L	W	W	W	W	L	W	W
2	L	W	W	W	W	L	W	W	L	W	W	W	W	L	W	W	W	W	L	W	W	L	W	W	W	W	L	W	W	L	W	W	W	W	L	W	W
1	L	W	W	L	W	L	W	W	L	W	W	L	W	L	W	W	L	W	L	W	W	L	W	W	L	W	L	W	W	L	W	W	L	W	L	W	W

фібоначчієвий запис числá n

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
	0	1	10	100	101	1000	1001	1010	10000	10001	10010	10100	10101	100000	100001	100010	100100	100101	101000	101001	101010	1000000	1000001	1000010	1000100	1000101	1001000	1001001	1001010	1010000	1010001	1010010	1010100	1010101	10000000	10000001	10000010

значення s, починаючи з якого всі позиції при поточному n виграшні

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
	∞	1	1	2	1	3	1	1	4	1	1	2	1	7	1	1	2	1	3	1	1	11	1	1	2	1	3	1	1	4	1	1	2	1	17	1	1

цикл обривається `break`-ом, і справжня кількість його ітерацій незрозуміла. Експериментально, складність усього алгоритму схожа на $\Theta(N \log N)$ з малим константним множником. І він проходить усі тести всіх блоків.

(При бажанні можна побудувати й ще значно ефективніший алгоритм. Продовживши аналіз масиву `minWinS`, можна помітити, що особливо великі значення досягаються на індексах, рівних числам Фібоначчі. Тож можна спробувати, що буде, якщо записувати значення N у т. зв. системі числення Фібоначчі, й отримати, що 2-й гравець може забезпечити собі виграш при тих і тільки тих початкових значеннях N, які у системі Фібоначчі закінчуються на хоча б два нулі. Як це довести і чи можна це використати в задачі 1.2:H (де значно більша потреба працювати з $s > 1$), нехай залишиться за межами цього посібника.)

Насамкінець, якщо визначити, що позиціями гри є пари $(n; s)$, але не помітити, що «якщо деяка позиція $(n^\star; s^\star)$ виграшна, то виграшні й усі позиції $(n^\star; s)$ при тому ж $n^\star$ та $s \geqslant s^\star$», може мати смисл реалізувати перевірку позицій на виграшність/програшність рекурсією із запам'ятовуваннями, зберігаючи результати вже переглянутих позицій не у двовимірному/зубчастому масиві, а в `Dictionary`, ключами якого є пари $(n; s)$. При грамотній реалізації (зокрема, оголошувати позицію виграшною, щойно знайшовши перший хід у програшну, а не крутити цикл до кінця) значна частина з $\Theta(N^2)$ позицій, теоретично досяжних з $(N; 1)$, практично не розглядаються, тож не потраплять у `Dictionary`, тож такий розв'язок *можна* написати так, що він пройде блок тестів $12345 \leqslant N \leqslant 43210$. Але пройти так останній блок, начебто, неможливо.

Що стосується інтерактивної версії 1.2:H, то я мав намір зробити так, щоб набрати бали на інтерактивній версії було *легше*, чим на основній версії 1.2:G. По-перше, нарахування балів за окремі тести (а не блоки) збільшує ймовірність отримати бали за те, що в деяких окремих тестах Ваша програма виграє випадково. (Враховуючи, що не в усіх тестах програма-суперниця грає ідеально, це можливо навіть для неправильної програми.) А є ж ще тести, в яких і слід програти…

1.3 Гра Нім

1.3.1 Правила гри Нім

Є N купок, кожна з яких містить деяку кількість паличок. Ці кількості відомі гравцям; позначимо їх $m_1, m_2, \ldots, m_k$. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з будь-якої однієї купки будь-яку кількість паличок, від 1 до зразу всіх паличок цієї купки. Палички можна лише забирати (ні додавати, ні перекладувати з купки в купку не можна). Ніяких інших варіантів ходу нема. Коли купка стає порожньою (кількість паличок=0), гра просто продовжується для решти купок. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом із ще деякими) з останньої купки. (Інакше кажучи, виграє той, після чийого ходу не лишається жодної палички в жодній купці. Ще інакше кажучи, гра має «нормальну умову завершення», тобто хто не може походити, той програє.)

Навіщо тут нові засоби? Чим не такі виграшні та програшні позиції? *Якби* ми дивилися на це суто теоретично, ігноруючи зокрема й питання обчислювальних ресурсів (часу й пам'яті) — все ті ж виграшні та програшні позиції з розд. 1.2 були б цілком адекватним засобом. Позначимо початкові кількості паличок у 1-й, 2-й, …, N-й купках як $m_1, m_2, \ldots, m_k$, а проміжні (протягом гри) кількості паличок у тих самих купках як $s_1, s_2, \ldots, s_k$. Тоді всі ті $(s_1, s_2, \ldots, s_k)$, де $\forall i\,(0 \leqslant s_i \leqslant m_i)$ — можливі позиції гри, позиція $(0, 0, \ldots, 0)$ програшна, й далі можна «розкручувати» виграшність та програшність решти позицій аналогічно розд. 1.2.

Але якщо згадати про витрати процесорного часу й пам'яті, все різко стає значно менш прийнятним. Наприклад, 5 купок з кількостями паличок 50, 100, 100, 200, 200 інтуїтивно не здаються великими для комп'ютера кількостями. Але якщо на це са́ме подивитися як на $51 \times 101 \times 101 \times 201 \times \times 201 \approx 2{,}1 \cdot 10^{10}$ — ставлення різко змінюється.

1.3.2 Виграшна стратегія гри Нім

Чесно кажучи, цю стратегію дуже важко описати так, щоб вона з'явилася природньо й очевидно. Тож не буду й пробувати пояснити, як (здогадатися про таке/додуматися до такого), просто пред'явлю дивне твердження і постараюся його хоча б довести, раз не вдається пояснити.

(Математично строго (хоч і переважно обмежуючись частковим випадком $k=3$) ця стратегія вперше була описана в [17]. Однак, те джерело теж лише дає факти, не пояснюючи, як вони були отримані.)

Логічний ксор та побітовий ксор. Для початку, це твердження буде використовувати побітовий ксор. А щоб з'ясувати/згадати, що таке побітовий ксор, варто спочатку з'ясувати/згадати, що таке логічний ксор (він же «виключне або» (слово "xor" і є скороченням від "e**X**clusive **OR**"), він же «додавання за модулем 2», він же «сума Жегалкіна»). Щоб прочитати про логічний ксор детальніше, можна звернутися до [18](link), [8, сс. 8–11, 14–15] та ще багатьох джерел. Однак, найголовніше для поточної ситуації наведу прямо тут: логічний ксор — це така бінарна логічна операція, результат якої істинний тоді й тільки тоді, коли істинний рівно один з двох аргументів. Те са́ме у ще кількох формах подання:

$\oplus$	0	1
0	0	1
1	1	0

a	b	$a\oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

$$0\oplus 0=0$$
$$0\oplus 1=1$$
$$1\oplus 0=1$$
$$1\oplus 1=0$$

Для логічного ксора виконуються комутативність $a\oplus b=b\oplus a$ та асоціативність $a\oplus(b\oplus c)=(a\oplus b)\oplus c$, а також тотожність $a\oplus 0=a$; враховуючи також уже згадану рівність $1\oplus 1=0$, можна дуже легко рахувати послідовний логічний ксор великої кількості аргументів. Нема потреби знаходити значення таких виразів «покроково» як

$$\underbrace{1\oplus 0}_{1}\oplus 1\oplus 1\oplus 0\oplus 1\oplus 1=1,$$

можна рахувати легше&швидше як «у виразі $1\oplus 0\oplus 1\oplus 1\oplus 0\oplus 1\oplus 1$ одиниць 5 шт., тобто непарна кіль-ть, тож результат=1» та «у виразі $1\oplus 0\oplus 1\oplus 0\oplus 0\oplus 1\oplus 1$ одиниць 4 шт., тобто парна кіль-ть, тож результат=0».

Побітовий ксор застосовується до *цілих невід'ємних* (це важливо!) чисел і отримується так: *двійкові* (це важливо!) подання чисел записуються у стовпчик (якби чи́сла були десятковими, це звучало б як «розряди одиниць під розрядами одиниць, розряди десятків під розрядами десятків, розряди сотень під розрядами сотень, …»; за́раз треба те са́мо, просто у двійковій системі), далі в кожному двійковому розряді окремо відбувається логічний ксор. Наприклад, знаходити $321\oplus 283\oplus\oplus 126$ (де 321, 283 та 126 — десяткові числа) слід так:

$$\begin{array}{rcl} 321\,Dec & = & 101000001\,Bin \\ 283\,Dec & = & 100011011\,Bin \\ 126\,Dec & = & 001111110\,Bin \\ \hline & = & 000100100\,Bin \quad = \quad 36\,Dec. \end{array} \tag{1}$$

(У розрядах, де нема одиниць чи є по 2 одиниці, вийшов нуль; у розрядах, де є по 1 одиниці, вийшла одиниця; розрядів, де було б по 3 одиниці, са́ме для цих чисел нема, а якби були, то там вийшли б одиниці. І взагалі, діє вже згадане правило «якщо кількість одиниць непарна, виходить 1, а якщо парна, виходить 0», просто це треба застосувати до кожного біта окремо. Зовсім окремо. Нема ніякого аналогу тих перенесень між розрядами, які бувають при додаванні чи відніманні у стовпчик.)

Хоч для ручного виконання й багато мороки переводити у двійкову систему й підраховувати в кожному біті, мовами програмування, як правило, це пишеться дуже легко, бо, як правило, є готова операція «побітовий ксор». Для C-подібних мов ця операція позначається "^" («кришка», вона ж «карет»; набирається на клавіатурі як "`Shift 6`"; ASCII-код $94_{Dec} = 5E_{Hex}$). Наприклад, мовою C# можна написати `Console.WriteLine(321^283^126)`, і буде виведено оце саме `36`, яке майже отак у стовпчик і рахується, але все це для комп'ютера просто дві стандартні операції "^".

(Згадаймо, що всере́дині комп'ютера чи́сла зберігаються у двійковому вигляді, а десятковий — лише видимість для людини. Наприклад, 321 не переводиться у 101000001 спеціально, щоб виконати "^"; воно, будучи `Int32`, завжди зберігається як 00000000 00000000 00000001 01000001.)

А навіщо всі ці ксори? Який вони мають стосунок до гри Нім? Виявляється, найпрямійший. ***Програшними позиціями гри Нім є ті й тільки ті, де ксор кількостей паличок у різних купках дорівнює 0. Відповідно, позиції, де ксор кількостей паличок у різних купках ненульови́й, виграшні.*** Не питайте мене, як здогадатися, що це так. Я не знаю. Тим паче неясно, як до цього додумався Ч. Л. Бу́тон (C. L. Bouton) у [17] ще на межі XIX і XX століть, коли не лише побітовий ксор, а й двійкова система числення не була поширеною.

(До речі, Бу́тон і не вживав терміну «ксор»; у [17] ця операція взагалі ніяк не названа. Чимало джерел, присвячених теорії ігор, називають ці операції (і логічну, й числову) «нім-сумою». Але, на мою думку, за́раз доречніше вживати нині поширенішу назву «ксор».)

Якщо факт «ксор=0 $\Leftrightarrow$ позиція програшна, ксор$\neq$0 $\Leftrightarrow$ позиція виграшна» звідкись уже відомий, то довести, що це так, не дуже складно.

Позиція $(0, 0, \ldots, 0)$, якою гра закінчується, повинна бути програшною. З цим усе гаразд: які двійкові розряди не бери, скільки купок не бери, завжди виходить $0 \oplus 0 \oplus \ldots \oplus 0 = 0$.

Про всяк випадок наголосимо, що побітовий ксор може бути 0 не лише на позиції $(0, 0, \ldots, 0)$, а й на багатьох інших. Наприклад:
$6 \oplus 3 \oplus 5 = 110_{Bin} \oplus 011_{Bin} \oplus 101_{Bin} = 000_{Bin} = 0$;
$7 \oplus 3 \oplus 4 = 111_{Bin} \oplus 011_{Bin} \oplus 100_{Bin} = 000_{Bin} = 0$;
$5 \oplus 0 \oplus 5 = 101_{Bin} \oplus 000_{Bin} \oplus 101_{Bin} = 000_{Bin} = 0$.

Повинно ніколи не існувати ходів із програшної позиції у програшну (це рівносильно із «всі ходи із програшної позиції ведуть лише до виграшних»). З цим трохи складніше, але не набагато. На рівні побітового ксора, це означає: якщо з позиції, побітовий ксор якої = 0, існують якісь ходи, то всі вони ведуть в такі позиції, побітовий ксор яких $\neq 0$. Хід у грі Нім означає, що в якійсь купці зменшилася кількість паличок. «Кількість зменшилася» означає, що у відповідному числі хоч якісь біти змінилися (якби ніякі біти не змінилися, то й подане ними число не змінилося б). Візьмемо будь-який один з тих двійкових розрядів, де біт змінився. Оскільки палички беруть лише з однієї купки, цей біт змінився лише в одному з чисел. Враховуючи згадану властивість логічного ксора «результат визначається тим, парна чи непарна кількість одиниць», це означає, що відповідний біт змінився. А коли деякий біт змінився, то змінилося й усе число (зміни значень інших бітів не можуть компенсувати зміну цього).

Нарешті, з кожної виграшної позиції повинен існувати хід у програшну. Тобто, якщо в деякої позиції побітовий ксор $\neq 0$, то з неї повинен бути хоча б один хід у позицію, де побітовий ксор $= 0$. От і розглянемо довільну позицію $(s_1, s_2, \ldots, s_k)$ таку, що $s_1 \oplus s_2 \oplus \ldots \oplus s_k = r \neq 0$, подумаємо, як її змінити, щоб це був допустимий правилами гри Нім хід і виходила позиція, для якої побітовий ксор $= 0$.

«Допустимий правилами гри Нім хід» передбачає, що з чисел s_1, s_2, …, s_k деяке одне $s_{i^\star}$ має замінитися на деяке $s^\star_{i^\star}$ $(0 \leqslant s^\star_{i^\star} < s_{i^\star})$, а решта s_1, …, $s_{i^\star-1}$, $s_{i^\star+1}$, …, s_k мають лишитися незмінними. А якщо зміною лише одного числа́ $s_{i^\star}$ потрібно змінити значення $s_1 \oplus s_2 \oplus \ldots \oplus s_k$ з $r \neq 0$ на 0, то $s_{i^\star}$ потрібно замінити на $s_{i^\star} \oplus r$.

Чому? Перетворимо аналітично $s_1 \oplus \ldots \oplus s_{i^\star-1} \oplus (s_{i^\star} \oplus r) \oplus s_{i^\star+1} \oplus \ldots \oplus s_k$. Враховуючи асоціативність та комутативність ксора, дію "$\oplus r$" можна перенести в кінець, решта виразу якраз і є тим $s_1 \oplus s_2 \oplus \ldots \oplus s_k$, яке дорівнює r, звідки $s_1 \oplus \ldots \oplus s_{i^\star-1} \oplus (s_{i^\star} \oplus r) \oplus s_{i^\star+1} \oplus \ldots \oplus s_k = (s_1 \oplus \oplus s_2 \oplus \ldots \oplus s_k) \oplus r = r \oplus r = 0$. Тобто, замінити $s_{i^\star}$ на $s_{i^\star} \oplus r$ правильно (справді веде до програшної

позиції). Чи може бути правильним замінити $s_{i^\star}$ не на $s_{i^\star} \oplus r$, а на $s_{i^\star} \oplus u$ при деякому $u \neq r$? Ні, це не буде ходом до програшної позиції ні при якому $u \neq r$, бо $s_1 \oplus \ldots \oplus s_{i^\star-1} \oplus (s_{i^\star} \oplus u) \oplus s_{i^\star+1} \oplus \ldots \oplus \oplus s_k = (s_1 \oplus s_2 \oplus \ldots \oplus s_k) \oplus u = r \oplus u \neq 0$.

З досі сказаного неясно, як знайти $i^\star$ (в *якій* з купок слід замінити s_i на $s_i \oplus r$, забравши з неї $s_i - s_i \oplus r$ паличок). Відповідь досить проста (не факт, чи зручна, але проста): ***слід перебрати різні чи́сла*** (в початкових термінах гри Нім — різні купки)***, щоб знайти таке число*** (купку)***, де $s_i \oplus r < s_i$; якщо потрібні всі «виграшні ходи»*** (які ведуть до програшних позицій)***, то слід перебирати всі чи́сла*** (купки)***, а якщо досить будь-якого одного «виграшного ходу», то цикл можна обривати (`break`) після першого знайденого.***

(Можна придумати, як це пришвидшити. Але то вже питання алгоритмів та структур даних, а не теорії ігор, тож тут не розглядається.)

Тут природні питання «а що, дія $s_i \oplus r$ при тому са́мому r справді може зменшувати одні s_i, але збільшувати інші?» та «а що, як жодного не знайдемо? (от перебрали всі-всі чи́сла (купки), а ситуація $s_i \oplus r < s_i$ так і не настала…)».

Відповідаю. На рівні окремого біта дія $a \oplus 1$ може як збільшити значення a ($0 \oplus 1 = 1$), так і зменшити ($1 \oplus 1 = 0$), тому нема ніц дивного в тому, що й на рівні побітового ксора над числами $s_i \oplus r$ може зменшувати одні s_i, але збільшувати інші. Наприклад, $6 \oplus 3 = 110_{Bin} \oplus 011_{Bin} = 101_{Bin} = = 5$, але $4 \oplus 3 = 100_{Bin} \oplus 011_{Bin} = 111_{Bin} = 7$ (отже, $6 \oplus 3 < 6$, але $4 \oplus 3 > 4$).

А щодо «жодного не знайдемо» — якщо правильно порахували $r = s_1 \oplus s_2 \oplus \ldots \oplus s_k$ і вийшло $r \neq 0$, то хоча б одне $s_i \oplus r < s_i$ знайдеться. Доведемо це. Знову розглянемо (1), де рахували ксор (ці дії повторено з невеличким доповненням у (1*); нехай тепер побітово ксоряться у стовпчик не любо-які чи́сла, а s_1, s_2, …, s_k).

(Хтось може сказати «навіщо нам ксор у стовпчик, якщо у програмі операція "^"?». Відповідаю. Коли числа невід'ємні в межах цілого типу, операція "^" і ксор у стовпчик дають однаковий результат. Тому, можна вибирати зручнішний спосіб на свій розсуд, і найзручніше мати операцію "^" у програмі, але ксор у стовпчик у доведенні.)

Оскільки вийшло $r \neq 0$ — значить, у результаті (рядку під рискою) є хоча б одна одиниця. От і розглянемо розряд, у якому утворена ця одиниця, а якщо результат містить кілька одиниць, то розглянемо розряд найлівішої (найстаршої) з одиниць.

$$\begin{array}{rcl} 321\,Dec & = & 101000001\,Bin \\ 283\,Dec & = & 100011011\,Bin \\ 126\,Dec & = & 001\mathbf{1}11110\,Bin \\ \hline & = & 000\mathbf{1}00100\,Bin \quad = \quad 36\,Dec; \end{array} \tag{1*}$$

Вже згадувалося, що результат логічного ксора (отже, й розряд результату побітового ксора) дорівнює 1 тоді й тільки тоді, коли проксорили непарну кількість одиниць. А непарна кількість — це, наприклад, одна, або три, або п'ять штук, …, але *не* 0 штук. Отже, хоча б у одному (або у трьох, або у п'яти, …) з чисел s_1, s_2, …, s_k той самий біт, який є найстаршим з рівних 1 у r, теж дорівнює 1. Отже, якщо проксорити з r са́ме це число, то в ньому са́ме цей розряд (біт) перетвориться з 1 на 0; отже, зменшиться.

(Як у (2), але ж (2) — лише приклад з конкретними числами, а попередні міркування — загальне доведення для всіх чисел.)

$$\begin{array}{rcl} 126\,Dec & = & 001\mathbf{1}11110\,Bin \\ 36\,Dec & = & 000\mathbf{1}00100\,Bin \\ \hline & = & 001\mathbf{0}11010\,Bin \quad = \quad 90\,Dec. \end{array} \tag{2}$$

При цьому ще старші розряди (якщо вони є) незмінні, бо ми розглядати розряд найстаршої одиничи в r (отже, для ще старших розрядів буде $a \oplus 0 = a$, де a — біт з s_i, 0 — біт з r). Неважливо, чи молодші двійкові розряди (якщо вони є) при цьому теж зменшуватимуться, чи лишатимуться незмінними, чи навіть збільшуватимуться, бо для всіх однорідних позиційних систем числення, включно з двійковою, збільшення значень молодших розрядів не може «пересилити» зменшення старшого розряду.

Підсумуємо, як грати в Нім: маючи позицію s_1, s_2, …, s_k, перш за все слід перевірити, чи це вже кінець гри ($s_1 = s_2 = \ldots = s_k = 0$), чи ще є палички; якщо є, слід обчислити $r = s_1 \oplus s_2 \oplus \ldots \oplus s_k$; якщо $r = 0$, позиція програшна, і слід походити як-небудь, наприклад, випадково вибравши хід серед допустимих (див. також тут); інакше ($r \neq 0$), позиція виграшна, тож слід знайти «виграшний хід у програшну позицію», а щоб його знайти, слід обчислювати $s_1 \oplus r$, $s_2 \oplus r$, …, доки не трапиться деяке $i^\star$ таке, що $s_{i^\star} \oplus r < s_{i^\star}$; коли воно трапиться, потрібно *залишити* у цій купці $s_{i^\star} \oplus r$ паличок, тобто забрати з неї $s_{i^\star} - (s_{i^\star} \oplus r)$ паличок.

Цей підсумок правильний, випливає з доведеного вище, але не претендує на ідеальність і є лише одним з кількох можливих способів.

Задача 1.3:A. «Нім — 0»

Є три ку́пки, кожна з яких містить деяку кількість паличок. (Далі описані правила гри Нім, але з поправкою, що в цій задачі купок рівно три.)

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Єдиний рядок містить рівно три цілих числа́, кожне в діапазоні від 1 до 40 — початкові кількості паличок у кожній з купок. Серед них можуть бути як однакові, так і різні.

Результати. Єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо дру́гий).

Приклади:

Вхідні дані	Результати
2 3 4	1
2 5 5	1
1 2 3	2

Розбір задачі. З усього досі описаного очевидно, що потрібно лише прочитати три числа в цілі змінні `m1`, `m2`, `m3` і якщо `m1^m2^m3==0` вивести 2, інакше 1, бо при `m1^m2^m3==0` 1-й гравець починає з програшної позиції, інакше з виграшної.

Задача 1.3:B. «Нім — 1»

(Правила гри Нім описані тут.)

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Перший рядок містить єдине ціле число k ($1 \leqslant k \leqslant 123$) — кількість купок. Дру́гий рядок містить рівно k чисел m_1, m_2, …, m_k, розділених одинарними пропусками (пробілами) — початкові кількості паличок у кожній з купок. Всі чи́сла m_1, m_2, …, m_k є цілими, у межах від 1 до 123456, серед них можуть бути як однакові, так і різні.

Результати. Перший рядок має містити єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо дру́гий).

Приклади:

Вхід	Рез-ти
2 3 4	1
2 5 5	2

Вхід	Рез-ти
3 1 2 3	2
2 4 8	1

Розбір задачі. Відрізняється від попередньої 1.3:A *лише* тим, що $s_1 \oplus s_2 \oplus \ldots \oplus s_k$ треба обчислювати для різних k, отже в циклі (як завжди, з 1-го курсу починаючи, обчислювали в циклі суму/добуток, просто тепер “^=” замість “+=” чи “*=”).

Задача 1.3:C. «Нім — 2»

Єдина відмінність від попередньої задачі 1.3:B «Нім–1»: якщо виграє перший гравець, то програма повинна знайти також сукупність усіх його виграшних перших ходів. Формат вхідних даних цілком ідентичний задачі 1.3:B «Нім–1».

Результати. Перший рядок має містити єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо дру́гий). Якщо відповідь з першого рядка `2`, то на цьому виведення слід припинити. А якщо відповідь з першого рядка `1`, то далі треба вивести також перелік всіх можливих перших ходів першого гравця, після яких дру́гий (при правильній грі першого) вже ніяк не зможе виграти. Цей перелік виводити в такому форматі: кожен такий хід в окремому рядку; кожен хід записується як пара чисел через пропуск: спочатку з якої купки слід забрати палички, потім скільки штук паличок треба забрати; якщо є багато різних виграшних перших ходів, вони повинні бути відсортовані за зростанням номера купки, а якщо є багато різних виграшних перших ходів, де палички беруться з однієї й тієї ж купки, то за зростанням кількості паличок, що беруться.

Кількість перших виграшних ходів виводити не треба. Вважати, що купки занумеровані з одиниці: 1, 2, …, k.

Приклади:

Вхід	Рез-ти
2 3 4	1 2 1
2 5 5	2

Вхід	Рез-ти
4 1 2 5 7	1 1 1 3 1 4 1

Примітка. В умові ϵ одне дрібне неможливе й непотрібне уточнення. Приблизно у стилі «якщо $2+2=5$, то виведіть слово ”хрпщ”» — його виводити все’дно не доведеться, бо все’дно $2+2\neq 5$. Знайти це неможливе й непотрібне уточнення — одне із завдань, які дуже бажано зробити у процесі розв’язування цієї задачі.

Розбір задачі. Відрізняється від попередньої 1.3:B *лише* тим, що потрібно також обчислювати $s_1 \oplus r$, $s_2 \oplus r$, …, $s_k \oplus r$ і запам’ятовувати (наприклад, дописувати в `List`) або відразу виводити всі ті $s_i - (s_i \oplus r)$, для яких $s_i \oplus r < s_i$. Усе це вже описано в розд. 1.3.2 й підсумовано тут, нема смислу повторювати втретє.

А неможливе й непотрібне уточнення — «якщо є багато різних виграшних перших ходів, де палички беруться з однієї й тієї ж купки, то …». Не може бути «різних виграшних ходів» для однієї купки. Їх або нема ні для якої купки (бо позиція програшна), або нема са́ме для цієї купки (бо $s_i \oplus r > s_i$), або такий хід лише один: забрати з цієї купки $s_i - (s_i \oplus r)$ паличок, залишивши $s_i \oplus r$.

Задача 1.3:D. «Нім — інтерактив»

(Правила гри Нім описані тут і цілком ідентичні вжитим у задачах 1.3:B–1.3:C.)

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Протокол взаємодії. На початку, один раз, Ваша програма повинна прочитати початкову позицію гри. Перший рядок містить єдине ціле число k ($1 \leqslant k \leqslant 12$) — кількість купок. Дру́гий рядок містить рівно k чисел m_1, m_2, …, m_k, розділених одинарними пропусками (пробілами) — початкові кількості паличок у кожній з купок. Всі чи́сла m_1, m_2, …, m_k є цілими, у межах від 1 до 1234, серед них можуть бути як однакові, так і різні.

Далі слід повторювати такий цикл:

1. Вивести два числа́, розділені пропуском, у окремому рядку — свій хід, тобто номер ку́пки, з якої вона забирає палички, та кількість паличок, які вона забирає. Це повинно бути ціле число від 1 до поточної кількості паличок у відповідній купці (обидві межі включно). Купки занумеровані з одиниці: 1, 2, …, k. Якщо внаслідок попередніх ходів деякі купки вже стали порожніми, всі купки зберігають свої початкові номери.
2. Якщо це була остання купка й вона після цього стає порожньою, вивести окремим рядком фразу “`I won!`” (без лапок, символ-у-символ згідно зразку) й завершити роботу.

3. Інакше, прочитати хід програми-суперниці, в такому ж форматі: номер ку́пки, з якої вона забирає палички, одинарний пропуск (пробіл), кількість паличок, які вона забирає. Гарантовано, що хід допустимий: купка з таким номером (нумерація з одиниці) існує й містить хоча б одну паличку, а кількість паличок є цілим числом від 1 до поточного залишку паличок у купці (обидві межі включно). Само собою, ця гарантія дійсна лише за умови, що Ваша програма правильно визначила, що гра ще не закінчилася.
4. Якщо це була остання купка й вона після цього стає порожньою, вивести окремим рядком фразу "`You won...`" (без лапок, символ-у-символ згідно зразку) й завершити роботу.

Все це слід повторювати, доки не будуть забрані всі палички з усіх купок (тобто, доки хтось не ви́грає). Програма-суперниця не виводить фраз "`I won!`" / "`You won...`" чи якихось їх аналогів.

Приклад:

Вхід, суперник	Ваша програма
2 3 4 2 2 1 1	 2 1 1 2 2 1 I won!

Примітки. Ніби порожні рядки між ходами зроблені, щоб краще було видно, хто коли ходить; вводити/виводити їх не треба.

Хід гри з прикладу можна прокоментувати так. Є дві купки, в одній 3 палички, в іншій 4. Позначимо як (3,4). Ваша програма забирає 1 паличку з купки №2, лишається (3,3); програма-суперниця забирає 2 палички з купки №2, лишається (3,1); Ваша програма забирає 2 палички з купки №1, лишається (1,1); програма-суперниця забирає останню 1 паличку з купки №1, лишається (0,1); Ваша програма забирає останню 1 паличку з останньої купки №2, повідомляє про свій виграш і завершує роботу.

Оцінювання. Оцінювання поблокове, тобто для нарахування балів за блок потрібно, щоб пройшли усі тести цього блоку. Тест з умови при перевірці не використовується. У п'яти блоках тестів (усіх, крім останнього) Ваша програма матиме справу з ідеальною програмою-суперницею, яка не робить помилок.

10% балів припадає на блок № 1, де $k = 1$ (в усіх тестах цього блоку можна і треба виграти).

Ще 15% балів припадає на блок № 2, де $k = 2$.

Ще 15% балів припадає на блок № 3, де $k = 3$.

Ще 15% на блок № 4, де $1 \leqslant k \leqslant 12$.

У кожному з блоків №№2–4 є як тести, де Ваша програма може і повинна вигравати, так і тести, де Ваша програма не має шансів виграти і повинна просто гідно, дотримуючись правил гри, програти.

Ще 10% балів припадає на блок № 5, де $1 \leqslant k \leqslant 12$, причому в жодному з тестів цього блоку виграти не можна, потрібно *лише* гідно, дотримуючись правил гри, програвати.

Решта 35% припадає на останній блок № 6, де Вашій програмі слід мати справу з різними програмами-суперницями, які грати не вміють — роблять ходи, які дотримуються формальних правил гри, але можуть вибирати не найкращий з допустимих ходів, дотримуючись кожна своїх власних уявлень про те, як варто грати в цю гру. Для проходження цього блоку Ваша програма має в кожному з тестів виграти, скориставшися (згідно правил гри) помилками чи іншими неадекватностями програми-суперниці. В цьому блоці № 6 всі тести відносно великі ($7 \leqslant k \leqslant 12$, всі $98 \leqslant m_i \leqslant$ $\leqslant 1234$).

За будь-яке порушення правил гри з боку Вашої програми, відповідний тест (а отже, й увесь відповідний блок) оцінюватиметься як не пройдений.

Розбір задачі. Знову майже все вже сказано раніше, але тепер це не лише поточний розд. 1.3 взагалі та попередні задачі 1.3:A–1.3:C зокрема, а крім них ще й розбір інтерактивної задачі 1.2:B.

1.3.3 Мі́зерний варіант гри Нім

Слово «мі́зерний» означає «гру з протилежною метою»; інакше кажучи, «у піддавки», прив'язуючись не до шашок, а до факту, що взяли гру і змінили у її правилах *лише* визначення, хто виграв.

Тобто, мі́зерний Нім, як і стандартний, передбачає, що спочатку є k купок з m_1, m_2, …, m_k паличками, на кожному ході можна забирати з будь-якої однієї купки будь-яку кількість паличок. Відмінність — хто забирає останню паличку з останньої купки, той *про*гра́є. Гравці знають про цю зміну правил гри і діють відповідно.

(Є свідчення, що коли у Нім грали для розваги до його аналізу Ч. Бутоном, зазвичай грали якраз у мі́зерний варіант. Однак, щодо як цих свідчень, так і щодо взагалі всіх свідчень про Нім до Ч. Бутона цілковитої впевненості нема: вони описують гру не досить детально та/або є сумніви щодо їх автентичності.)

Аналіз мі́зерного варіанту Німа частково (для $k=3$) даний сами́м Ч. Л. Бу́тоном безпосередньо у [17]; загальне ж формулювання таке:

$$\begin{gathered}\text{програшними є ті й тільки ті позиції, в яких}\\ s_1 \oplus s_2 \oplus \ldots \oplus s_k \oplus u(s_1, s_2, \ldots, s_k) = 0,\end{gathered} \tag{3}$$

де

$$u(s_1, s_2, \ldots, s_k) = \begin{cases} 0, & \max(s_1, s_2, \ldots, s_k) \geqslant 2,\\ 1, & \max(s_1, s_2, \ldots, s_k) \leqslant 1.\end{cases} \tag{4}$$

Доведення варто розділити на дві частини. Спочатку розглянемо ситуацію, коли від початку гри вже $\max(s_1, s_2, \ldots, s_k) \leqslant 1$, тобто або паличок від самого початку нема (тоді твердження виконується: отримання позиції зовсім без паличок є виграшною для мі́зерного варіанту позицією, і якраз $\underbrace{0 \oplus 0 \oplus \ldots \oplus 0}_{=0} \oplus 1 = 1 \neq 0$), або скрізь, де палички є, їх по одній у купці. Коли паличок по одній у купці, має місце $s_1 \oplus s_2 \oplus \ldots \oplus s_k = 1 \oplus 1 \oplus \ldots \oplus 1$ (якщо серед s_1, s_2, …, s_k є нулі — вони не впливають на ксор, і їх можна викреслити, але кількість одиниць тоді менша k; позначимо її q).

$$\underbrace{1 \oplus 1 \oplus \ldots \oplus 1}_{q \text{ штук одиниць}} = q \bmod 2 = \begin{cases} 1, & q \text{ непа́рне},\\ 0, & q \text{ па́рне}.\end{cases} \tag{5}$$

Коли паличок по одній у купці, «забрати з купки будь-яку кіль-ть» вироджується у «забравши єдину паличку, зробити купку порожньою». Гравець по суті не має вибору («вибирати, яку купку зробити порожньою» — сумнівний вибір, бо ксор все'дно комутативний асоціативний, і все'дно кіль-ть непорожніх купок зменшиться рівно на одну). Тобто, починаючи з місця, де $\max(s_1, s_2, \ldots, s_k) = 1$, і до кінця гри результат (3) неминуче буде щоходу змінюватися «…, 0, 1, 0, 1, 0, 1» (закінчуючи одиницею, бо (3) містить фрагмент $\oplus\, u(\ldots)$), тож при па́рному q результатом (3) буде 1, а при непа́рному буде 0. Це якраз узгоджується з тим, що якщо кіль-ть одиничок непа́рна, то останню паличку-купку забере (і тим, враховуючи мі́зерність задачі, програ́є) сам гравець, якому дісталася позиція, оцінка (3) якої =0, а якщо кіль-ть одиничок па́рна, то останню паличку-купку забере і програ́є суперник, а гравець, якому дісталася позиція, оцінка (3) якої =1, ви́грає.

Щоб перейти до випадку $\max(s_1, s_2, \ldots, s_k) \geqslant 2$, варто наголосити: оскільки палички забирають, але не додають, неможливо походити із позиції, де $\max(s_1,\ s_2, \ldots, s_k) \leqslant 1$, у позицію, де $\max(s_1,\ s_2, \ldots, s_k) \geqslant 2$. Оскільки закінчується Нім (навіть мі́зерний) лише у позиції $(0, 0, \ldots, 0)$, це означає, що можливі лише два випадки: (а) взагалі весь час $\max(s_1,\ s_2, \ldots, s_k) \leqslant 1$, і цей випадок вже розібраний; (б) протягом гри відбувається *рівно один* перехід від $\max(s_1,\ s_2, \ldots, s_k) \geqslant 2$ до $\max(s_1,\ s_2, \ldots, s_k) \leqslant 1$, і цим можна й варто користуватися.

Враховуючи властивості максимума та правило, що вимагає на кожному кроці брати палички лише з однієї купки (зменшувати лише одне s_i), перехід від $\max(s_1,\ s_2, \ldots, s_k) \geqslant 2$ до $\max(s_1,\ s_2, \ldots, s_k) \leqslant 1$ неминуче означає, що при цьому деяке рівно одне $s_{i^\star}$ зменшується від старого значення $s_{i^\star} \geqslant 2$ до нового значення $0 \leqslant s_{i^\star} \leqslant 1$. Це означає, що *можна* втрутитися са́ме в цей перехід, і переробити його: якщо він за звичайним алгоритмом для Німа з «нормальним» завершенням давав 0, то нехай

тепер дає 1, а якщо давав 1, то нехай тепер дає 0. Оскільки старе значення було $s_{i^\star} \geqslant 2$, то після такого втручання зменшення лишається зменшенням (хоч і іншим), із цим усе гаразд.

Також, завдяки цій властивості, можна, поки $\max(s_1,\ s_2, \ldots, s_k) \geqslant 2$, дотримуватися стратегії «робити побітовий ксор нулем» (і при цьому спиратися на старе доведення для «нормального» Німа), а в момент переходу до $\max(s_1,\ s_2, \ldots, s_k) \leqslant 1$ почати дотримуватися стратегії «робити побітовий ксор кількостей одиницею, і тим самим робити $s_1 \oplus s_2 \oplus \ldots \oplus s_k \ \oplus\ u(s_1,\ s_2, \ldots,\ s_k)$ нулем» (і при цьому спиратися на нещодавнє доведення). ■

1.4 Числа Шпрага–Ґранді (Sprague–Grundy, SG)

(На жаль, прізвища авторів цієї ідеї транслітерують дуже вже по-різному: Sprague — Шпраг, Шпрег, Спрег, Спреґ, Спраг; Grundy — Ґранді, Гранді, Грюнді. Частково це пов'язано з тим, що сам R. Sprague жив у Німеччині й писав німецькою, але його предки з прізвищем Sprague жили і працювали у Британії, причому і в Лондоні (Англія), і в Единбурзі (Шотландія); в такій ситуації справді неясно, з якої мови краще транслітерувати. У випадку P. Grundy очевидно, що слід спертися на англійську, але є зразу два традиційно-холіварні питання «"G" — це "Ґ" чи "Г"?» та «"u", яке читається [ʌ] — це "а" чи "у"?». Я планую дотримуватися написань «Шпраг» (з німецької) та «Ґранді» (з англійської), але не наполягаю на цій версії.

R. Sprague та P. Grundy працювали над цією теорією не разом, а незалежно й ≈одночасно у 1930-х рр.)

Всі ігри цього розд. 1.4 задовольнятимуть обмеження розд. 1.2 («двох гравців», «послідовна», «дискретна», «детермінована», «скінче́нна&ациклічна», «з повною інформацією», «закінчується виграшем одного з гравців і програшем іншого, без нічиїх і без числового виграшу»), а крім цього ще й будуть «неупередженими» й матимуть «нормальну умову завершення» (див. розд. 1.1.7 та 1.1.8). Всі ці умови одночасно. Отже, це досить вузький клас ігор.

1.4.1 Функція mex [мекс]

Назва «mex» (читається [мекс]) є скороченням від «**M**inimum **EX**cluded value»; суть цієї функції така: аргументом її є *множина* цілих невід'ємних чисел, результатом — одне, найменше серед відсутніх у множині-аргументі, ціле невід'ємне число. При бажанні це можна записати також формулою:

$$\operatorname{mex}(A) \stackrel{\text{def}}{=} \min\{n \in \mathbb{N}_0 \mid n \notin A\}, \tag{6}$$

де $\mathbb{N}_0$ — множина цілих невід'ємних чисел. Для кращого сприйняття, наведу також приклади:

- $\operatorname{mex}(\{1,2,3,4,5\}) = \operatorname{mex}(\{2,5\}) = \operatorname{mex}(\varnothing) = 0$ (множи́ни різні, але в кожному з цих випадків 0 не належить множині,[4] тож можна взяти 0 як мінімальне серед взагалі всіх цілих невід'ємних).
- $\operatorname{mex}(\{0\}) = 1$ (0 не можна, бо $0 \in \{0\}$; наступне мінімальне 1, його можна).
- $\operatorname{mex}(\{0,1,2,5\}) = 3$ (чи́сла 0, 1 та 2 належать множині, тому не можуть бути результатом, а наступне ціле число 3 не належить, тому є результатом; наявність числа́ 5 ніяк цьому не заважає).
- $\operatorname{mex}(\{1,2,4,7,12,3,0,9,5\}) = 6$, бо наявність у множині кожного з чисел 0, 1, 2, 3, 4, 5 забороняє кожному з цих чисел можливість бути результатом mex, а наступне ціле число 6 відсутнє у множині, тому і є результатом.

Те, що в останньому прикладі чи́сла задані в якомусь дивному порядку, ніяк на це не впливає. Бо, за стандартними уявленнями (згаданими, зокрема, у [8, с. 47], [20, п. «Елемент множини»], та багатьох інших джерелах), множина може лише або містити елемент, або ні, причому порядок не впливає на результат. Водночас, це означає, що коли потреба рахувати mex виникає вже на рівні написання свого коду (програми), може бути смисл сортувати послідовність елементів, бо так значно легше пропускати наявні 0, 1, 2, … й «бачити», якого чергового значення вперше нема.

[4] хто забув, чому $0 \notin \varnothing$ — може просто повторити (наприклад, у [8, с. 47], чи тут, чи ще десь — джерел багато), що таке порожня множина; в будь-якому разі, для подальшого вивчення теорії Шпрага–Ґранді абсолютно необхідно розрізняти 0, $\{0\}$ і $\varnothing$

(Са́ме так. Са́ме «порядок не впливає на результат, тому відсортуємо». Це може здатися бздурами (навіщо сортувати, як не впливає?!), але насправді навпаки. Якби порядок впливав на результат, сортувати було б не можна, бо змінювало б результат. А раз не впливає — можна. Чи варто — інше питання, яке слід дослідити окремо. І са́ме тут виявляється, що варто, і в попередньому абзаці вже пояснено, чому: коли чи́сла виписані в порядку 0, 1, 2, 3, 4, 5, 7, 9, 12 — можна значно легше й швидше побачити, що 0, 1, 2, 3, 4, 5 наявні й першим відсутнім є са́ме 6.)

Ще одне тісно пов'язане з цим питання: чи можуть елементи повторюватися? Наприклад: чи взагалі має смисл запис $\mathrm{mex}(\{0,1,1,2,2,2,5\})$, і якщо має, то яке значення цього виразу? Як побачимо трохи далі, для цілей теорії Шпрага–Ґранді зручніше вважати, що са́ме в mex повтори можливі, причому кратність не впливає на результат (наприклад, $\mathrm{mex}(\{0,1,1,2,2,2,5\}) = \mathrm{mex}(\{0,1,2,5\}) = 3$). Водночас, в деяких інших питаннях цієї ж теорії Шпрага–Ґранді важливо враховувати чи то точну кількість, чи то парність повторень, а замінювати, наприклад, два входження на одне неприпусти́мо.

(Якби ми намагалися розібратися з поняттям mex повністю, слід було б продовжувати. Навіть у статті вікіпедії [19] згадано, що для функції mex є узагальнення. Наприклад, згідно (6) вийде «$\mathrm{mex}(\mathbb{N}_0)$ не існує, бо нема такого цілого невід'ємного, яке не належало б множині цих са́мих цілих невід'ємних», а згідно узагальнення цей вираз має деякий смисл. Так буває. Наприклад, $5-8=-3$, але переважна більшість учнів 1–4 класів думають, ніби «з 5 відняти 8 неможливо», бо не знають про від'ємні чи́сла.

Для знаходження $\mathrm{mex}(\mathbb{N}_0)$ (а також $\mathrm{mex}(\mathbb{N}_0\cup\{\mathrm{mex}(\mathbb{N}_0)\})$, тощо) потрібні не від'ємні чи́сла, а зовсім інше узагальнення поняття числа́, яке тільки в цій темі й розглядають. Однак, ми утримаємося від розгляду цього узагальненння. Більш того: вважатимемо, ніби чи́сла, з якими працює mex, поміщаються в тип `int`. Теоретично це неправда, але для потреб цієї дисципліни прийнятно.)

1.4.2 Рекурентне означення числа́ Шпрага–Ґранді для позиції гри

У розд. 1.2 ми співставляли кожній позиції її виграшність/програшність; тепер, замість виграшності/програшності будемо співсталяти кожній позиції деяке ціле невід'ємне число — оце са́ме число Шпрага–Ґранді. Протягом прочитання самого лише поточного розд. 1.4.2 може виникнути враження, ніби це якийсь на рівному місці переускладнений спосіб, який кінець кінцем дає лише те са́ме, що й виграшні/програшні позиції, тільки важче; прошу повірити, що далі в розд. 1.4.4 стане зрозуміло, що ці чи́сла якраз *дають* принципово нові (порівнюючи з виграшністю/програшністю) можливості; але, щоб отримати змогу розуміти ті нові можливості й користуватися ними, спочатку треба розібратися з простішими варіантами застосування.

Тут і далі, ***SG*** позначатиме число Шпрага–Ґранді (**S**prague–**G**rundy value). Ось означення, яке будемо використовувати замість означень виграшних та програшних позицій: ***Якщо з деякої позиції A можна за один хід одного гравця дійти до позицій B_1, B_2, …, B_q*** *(де* ***q*** *лише позначає кількість позицій, у які можна перейти з* ***A****, ніякого глибшого смислу нема),* ***то для знаходження $SG(A)$ треба знайти $SG(B_1)$, $SG(B_2)$, …, $SG(B_q)$*** *(тобто, SG кожної окремо з позицій B_1, B_2, …, B_q)* ***і взяти з них тех [мекс]***. Або, те са́мо формулою:

$$SG(A) = \mathrm{mex}(\{SG(B_1), SG(B_2), \ldots, SG(B_q)\}). \tag{7}$$

Як раніше вже згадано, $\mathrm{mex}(\varnothing)=0$; тому, частковий випадок «якщо з позиції нема жодного ходу, SG(такої позиції)$=0$» по суті задається щойно згаданим основним означенням. Однак, якщо проговорити це явно і згадати, що в поточному розд. 1.4 дозволені лише ациклічні ігри, стає зрозуміліше, чому таке означення SG *не* створює ніякого «зачарованого ко́ла» вигляду «Ви не взнаєте SG, бо, щоб знайти SG, треба знати SG».

Навпаки, це означення цілком конструктивне: спочатку визначається SG=0 для деяких із (можливо, всіх; в якійсь грі зручнішим буде одне, в іншій інше) тупикових позицій, потім SG деяких із позицій, з яких є ходи лише в тупикові, й так доки не будуть визначені SG усіх потрібних позицій. Цілком аналогічно тому, як у розд. 1.2 визначалася (ви/про)грашність, й може робитися в тому ж порядку. Причому це може бути хоч рекурсивний спосіб (рідко доцільний, але ж можливий), хоч «починати з дальніх від старту позицій і йти цикл(ом/ами) "назустріч ходам"…», але *без(!)* «оптимізаційного прийому».

(«Оптимізаційний прийом» *не* переноситься на знаходження SG, бо для оголошення деякої позиції A_i виграшною досить знати, що з неї є хід у деяку програшну $B^\star$, і байдуже, які ще ходи з A_i існують, а для чисел Шпрага–Ґранді так не можна: ну, відомо, що існує хід з A_i в $B^\star$; ну, відомо, що $SG(B^\star)=0$… і що з того, як для визначення $SG(A_i)$ потрібне не лише $SG(B^\star)$, а ще й SG кожної позиції, куди можна піти з A_i? Самі́ лише наведені тут факти дозволяють визначити, що $SG(A_i)>0$, але SG мусить бути конкретним числом (1, чи 2, чи 3, …), а не просто «ненульови́м».)

Взаємозв'язок між числами Шпрага–Ґранді та виграшністю/програшністю. ***Якщо для гри виконуються всі згадані на початку цього розд. 1.4 обмеження, має місце відповідність***

$$\begin{array}{l}\boldsymbol{SG(A)=0} \;\Leftrightarrow\; \boldsymbol{A}\text{ програшна},\\ \boldsymbol{SG(A)>0} \;\Leftrightarrow\; \boldsymbol{A}\text{ виграшна},\\ (\text{де }\boldsymbol{A}\text{ — позначення позиції},\\ \Leftrightarrow\text{ — «тоді й тільки тоді, коли»}).\end{array} \tag{8}$$

Важливо, що тут стверджується лише одностороннє слідування «якщо порахувати всі SG згідно з формулами (6)–(7), то для них виконуються (дотримані) співвідношення (8)»; зворотне ж твердження «якщо для деяких співставлених позиціям чисел дотримані співвідношення (8), то ці числа можна вважати числами Шпрага–Ґранді і для них виконуються співвідношення з формул (6)–(7)», взагалі кажучи, *не(!)* є гарантовано правильним.

Перейдемо до доведення.

$SG(A)=0$ може бути лише або коли ходів з позиції A взагалі не існує (враховуючи, що в цьому розд. 1.4 всі ігри мають нормальну умову завершення у смислі розд. 1.1.7, це означає програшність позиції A), або коли всі ходи з A ведуть в такі B_1, B_2, …, B_q, що серед $SG(B_1)$, $SG(B_2)$, …, $SG(B_q)$ нема жодної $SG(B_i)=0$. Все це якраз відповідає тому, що позиція програшна, коли або тим фактом, що гравцю дісталася ця позиція, він вже програв, або абсолютно кожен хід з неї веде до виграшної позиції (яка дістанеться супернику).

Аналогічно, $SG(A)>0$ означає, що $\mathrm{mex}(\{SG(B_1),\ SG(B_2),\ldots,\ SG(B_q)\})>0$, а цей mex додатний тоді й тільки тоді, коли серед $SG(B_1)$, $SG(B_2)$, …, $SG(B_q)$ є хоча б один 0. Що якраз відповідає тому, що позиція виграшна, коли існує хоча б один (можна й більше) хід з неї, що веде до програшної позиції (яка дістанеться супернику). Частини «фактом, що гравцю дісталася ця позиція, він виграв» тут нема, але для ігор з «нормальною умовою завершення» такий варіант неможливий, тому байдуже, чи він згаданий, чи ні.

1.4.3 Приклади підрахунку SG для деяких ігор з розд. 1.2.

Розглянемо повторно деякі (не всі) ігри з розд. 1.2, але знайдемо для позицій (тих са́мих) не виграшність/програшність, а числа Шпрага–Ґранді.

SG гри Баше. Розглянемо в точності класичну гру Баше згідно розд. 1.2.1 і побудуємо для її позицій чи́сла Шпрага–Ґранді.

$SG(0)=0$, бо коли паличок вже нема (0 штук), то нема ходів і застосовується $\mathrm{mex}(\varnothing)=0$.

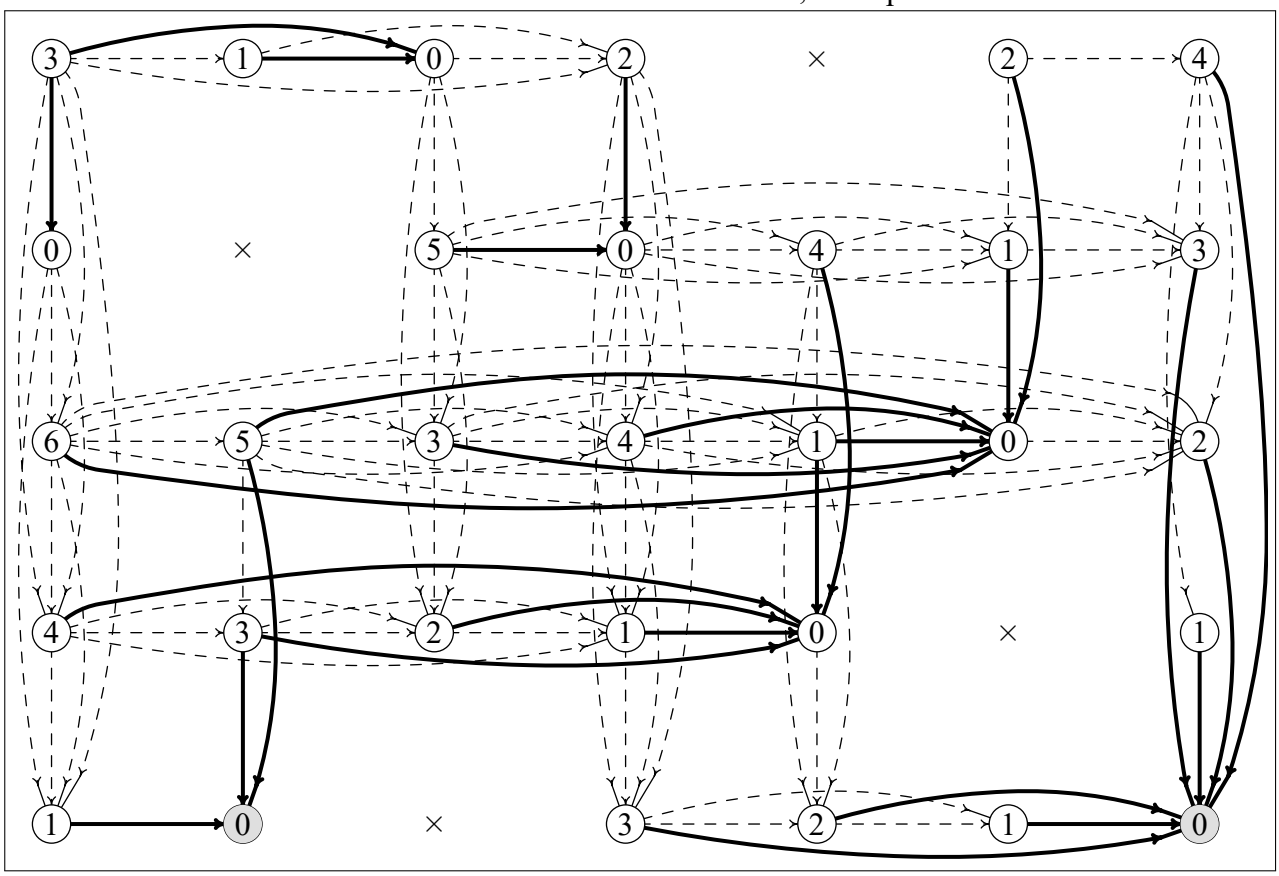
$SG(1)=1$, бо з позиції 1 (лишилася 1 паличка) є лише один хід (забрати цю паличку й потрапити в позицію 0), тож $SG(1)=\mathrm{mex}(\{SG(0)\})=\mathrm{mex}(\{0\})=1$.

$SG(2)=2$, бо з позиції 2 є два ходи (забрати обидві й потрапити в позицію 0, чи забрати лише одну й потрапити в позицію 1), тому $SG(2)=\mathrm{mex}(\{SG(0),SG(1)\})=\mathrm{mex}(\{0,1\})=2$.

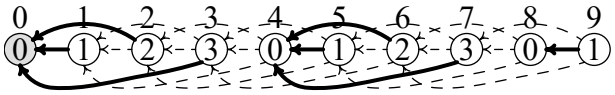
$SG(3)=3$, бо з позиції 3 є три ходи (забрати всі три й потрапити в позицію 0, чи забрати дві й потрапити в позицію 1, чи забрати одну й потрапити в позицію 2), тому $SG(3)=\mathrm{mex}(\{SG(0),\ SG(1),\ SG(2)\})=\mathrm{mex}(\{0,1,2\})=3$.

$SG(4)=0$, бо з позиції 4 є три ходи (забрати три й потрапити в позицію 1, чи забрати дві й потрапити в позицію 2, чи забрати одну й потрапити в позицію 3), тому $SG(4)=\mathrm{mex}(\{SG(1),\ SG(2),\ SG(3)\})=\mathrm{mex}(\{1,2,3\})=0$.

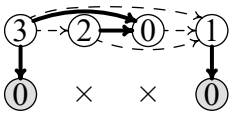
Далі все повторюється аналогічно, й очевидно, що $SG(i)=i \bmod 4$, де mod — залишок, той са́мий, який у C-подібних мовах позначається "%".



Граф гри, якщо позначити на ньому замість W/L значення SG, має такий вигляд:



SG «Фішки на мінному полі». Беремо «основний» варіант (наприклад, задачу 1.2:C), у якому той, хто не має ходу, програє, бо в цьому розд. 1.4 лише так і можна.



Від написання виразів вигляду $SG(\dots) = \mathrm{mex}(\{\dots\dots\dots\})$ цього разу утримаємося; пропонуємо провести відповідні міркування самостійно, дивлячись на наведені графи (один — тут ліворуч, інший — вгорі стор. 43).

Як бачимо, навіть на досить малому полі гри (див. граф ліворуч попереднього абзацу) можуть виникати ситуації, коли з деякої позиції (конкретно в тому випадку — лівої-верхньої) є ходи у різні позиції з однаковим числом Шпрага–Ґранді (конкретно в тому випадку — з лівої-верхньої клітинки-позиції можна піти хоч на 1 униз, хоч на 2 праворуч, обидві ці клітинки мають однакове $SG(\dots) = 0$. Природньо, що на більшому графі нагорі стор. 43 таких ситуацій більше, причому там повторюються не лише чи́сла $SG(\dots) = 0$, а й деякі інші. Тому раніше й говорилося, що для означення mex зручніше вважати, що множина може містити повтори елементів, які не впливають на її значення (як-то $\{0, 1, 2\} = \{0, 1, 0, 2\}$). Втім, щоб не тонути у суто теоретичній дискусії, можна сказати ще й так: дуже важливо написати свою програму так, щоб правильно знаходити mex і у випадку, коли серед значень $SG(B_1)$, $SG(B_2)$, …, $SG(B_q)$ повторень нема, і у випадку, коли такі повторення є.

1.4.4 Сума ігор, зв'язок суми ігор з грою Нім та числами Шпрага–Ґранді

Перш за все, варто зрозуміти, що хоч операцію, яку за́раз розглядаємо, і називають сумою ігор, вона зовсім *не* схожа на суму чисел, і назва «сума» поширена, але не є цілком загальноприйнятою.

(Наприклад, у [14, с. 12] є словосполучення «disjunctive sum of games». Але, наприклад, у статті «Теорема Шпрага–Ґранді» [21] на час написання цього тексту (березень'2024) є «поєднання ігор» та «об'єднана гра», а «суми ігор» нема

(зате є словосполучення «додавши позиції»). Причому, це не є проблемою лише uk-wiki (в en-wiki є підрозділ «Combining Games», де нема згадок про «games sum» чи «sum of games», але є фраза «games can be combined by adding their positions together»).

Якби я мав повну свободу вибору, як назвати це поняття — можливо, погодився б із «поєднанням ігор» (але тоді са́ме *по*єднанням; на об'єднання множин ця дія схожа ще менше, чим на суму чисел), але, ймовірно, віддав би перевагу назві «добуток ігор», бо з усіх стандартних математичних дій найближчою здається декартів добуток множин, хоч і він не зовсім відповідає теперішній дії. (Джерела, де просто описаний декартів добуток — зокрема, [8, с. 55], [22] та багато інших; джерело, де означення нинішньої операції дають з активним використанням декартового добутку — [15, с. 20].)

Не відповідає він, головним чином, тим, що у багатьох ситуаціях зручно вважати, що теперішня дія асоціативна комутативна: поєднуючи, наприклад, три гри, часто зручно вважати неважливим, яка з них перша, яка дру́га і яка третя, а декартів добуток категорично не комутативний (наприклад, крісло 19 у ряді 2 — безсумнівно інше, чим крісло 2 у ряді 19). Однак, сам факт, що ігри водночас і поєднуються, і лишаються в деякому смисл відрізнюваними одна від одної, настільки нагадує декартів добуток, де елементи водночас і поєднуються у пари/трійки/…, і водночас лишаються відрізнюваними складовими цих пар/трійок/…, що певну відмінність у питаннях комутативності й асоціативності я вважаю не дуже істотною. Кінець кінцем, можна сказати, що є декартів добуток, а є шпрагоґрандівський добуток, вони обидва поєднують сутності, водночас лишаючи їх відрізнюваними; щоправда, декартів і шпрагоґрандівський добутки не однакові, бо по-різному ставляться до деяких інших властивостей, як-то комутативності й асоціативності. І те, що ні Шпраг, ні Ґранді не використовували рівно такої термінології — сумно, але не дуже страшно. В науці є випадки, коли «іменні» речі нині позначають не зовсім те, що думали ті, чиїми іменами воно названо. Той самий Рене Декарт знав не всі тонкощі сучасного означення декартового добутку, а Евклід, найімовірніше, взагалі не розпізнав би, що сучасна версія алгоритма Евкліда має щось спільне з його ідеями.

Однак, усе написане тут дрібнішим шрифтом — лише мої власні думки й побажання. Чи хочеться мені переназвати цю операцію інакше, чи ні, а наразі найпоширенішим варіантом її назви здається все-таки «сума ігор», тому надалі використовую цю назву, хоч і маю щодо неї деякі сумніви.)

Нарешті, формальне означення: *сума ігор* — це операція, яка поєднує кілька (позначимо кількість як k) окремих ігор у одну; при цьому гра-сума включає в себе всі графи всіх поєднуваних ігор, поточні позиції в кожному з цих графів (гра-сума, у деякому смислі, має відразу k поточних позицій — по одній на кожну гру), а один окремий хід у грі-сумі передбачає, що з k поєднуваних ігор вибирається одна конкретна, і в ній робиться один окремий хід.

Відповідно, коли відома сума ігор, а треба сказати про окрем(у/і) з тих ігор, які утворюють цю суму — будемо казати «*підгра*» (коли одна), «*підігри*» (коли кілька/багато), або «*ігри-"доданки"*».

Рівно в наведеному вигляді означення може бути малозрозумілим, бо надто мало схожого хоч із програмуванням, хоч із більш відомими галузями математики. Тому пропоную усвідомити його, зайшовши з несподіваного боку: *вже розглянута у розд. 1.3* ***гра Нім якраз і є сумою ігор***.

Якими повинні бути ігри, щоб їх сума дала гру Нім? Кількома примірниками дуже тупої гри: є одна купка, в ній m паличок, і за один хід можна забирати абсолютно будь-яку кількість паличок від 1 до всіх. Хто забирає останню паличку — виграє́; хто не має ходів — програє́. В цій грі виграшну стратегію майже завжди має 1-й гравець: йому слід на першому ж своєму ході забрати всі палички й тим виграти; єдине виключенння — коли з самого початку $m = 0$, тоді ніхто не робить жодного ходу, і вважається, що програв 1-й гравець (відповідно, 2-й виграв).

Очевидно, що виграш у гру-суму не виражається через виграші в окремі ігри-«доданки» (під-ігри). Хоча б тому, що гра Нім вже проаналізована в розд. 1.3, й уже доведено, що єдиний (са́ме *єдиний(!)*, а *не* «допустимий, але крім нього є ще інш(ий/і)») спосіб гарантовано виграти — щоразу робити побітовий ксор кількостей паличок усіх купок рівним 0, й це не має нічого спільного з тим, щоб тупо забирати побільше паличок.

(Звісно, іноді трапляються такі рідкісні позиції, де «зробити побітовий ксор рівним 0» і «забрати побільше паличок» якраз дають однакові ходи; наприклад, для позиції $(100, 1, 1)$ обидва способи дають однакову вказівку забрати 100 з 1-ї купки… Але якщо говорити про універсальні алгоритмічні способи та про всі можливі стартові позиції, то спільного все-таки нема.)

Гібрид Німа й Баше — постановка задачі. Є k купок, кожна містить деяку кількість паличок. Ці кількості відомі гравцям; позначимо їх m_1, m_2, …, m_k. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з будь-якої однієї купки або 1, або 2, або 3 палички (але, звісно, не більше, чим їх є в цій купці). Палички можна лише забирати (ні додавати, ні перекладувати з купки в купку не можна). Ніяких інших варіантів ходу нема. Коли купка стає порожньою (кількість

паличок=0), гра просто продовжується для решти купок. Ходять гравці по черзі, пропускати хід не можна. Виграє́ той, хто забирає останню паличку (можливо, разом із ще деякими) з останньої купки. (Інакше кажучи, гра має «нормальну умову завершення»: хто не може походити — програє́.)

В цій грі не працюють ні стратегія для Німа, ні стратегія для Баше.

(Чому не працює стратегія для Баше? Хоча б тому, що ніякої «стратегії для Баше, але кількох купок» ми (поки що) взагалі не знаємо. От що робити, отримавши, наприклад, позицію $(2,3)$? «Забрати всі палички з однієї із купок, щоб "виграти в межах цієї купки"» нічого не дає, бо після цього суперник забере всі палички з іншої купки й ви́грає. Хоча з позиції $(2,3)$ можна (згідно зі стратегією для Німа) піти в позицію $(2,2)$ й потім виграти.

Чому не працює стратегія для Німа? Ну, порахуємо спочатку $r = m_1 \oplus m_2 \oplus \ldots \oplus m_k$, потім для кожного i значення $m_i - (m_i \oplus r)\ldots$ а воно візьме й не потрапить, взагалі ні для одного i, у проміжок «від 1 до 3». Наприклад, для позиції $(1,3,6)$: маємо $r = 1 \oplus 3 \oplus 6 = 001_{Bin} \oplus 011_{Bin} \oplus 110_{Bin} = 100_{Bin} = 4$, після чого $1 \oplus 4 = 001_{Bin} \oplus 100_{Bin} = 101_{Bin} = 5 > 1$ та $2 \oplus 4 = 010_{Bin} \oplus 100_{Bin} = 110_{Bin} = 6 > 2$ не дають корисних ходів, а $6 \oplus 4 = 110_{Bin} \oplus 100_{Bin} = 010_{Bin} = 2$ дало б корисний хід «забрати $6-2=4$ паличок» у грі Нім, але ж тепер заборонено забирати 4 палички одним ходом. І що далі?)

А чи працює спосіб «намагатися робити нулем $SG(s_1) \oplus SG(s_2) \oplus \ldots \oplus SG(s_k)$»? Так, працює. Але вже зрозуміти, про що мова, могли хіба люди з феноменальною інтуїцією, а решта потребу́ють детальнішого роз'яснення (й це нормально, тож цим і займемося).

Я за́раз намагатимусь паралельно розказувати і са́ме про гібрид Німа й Баше, і про суму ігор взагалі. Не всі міркування будуть справді строгими, але я намагатимусь і використати відносно простий приклад гібриду Німа й Баше для більшої наочності (де це доречно), й пояснити, чому міркування можна узагальнити на будь-яку суму ігор.

Гібрид Німа й Баше відрізняється від Німа не лише тим, що втрачено можливість забирати з кожної купки скільки завгодно паличок, а ще й тим, що всере́дині кожної купки є вже з'ясована залежність $SG(s_i) = s_i \bmod 4$. Для суми довільних ігор конкретні способи обчислення SG окремих підігор (ігор-«доданків») можуть бути різними, але для кожної підгри такий спосіб існує, і його можна з'ясувати, знаючи правила гри.

Якщо в гібриді Німа й Баше для деякої купки $i^\star$ має місце, наприклад, $SG(s_{i^\star}) = 2$, то в цій купці точно можна зробити хід «забрати 1 паличку» й отримати $SG(s_{i^\star} - 1) = 1$ та хід «забрати 2 палички» й отримати $SG(s_{i^\star} - 2) = 0$, а чи можна зробити хід «забрати 3 палички» невідомо: $SG(s_{i^\star}) = 2$ може бути і при $s_{i^\star} = 2$ (тоді неможливо, бо нема стільки паличок), і коли $s_{i^\star}$ становить 6, або 10, або 14, … (для них можна й вийде $SG(s_{i^\star} - 3) = 3$).

Може здатися, ніби ці міркування спираються на правила са́ме гри Баше й для них нема аналогу в довільній сумі довільних ігор… Але в тім-то й справа, що *деякий частковий аналог якраз є!*

Навіть не знаючи деталей способу/алгоритму обчислення $SG(s_{i^\star})$, ми з само́го лише факту $SG(s_{i^\star}) = 2$ знаємо (згідно (7)), що для $SG(s_{i^\star}) = 2$ необхідно, щоб з позиції $s_{i^\star}$ були ходи і в позицію, $SG(\text{якої}) = 0$, і в позицію, $SG(\text{якої}) = 1$. Це легко узагальнюється й на інші значення $SG(s_{i^\star})$: якщо $SG(s_{i^\star}) = v > 0$, то для кожного w з проміжку $0 \leqslant w \leqslant v-1$ існує хід в межах цієї підгри в позицію, $SG(\text{якої}) = w$ (яке б w із проміжку не взяли, хід у таку позицію існує; якби для деякого $w^\star$ ($w^\star < v$) ходу не існувало, mex повернув би $w^\star$ замість v).

(При цьому може існувати також хід у позицію, $SG(\text{якої}) > v$. Адже, наприклад, SG=2 цілком може бути отримане як, наприклад, $\text{mex}(\{0,1,4,5\})$, і тоді один хід зменшує SG до 1, інший зменшує до 0, але є й ще інші ходи, один з яких збільшує до 4, інший до 5.

Ще конкретніший приклад: у гібриді Німа й Баше можна, маючи 6 паличок у купці, забрати хоч 1 (лишити 5, $SG(5) = 1$), хоч 2 (лишити 4, $SG(4) = 0$), хоч 3 (лишити 3, $SG(3) = 3$).

Ходів у позиції з більшим SG може існувати навіть багато (чи то в різні позиції з однаковим більшим SG, чи то в позиції з різними більшими SG, чи то і так, і так водночас); однак, їх може й не існувати жодного. Нічого схожого на універсальний (відразу для всіх ігор) критерій розрізнення, в яких випадках існує хід у позицію з більшим SG, начебто, нема, бо тут дуже багато залежить від конкретної гри. Однак, якщо треба, такі критерії можна шукати для кожної гри окремо.

Само собою, не може бути ходу в позицію, $SG(\text{якої}) = v$, бо це протирічило б означенню mex як minimum *excluded*.)

Тепер розпишу детальніше вже згадане центральне твердження поточного пункту: ***у сумі ігор, програшними є ті й тільки ті позиції, в яких $SG(s_1) \oplus SG(s_2) \oplus \ldots \oplus SG(s_k) = 0$, де $SG(s_1)$, $SG(s_2)$, …, $SG(s_k)$ — чи́сла Шпрага–Ґранді підігор; відповідно, виграшними є всі інші позиції,***

а «виграшними ходами» (які віддають супернику програшну позицію) є ті й тільки ті ходи, які роблять ксор усіх SG рівним 0.

Тепер повторю з потрібними модифікаціями більшість міркувань розд. 1.3.2 (де доводили аналогічне твердженння просто для Німа, без SG).

Із принципом «позиція, якою гра закінчується, повинна бути програшною» все гаразд (поки гра має «нормальну умову завершення», а в усьому розд. 1.4 це так). Адже в сумі ігор нема ходів тоді й тільки тоді, коли ходів нема в жодній з підігор (ігор-«доданків»); коли у підгрі нема ходів, $SG(\text{відповідної підгри})=0$, тож $0\oplus 0\oplus\ldots\oplus 0=0$.

Втім, у грі Нім «позиція $(0,0,\ldots,0)$» і «нема ходів» були одним і тим са́мим, а тепер можливі ситуації, коли $SG(s_1)=SG(s_2)=\ldots=SG(s_k)=0$, але ходи є. Наприклад, так буде у гібриді Німа й Баше на позиції $(4,8,0,20,0)$: $SG(4)\oplus SG(8)\oplus SG(0)\oplus SG(20)\oplus SG(0)=0\oplus 0\oplus 0\oplus 0\oplus 0=0$, а у трьох з п'яти купок ходи ще є. Очевидно, схожі ситуації можливі й для інших сум ігор. Це не створює проблем, просто закінчення підігор слід перевіряти за смислом цих ігор, а не за їхніми SG.

(І взагалі, в більшості випадків варто зберігати позиції (під)ігор, якими вони є, з усіма особливостями їхніх правил. Чи краще обчислювати SG позицій підігор щоразу, коли знадобляться, чи зберігати як позиції, так і обчислені для них SG — залежить від особливостей гри (наскільки складно повторно обчислювати SG, наскільки збільшується обсяг пам'яті, якщо зберігати, тощо). Але спроби «один раз перейти до SG й надалі працювати лише з ними», хоч і бувають зручними й ефективними, але часто призводять лише до плутанини й неправильних результатів.)

Твердження «побітовий ксор може=0 не лише при $SG(s_1)=SG(s_2)=\ldots=SG(s_k)=0$, а й на інших позиціях» лишається правильним і важливим.

(Чого б воно могло стати неправильним? Там же просто наводилися приклади у стилі «$1\oplus 3\oplus 2=0$», в них ніц не змінилося, і для переважної більшості правил гри значення SG цілком можуть бути са́ме такими…)

У доведенні твердження «нема ходів з програшної у програшну, тобто з позиції, де $SG(s_1)\oplus\oplus SG(s_2)\oplus\ldots\oplus SG(s_k)=0$, у позицію, де теж $SG(\ldots)\oplus SG(\ldots)\oplus\ldots\oplus SG(\ldots)=0$» є деякі дрібні відмінності. Хід у сумі ігор є ходом у лише одній з підігор; згідно (7) та (6), при ході SG змінюється (несуттєво, зменшується чи збільшується, але точно змінюється; якби не змінилося, це протиріччило б означенню mex як minimum *excluded*), отже біти лише одного аргумента побітового ксора змінюються, а тому й результат побітового ксора змінюється.

Далі в розд. 1.3.2 доводилося «з будь-якої виграшної позиції існує хоча б один хід у програшну» і давався конструктивний спосіб знайти всі такі ходи: «спочатку порахувати $r=s_1\oplus s_2\oplus\ldots\oplus\oplus s_k$, потім шукати так(е/і) i, щоб виходило $s_i\oplus r<s_i$», й пояснювалося, чому «при $r\neq 0$ так(е/і) i (одне, або три, або п'ять, …) обов'язково знайдеться». Тепер твердження змінюється на «знайти $r=SG(s_1)\oplus SG(s_2)\oplus\ldots\oplus SG(s_k)$ й шукати таку підгру $i^\star$, де з позиції $s_{i^\star}$ існує хід у таку позицію $t_{i^\star}$, щоб виконувалося $SG(t_{i^\star})=SG(s_{i^\star})\oplus r$». Враховуючи вже наведене пояснення, чому з позиції з більшим SG існують ходи в позиції з усіма меншими SG, обов'язково існує (одне, або три, або п'ять, …) таке $i^\star$ з властивістю $SG(t_{i^\star})<SG(s_{i^\star})$, з абсолютно тих са́мих причин, що й у розд. 1.3.2.

Щоправда, тепер додатково можуть з'явитися (а можуть і не з'явитися; залежить від правил конкретної гри та від конкретної позиції) також ситуації (неможливі для гри Нім), коли додатково існує ще й такий хід (чи кілька ходів), який теж робить побітовий ксор усіх SG нулем, але $SG(t_{i^\star})>>SG(s_{i^\star})$.

(Наприклад, у Німі з позиції $(1,2)$ є лиш один «виграшний хід»: узяти 1 паличку з 2-ї купки і тим прийти у позицію $(1,1)$. Але у гібриді Німа й Баше з позиції $(5,2)$ (яка при поелементному взятті SG дає ту ж пару $(1,2)$) є вже два «виграшні ходи». Один — аналогічно привести пару SG до $(1,1)$, таким са́мим зменшенням дру́гого SG на 1 (взяти 1 паличку з 2-ї купки, і тим перейти у позицію $(5,1)$). А ще один «виграшний хід» — узяти 3 палички з 1-ї купки, тобто перейти до позиції $(2,2)$, змінивши тим пару SG з $(1,2)$ на $(2,2)$.

І все це — у досить простому гібриді Німа й Баше, для інших сум ігор все може бути ще складніше.)

Підсумую досі сказане.

- Коли треба лише довести/перевірити факт виграшності, зміни прості: слід вчасно бра́ти SG позицій окремих підігор і порівнювати з нулем побітовий ксор $r=SG(s_1)\oplus SG(s_2)\oplus\ldots\oplus\oplus SG(s_k)$.

- Коли треба знайти будь-який один «виграшний хід» (як завжди, це хід, що веде у програшну позицію, яка дістанеться супернику; при $r \neq 0$ такий хід гарантовано існує серед тих i, де $SG(s_i) \oplus \oplus r < SG(s_i)$), зміни складніші й потребу́ють більшого перебору або більших запам'ятовувань, бо слід ще визначити той хід, який веде у позицію, $SG(\text{якої}) = SG(s_i) \oplus r$, а перетворювати SG у хід у більшості ігор складно.
- Коли потрібні *всі* «виграшні ходи» (в тому ж смислі), перебирати та/або запам'ятовувати треба ще більше, бо і можливі різні ходи, що приводять у позиції з однаковими SG, і не можна бути впевненим, чи у випадку $SG(s_i) \oplus r > SG(s_i)$ нема ходів, що ведуть у позицію, $SG(\text{якої}) = SG(s_i) \oplus r$, чи са́ме в цьому випадку (він/вони) є.

(Для загадних запам'ятовувань може бути помічним, наприклад, `Dictionary`, ключами якого є значення SG, а значеннями — списки (`List`-и) позицій, які мають таку SG (див. також розбір задачі 1.4:E). Але деталі цього сильно залежать від особливостей конкретної гри, і взагалі це питання алгоритмів та структур даних, а не теорії ігор.)

Корисним для розуміння щойно сказаного може бути також цей приклад відновлення ходів у грі Ґранді, де вжиті, по суті, щойно вказані міркування. Однак: (1) приклад може лише ілюструвати правило, але не може його замінити; (2) той приклад не для «гібриду Німа й Баше», а для гри Ґранді, тож, щоб зрозуміти приклад, треба спочатку зрозуміти правила тієї гри.

Задача 1.4:A. «Гібрид Німа й Баше — 1»

(Правила цієї гри описані тут. Вони відрізняються від правил гри Нім в точності тим, що за хід можна брати не більше 3-х паличок.)

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Перший рядок містить єдине ціле число k ($1 \leqslant k \leqslant 123$) — кількість купок. Дру́гий рядок містить рівно k чисел m_1, m_2, …, m_k, розділених одинарними пропусками (пробілами) — початкові кількості паличок у кожній з купок. Всі чи́сла m_1, m_2, …, m_k є цілими, у межах від 1 до 123456, серед них можуть бути як однакові, так і різні.

Результати. Перший рядок має містити єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо дру́гий).

Приклади:

Вхід	Рез-ти
2 3 4	1
2 5 5	2

Вхід	Рез-ти
3 1 2 3	2
2 4 8	2

Задача 1.4:B. «Гібрид Німа й Баше — інтерактив»

(Ця задача відрізняється від задачі 1.3:D «Нім — інтерактив» в точності тим, що за один хід можна забирати не більше 3-х паличок.)

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Протокол взаємодії. На початку, один раз, Ваша програма повинна прочитати початкову позицію гри. Перший рядок містить єдине ціле число k ($1 \leqslant k \leqslant 12$) — кількість купок. Дру́гий рядок містить рівно k чисел m_1, m_2, …, m_k, розділених одинарними пропусками (пробілами) — початкові кількості паличок у кожній з купок. Всі чи́сла m_1, m_2, …, m_k є цілими, у межах від 1 до 1234, серед них можуть бути як однакові, так і різні.

Далі слід повторювати такий цикл:

1. Вивести два числа́, розділені пропуском, у окремому рядку — свій хід, тобто номер ку́пки, з якої вона бере палички, та кількість паличок, які вона бере. Кількість мусить бути цілим числом від 1 до 3 (обидві межі включно) і не перевищувати поточної кількості паличок у цій купці. Купки занумеровані з одиниці: 1, 2, …, k. Якщо внаслідок попередніх ходів деякі купки вже стали порожніми, всі купки зберігають свої початкові номери.

2. Якщо це була остання купка й вона після цього стає порожньою, вивести окремим рядком фразу “`I won!`” (без лапок, символ-у-символ згідно зразку) й завершити роботу.
3. Інакше, прочитати хід програми-суперниці, в такому ж форматі: номер ку́пки, з якої вона бере палички, пропуск, кількість паличок, які вона бере. Гарантовано, що хід допустимий: купка з таким номером (нумерація з одиниці) існує й непорожня, а кількість паличок є цілим числом, не меншим 1 і не більшим ні 3, ні поточної кількості паличок у цій купці. Само собою, ця гарантія дійсна лише за умови, що Ваша програма правильно визначила, що гра ще не закінчилася.
4. Якщо це була остання купка й вона після цього стає порожньою, вивести окремим рядком фразу “`You won...`” (без лапок, символ-у-символ згідно зразку) й завершити роботу.

Все це слід повторювати, доки не будуть забрані всі палички з усіх купок (тобто, доки якась із програм не ви́грає). Програма-суперниця не виводить фраз “`I won!`” / “`You won...`” чи якихось їх аналогів.

Приклад:

Вхід, суперник	Ваша програма
2 4 8 2 3 1 2 2 3	 2 1 1 2 2 1 You won...

Примітка. Ніби порожні рядки між ходами зроблені щоб краще було видно, хто коли ходить; вводити/виводити їх не треба.

Загальний хід гри з прикладу можна прокоментувати так. Є дві купки, в одній 4 палички, в іншій 8. Позначимо як (4,8). Ваша програма забирає 1 паличку з купки №2, лишається (4,7); програма-суперниця забирає 3 палички з купки №2, лишається (4,4); Ваша програма забирає 2 палички з купки №1, лишається (2,4); програма-суперниця теж забирає 2 палички з купки №1, лишається (0,4); Ваша програма забирає 1 паличку з купки №2, лишається (0,3); програма-суперниця забирає всі 3 палички з купки №2 і цим виграє́; Ваша програма повідомляє, що вона помітила, що гра закінчилася виграшем програми-суперниці, й завершує роботу.

У звичайному Німі (без обмеження на кількість паличок, які можна забирати за раз) Вашій програмі варто було б почати з ходу `2 4`, після якого лишилося б (4,4). Однак, у цій задачі не можна забирати відразу 4 палички, тому такий хід неможливий.

Оцінювання. Оцінювання цієї задачі потестове (бали за кожен тест ставляться окремо й незалежно від інших тестів), програма-суперниця завжди ідеальна (якщо Вашій програмі дісталася програшна позиція, вона може *лише* гідно, дотримуючись правил гри, програти, а перехопити ініціативу неможливо). Однак, Вам слід забезпечити, щоб Ваша програма в будь-якому разі коректно доводила процес до кінця (виграшу чи програшу).

За будь-яке порушення правил гри з боку Вашої програми, відповідний тест оцінюватиметься як не пройдений.

Розбір задач 1.4:A, 1.4:B. Якщо хотіти лише вказівок без пояснень, то: «У 1.4:A, робити як у 1.3:B, але спочатку взяти від кожної кількості паличок “`s[i]%=4`”. У 1.4:B, робити як у 1.3:D, але з урахуванням попереднього речення, а також пам’ятаючи, що кінець гри настає лише за умови $s_1 = s_2 = \ldots = s_k = 0$.». Якщо хотіти все-таки пояснень (*чому* так?) — нема іншої ради, як перечитати все від початку поточного розд. 1.4.4.

Задача 1.4:C. «Кілька фішок на мінному полі»

Правила гри переважно відповідають задачі 1.2:C «Фішка на мінному полі—1» (включно з «фішка може рухатися або праворуч, або вниз, на будь-яку кількість клітинок у межах по́ля, не стаючи

на міни й не перестрибуючи їх» та «нормальною умовою завершення», тобто «хто не може ходити — програє»), але спочатку маємо k фішок, які стоять на різних незамінованих клітинках. Кожен гравець на кожному своєму ході може рухати будь-яку фішку, але лише одну. Перестрибувати іншу фішку (чи кілька інших фішок) можна. Якщо різні фішки потрапляють в одну клітинку, вони негайно щезають.

Напишіть програму, яка визначить, хто ви́грає при правильній грі обох гравців, а якщо ви́грає 1-й гравець, то також знайде сукупність його «виграшних ходів» (після яких він все ще не втрачає свій виграш при правильній грі обох гравців).

Вхідні дані. Перший рядок містить два цілі числá N та M, розділені одним пробілом — спочатку кількість рядків, потім кількість стовпчиків. Обидва ці значення у межах від 1 до 123. Далі йдуть N рядків, що задають мінне поле. Кожен з них містить рівно по M символів `.` (позначає вільну клітинку) та/або `*` (позначає заміновану клітинку). Ці символи йдуть без роздільників, і кожен з цих N рядків містить лише ці символи та переведення рядка наприкінці.

Далі окремим рядком записане ціле число k ($1 \leqslant k \leqslant 12$) — початкова кількість фішок. Далі йдуть ще k рядків, кожен з яких містить по два цілі числá, розділені одним пробілом — спочатку номер рядка, потім номер стовпчика початкового розміщення чергової фішки, причому рядки нумеруються від 1 до N згори донизу, стовпчики — від 1 до M зліва направо. Гарантовано, що початкові розміщення вісх фішок різні, й кожна фішка розміщена в незамінованій клітинці.

Результати. Перший рядок має містити єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо дру́гий). Якщо відповідь з першого рядка `2`, то на цьому виведення слід припинити. А якщо відповідь з першого рядка `1`, то далі треба вивести також перелік всіх можливих перших ходів першого гравця, після яких дру́гий (при правильній грі першого) вже ніяк не зможе виграти. Цей перелік виводити в такому форматі: в один рядок через одинарні пробіли номер фішки, напрям (єдина буква “`D`”/“`R`” без лапок) та кількість клітинок, на які слід перемістити фішку. Порядок виведення «виграшних ходів» може бути довільним, але вони мусять бути згадані всі, кожен по одному разу. Номер фішки слід задавати числом від 1 до k, в порядку, як вони задані у вхідних даних.

Приклади:

Вхід	Рез-ти
2 4 .... .**. 2 1 1 2 4	1 1 D 1 1 R 2
1 1 . 1 1 1	2

Вхід	Рез-ти
5 7 ....*.. .*..... ....*. ..*.... 4 1 1 2 5 3 3 4 2	1 1 D 3 2 R 2 3 R 1

Розбір задачі. Оскільки щоразу можна ходити лише однією фішкою, маємо якраз-таки суму ігор, де кожна підгра являє собою звичайну «Фішку на мінному полі». Тобто, можна рахувати SG окремих ігор згідно описаного тут (див. також цей граф), побітово ксорити ці SG, й так отримувати SG суми ігор. Відповідно, позиція програшна $\Leftrightarrow$ ксор $= 0$, а «виграшними ходами» є ті й тільки ті, які роблять нулем цей побітовий ксор SG клітинок (тих, де розміщені фішки).

Правило ж знищення фішок, що потрапили в одну клітинку, в поточній постановці задачі взагалі ні на що не впливає, бо:

1. оскільки спочатку всі фішки в різних клітинках, і щокроку рухається лише одна фішка, то в одній клітинці не можуть зібратися відразу 3 чи більше фішок: або вони взагалі не збираються в одній клітинці, або їх збирається рівно дві, й інших варіантів нема;
2. коли дві фішки в одній клітинці, їхні SG однакові, й нема різниці між значеннями виразу, де двічі доксорили однакове значення, і виразом, де цього не робили: $a \oplus x \oplus x = a \oplus 0 = a$.

(Якби потрібно було писати програму, що інтерактивно грає — ігнорування цього правила впливало б, наприклад, тим, що «можна було б помилково походити вже неіснуючою фішкою» чи «можна було б не помітити, що гра скінчилася».)

Чи можна розв'язати «Кілька фішок на мінному полі» самóми лише виграшними/програшними позиціями, без SG? Теоретично, ігноруючи ефективність — так. Практично — нереально. Згадаймо пункт «Навіщо тут нові засоби?» розд. 1.3.1. Там пропонувалося розв'язати Нім через багатовимірний масив позицій розміру $(m_1+1) \times (m_2+1) \times \ldots \times (m_k+1)$, й це було відкинуто через жахливу неефективність, але так і лишилося єдиним способом розв'язати Нім без ксорів. Відповідно, для аналізу «Кількох фішок на мінному полі» самóми виграшними/програшними позиціями без SG можна запропонувати хіба «вважати позиціями поєднання координат всіх k фішок», що потребу́є $O\big((M \cdot N)^k\big)$ пам'яті, незрівня́нно більше теперішніх $\Theta(M \cdot N)$.

(Це можна оптимізувати, враховуючи ідеї «з цих $(M \cdot N)^k$ поєднань задіяна лише мала́ частина, й можна пам'ятати в `Dictionary` лише використані», «можна не розрізняти позиції, де фішки лише обміняні місцями», тощо. Але все це *дуже* складно&громіздко, а кількість позицій (отже, й обсяг перебору) при цьому скорочується далеко не так стрімко, як зросла від окремого врахування координат усіх фішок.)

Це і є причиною, чому числа Шпрага–Ґранді, де вони застосовні й використовуються повноцінно, з ксором SG(підігор), практично нереально замінити виграшними/програшними позиціями. Наступний розд. 1.4.5 описує купу інших прикладів, де замінити SG ще важче (як і за́раз, суто теоретично можна, а практично збільшення витрат від такої заміни робить її нереальною).

Водночас, все це *не* означає, ніби виграшні/програшні позиції варто забути й залишити тільки числа Шпрага–Ґранді. Кожен з розд. 1.2.3, 1.2.4, 1.4.6, 1.4.7, 1.5, 1.6 (і це ще не повний перелік) є прикладом ситуації, коли застосувати Шпрага–Ґранді неможливо, а виграшні/програшні позиції помічні.

1.4.5 Ігри, де дозволяється перетворювати гру в суму ігор вже у процесі гри, та SG для них

Гра Ґранді (початок). Далі буде багато ігор, де дозволяється перетворювати гру в суму ігор вже у процесі гри, але для початку розглянемо *гру Ґранді* [23]. Правила цієї гри такі. Спочатку є одна купка з N паличок. Гравців двоє. Гравці можуть робити однакові ходи (гра неупереджена). Кожен хід являє собою розбиття (розділення, розщеплення, …) купки на дві, причому обов'язково непорожні й обов'язково різних розмірів. Зокрема:

- купки розмірів 1 та 2 розбити не можна;
- купку розміру 3 можна розбити лише як 2+1;[5]
- купку розміру 4 можна теж лише одним способом, як 3+1;
- купку розміру 5 можна розбити вже двома способами, як 4+1 та як 3+2;
- купку розміру 6 можна розбити теж двома способами, але іншими, як 5+1 та як 4+2;
- купку розміру 7 можна розбити вже трьома способами, як 6+1, 5+2 та 4+3;

і так далі.

На подальших ходах, можна знов бра́ти купку (одну, скільки б їх наразі не було), й розбивати її за тими ж правилами. Наприклад, із 6+1 можна отримати або 5+1+1, або 4+2+1 (в обох випадках розбивши 6). Наприклад, із 5+2 можна отримати або 3+2+2, або 4+2+1 (в обох випадках розбивши 5). А, наприклад, із 4+3 можна отримати або 3+3+1, або 4+2+1 (у першому випадку єдиним дозволеним способом розбивши 4, у дру́гому 3).

Як скрізь у розд. 1.4, гра має нормальну умову завершення (хто не має ходу, програє).

[5] У цій грі 1+2 не згадують як розбиття (вважаючи його рівним 2+1) і пишуть кількості в порядку незростання

Можливі питання щодо гри класичні для послідовних ігор: або, маючи початкову чи проміжну позицію, сказати, хто ви́грає при правильній грі обох гравців, або зіграти інтерактивно, або знайти всі «виграшні ходи» з деякої проміжної позиції.

Як бачимо, хоч і заборонено розбивати купку на дві рівні, на подальших ходах рівні купки утворюватися можуть. Власне, якби заборона утворювати такі ж купки, як вже наявні, була — неясно, наскільки це ускладнило би гру (чи, раптом, спростило?), але очевидно, що якби заборона була, то гра перетворювалась би не у суму (згідно розд. 1.4.4) менших примірників такої ж гри, а у щось зовсім інше. Бо для суми ігор важливо, що хід у грі-сумі є ходом в одній із підігор (ігор-«доданків»), і при цьому різні підігри *не* впливають одна на іншу.

Тут уперше чи то не вдається, чи то незручно *спочатку* шукати SG окремих ігор, а *потім* застосовувати до них побітовий ксор.

Геть не проблема позначити ситуацію «є купка з 1 паличкою» як «позиція 1», ситуацію «є купка з 2 паличками» як «позиція 2», і сказати, що раз із жодної з них нема ходів, то $SG(1)=SG(2)=0$.

Але що робити далі, коли ходи є? Навіть щоб знайти $SG(3)$, згідно з формулою (7), потрібно порахувати SG позиції «дві купки, в одній 1 паличка, в іншій 2 палички». А такої позиції нема, є лише окремо позиція «одна купка, в ній 1 паличка» та окремо позиція «одна купка, в ній 2 палички». Можна, звісно, заявити «так із тої позиції з двома купками $(2,1)$ теж нема ходів; значить, $SG(2,1)=0$; значить, $SG(3)=\mathrm{mex}(\{SG(2,1)\})=\mathrm{mex}(\{0\})=1$»… І це навіть правда… Але це лише тимчасово відтягує питання, не вирішуючи його по суті. Такими засобами не вирішиш, як рахувати, наприклад, $SG(7)$, де є і кілька варіантів першого ходу, і після кожного з цих варіантів можливі продовження гри.

На жаль, тут мені знову не вдається розписати все так, щоб відповідь з'явилася цілком природньо й очевидно. Втім, тепер ця відповідь виявляється вже більш-менш очікуваною з попередніх викладок.

Теорема Шпрага–Ґранді (без доведення). ***Якщо гра являє собою суму кількох ігор, і поточні позиції в підіграх (іграх-«доданках») являють собою s_1, s_2, …, s_k, то число Шпрага–Ґранді всієї гри-суми дорівнює $SG(s_1)\oplus SG(s_2)\oplus\ldots\oplus SG(s_k)$:***

$$SG(s_1,s_2,\ldots,s_k)=SG(s_1)\oplus SG(s_2)\oplus\ldots\oplus SG(s_k). \tag{9}$$

Дехто помилково сприймає щойно наведене твердження за означення. Це не так, і недаремно воно називається *теоремою* Шпрага–Ґранді. Його можна і правильно (з точки зору математичної обґрунтованості) *доводити*, спираючись на *раніше* зроблені означення та допоміжні твердження.

(Означення мало би вперше запроваджувати якесь нове поняття. Тут це не так: і поняття SG, і поняття суми ігор вже означені раніше. Хоч ми раніше й не шукали SG(суми ігор), результат повинен дорівнювати тому, якби застосовували (7) до всіх-всіх варіантів зробити хід у сумі ігор (отже — всіх варіантів усіх підігор). Те, що замість цього можна порахувати SG кожної підгри окремо й узяти ксор (і вийде *такий самий (!)* результат) — треба доводити, в цьому й полягає теорема.)

Ще можна помилково подумати, ніби міркування з розд. 1.3 та 1.4 цілком доводять цю теорему. На жаль, це теж не так, хоч рівно така ж формула $SG(s_1)\oplus SG(s_2)\oplus\ldots\oplus SG(s_k)$ вже з'являлася раніше. Головним чином тому, що тоді досліджувалося *лише* те, що рівність цього побітового ксора нулю означає програшність, а не нулю — виграшність. Відразу після формули (8) навіть була примітка про те, що дотримання умов «програшність відповідає 0» та «виграшність відповідає не-нулю» ще не гарантує, чи можна такі числа підставляти у формули (6)–(7) і отримувати узгоджені результати, які не почнуть самі собі протирічити.

Враховуючи, що ця дисципліна читається студентам інженерної, а не «чисто-математичної» галузі знань, а цей розд. 1.4 і так переобтяжений доведеннями, пропоную або просто повірити у правильність цієї теореми, або вивчити не розглянуті тут складові її доведення за додатковими джерелами інформації (хоч би й [21]). І в наступному пункті почнемо користуватися цією теоремою. (Чого, з точки зору чистих математиків, не слід робити, не перевіривши доведення. Але ми будемо. Бо галузь знань інженерна, а не математична.)

Незалежно від того, чи бажаєте вивчати пропущене доведення, чи просто користуватися ним як твердженням, відповідальність за яке несуть інші, слід чітко зрозуміти, що це в аргументах mex згідно

формул (6)–(7) можна було відкидати повтори, у стилі $\mathrm{mex}(\{0,1,1,2,2,2,5\}) = \mathrm{mex}(\{0,1,2,5\}) = 3$. Однак, при визначенні SG(суми ігор) використовується побітовий ксор, який має інші, чим mex, властивості; якщо сума ігор містить дві (чи чотири, чи шість, …) однакових ігор — найнадійніше (але й найдовше) рахувати всі. В деяких випадках (зокрема, коли потрібно лише визначити виграшність, але не потрібно ні грати до кінця, ні знаходити всі ходи), можна (враховуючи $a \oplus a = 0$, $x \oplus 0 = x$) викреслювати парні кількості однакових ігор. Але абсолютно ніколи не можна перетворювати парну кількість однакових ігор в одну таку гру. Хоч би тому, що $2 \oplus 3 \oplus 2 \oplus 3 = 0 \neq 1 = 2 \oplus 3$.

Гра Ґранді (продовження). Продовжимо знаходження чисел Шпрага–Ґранді для гри Ґранді, тепер користуючись теоремою Шпрага–Ґранді. $SG(1) = SG(2) = 0$ вже визначені й пояснені, щодо них ні претензій, ні намірів щось пояснити якось інакше нема. Бо са́ме в цих позиціях все'дно неможливі розбиття (розщеплення) купок.

$SG(3) = 1$ ми раніше ніби вже визначили, але якось довго і криво. Знайдемо ще раз, тепер користуючись теоремою:

$SG(3) =$	єдиний хід: 3→2+1; ф-ла (7)
$= \mathrm{mex}(\{SG(2,1)\}) =$	ф-ла (9)
$= \mathrm{mex}(\{SG(2) \oplus SG(1)\}) =$	знайдено раніше $SG(1)=SG(2)=0$
$= \mathrm{mex}(\{0 \oplus 0\}) =$	$0 \oplus 0 = 0$
$= \mathrm{mex}(\{0\}) = 1.$	ф-ла (6)

Як бачимо, формула (9) (яка і є «серцевиною» теореми Шпрага–Ґранді) дозволила знайти те саме легше й швидше. Але стара добра ациклічність (*завдяки* якій завжди, коли для обчислення SG більшої позиції потрібні SG менших позицій, вони вже готові), теж важлива.

Тепер знайдемо $SG(4)$:

$SG(4) =$

єдиний хід: 4→3+1; ф-ла (7)

$= \mathrm{mex}(\{SG(3,1)\}) =$

ф-ла (9)

$= \mathrm{mex}(\{SG(3) \oplus SG(1)\}) =$

знайдено раніше $SG(1)=0$, $SG(3)=1$

$= \mathrm{mex}(\{1 \oplus 0\}) =$

$1 \oplus 0 = 1$

$= \mathrm{mex}(\{1\}) =$

ф-ла (6)

$= 0.$

Тепер знайдемо $SG(5)$:

$SG(5) =$

два можливі ходи 5→4+1 і 5→3+2; ф-ла (7)

$= \mathrm{mex}(\{SG(4,1),\ SG(3,2)\}) =$

ф-ла (9)

$= \mathrm{mex}(\{SG(4) \oplus SG(1),\ SG(3) \oplus SG(2)\}) =$

знайдено раніше $SG(1)=SG(2)=SG(4)=0$, $SG(3)=1$

$= \mathrm{mex}(\{0 \oplus 0,\ 1 \oplus 0\}) =$

$0 \oplus 0 = 0$; $1 \oplus 0 = 1$

$= \mathrm{mex}(\{0,1\}) =$

ф-ла (6)

$= 2.$

Тепер знайдемо $SG(6)$:

$SG(6) =$

два можливі ходи 6→5+1 і 6→4+2; ф-ла (7)

$= \text{mex}(\{SG(5,1),\ SG(4,2)\}) =$

ф-ла (9)

$= \text{mex}(\{SG(5)\oplus SG(1),\ SG(4)\oplus SG(2)\})=$

знайдено раніше
$SG(1)=SG(2)=SG(4)=0$,
$SG(3)=1$, $SG(5)=2$

$= \text{mex}(\{2\oplus 0,\ 0\oplus 0\}) =$

$2\oplus 0 = 2$; $0\oplus 0 = 0$

$= \text{mex}(\{2,0\}) =$

ф-ла (6)

$= 1.$

Тепер знайдемо $SG(7)$:

$SG(7) =$

три можливі ходи 7→6+1, 7→5+2
і 7→4+3; ф-ла (7)

$= \text{mex}(\{SG(6,1),\ SG(5,2),\ SG(4,3)\}) =$

ф-ла (9)

$= \text{mex}(\{SG(6)\oplus SG(1),\ SG(5)\oplus SG(2),\ SG(4)\oplus SG(3)\})=$

знайдено раніше
$SG(1)=SG(2)=SG(4)=0$,
$SG(3)=SG(6)=1$, $SG(5)=2$

$= \text{mex}(\{1\oplus 0,\ 2\oplus 0,\ 0\oplus 1\}) =$

$1\oplus 0 = 0\oplus 1 = 1$
$2\oplus 0 = 2$

$= \text{mex}(\{1,2,1\}) =$

ф-ла (6)

$= 0.$

Тобто, наразі знайдено такі значення SG:

i	1	2	3	4	5	6	7
$SG(i)$	0	0	1	0	2	1	0

(10)

Як бачимо, подальше продовження потребує суто технічної роботи, без нових ідей (якщо буде треба писати на основі цих міркувань програму, все потрібне вже розглянуто), а знайдений проміжок $SG(n)$ при $1 \leqslant n \leqslant 7$ відповідає вказаному в [23] (що свідчить на користь того, що тут усе правильно).

Завдяки теоремі Шпрага–Ґранді (формулі (9)), не варто зберігати чи будувати за якимись складни́ми залежностями які б не було оцінки для наборів з багатьох купок. Наприклад, для дослідження позиції, що являє собою розбиття 7+6+6+5+3+2+2, досить звернутися таблиці (10), перетворити розмір кожної купки у відповідне SG і взяти від них побітовий ксор: $SG(7,6,6,5,3,2,2) = SG(7)\oplus SG(6)\oplus SG(6)\oplus SG(5)\oplus SG(3)\oplus SG(2)\oplus SG(2) = 0\oplus 1\oplus 1\oplus 2\oplus 1\oplus 0\oplus 0 = 2\oplus 1 = 3$; якщо потрібна лише (ви/про)грашність, то після цього досить сказати, що $3 > 0$, тож згідно (8) така позиція виграшна. (Приклад, як відновлювати ходи, починається буквально через кілька абзаців.)

Було б неправильно зберігати замість чисел Шпрага–Ґранді лише позначки (ви/про)грашності, й робити ксор із ними. Хоч будь-яке SG, строго більше 0, й означає виграшну позицію, але, наприклад, $2\oplus 1 = 3 \neq 0 = 2\oplus 2$, але *обидві* ці ситуації підходять під *один* опис «гра розбита в суму двох ігор, в обох підіграх позиції виграшні». Тобто, ***числові значення SG несуть у собі більше інформації, чим сама лише (ви/про)грашність, хоч і важко пояснити словами, що то за інформація.*** (В усякому разі, позиція, $SG(\text{якої}) = 3$, не є «утричі виграшнішою» за позицію, $SG(\text{якої}) = 1\ldots$)

Приклад нетривіального відновлення ходів через відомі SG для гри Ґранді. Продовжимо розгляд того ж прикладу «поточна позиція є розбиттям 7+6+6+5+3+2+2», і знайдемо для самóї цієї позиції всі можливі «виграшні ходи», потім застосуємо один з них, і доведемо (ну, майже доведемо…) гру до кінця. Переможного кінця, бо раніше вже з'ясовано, що $SG(7,6,6,5,3,2,2)=3\neq 0$.

Отже, нам треба знайти всі такі ходи, щоб замість SG=3 стало SG=0. Значить, для кожної підгри треба подивитися, чи можна в ній походити так, щоб зміна SG усіх разом узятих підігор стала нулем, тобто спробувати, чи можна…

- з позиції «7 паличок», де $SG(7)=0$, перейти до позиції, SG якої буде дорівнювати $0\oplus 3=3$;
- з позиції «6 паличок», де $SG(6)=1$, перейти до позиції, SG якої буде дорівнювати $1\oplus 3=2$;
- з позиції «5 паличок», де $SG(5)=2$, перейти до позиції, SG якої буде дорівнювати $2\oplus 3=1$;
- з позиції «3 палички», де $SG(3)=1$, перейти до позиції, SG якої буде дорівнювати $1\oplus 3=2$;
- з позиції «2 палички», де $SG(2)=0$, перейти до позиції, SG якої буде дорівнювати $0\oplus 3=3$

(повтори такого вигляду, як «є по дві, а не по одній, підгри з 6-ма та з 2-ма паличками», я ігнорую, бо для обох підігор «6 паличок» між собою та для обох підігор «2 палички» між собою все однаковісіньке; однак, $SG(6)=SG(3)=1$ не означає такої однаковості, бо там однакові SG, але ж сукупності ходів різні; як побачимо, сáме в цьому випадку ця різниця проявиться досить істотно).

От і розглянемо кожен з цих варіантів. Чи можна з позиції «7 паличок», де $SG(7)=0$, перейти до позиції, SG(якої) $=0\oplus 3=3$? Ні, бо при обчисленні SG(7), останньою дією було $\mathrm{mex}(\{1,2,1\})=0$, тобто серед варіантів, куди є хід, не було позиції, SG(якої)=3.

Чи можна з позиції «6 паличок», де $SG(6)=1$, перейти до позиції, SG(якої) $=1\oplus 3=2$? Так, бо при обчисленні SG(6) останньою дією було $\mathrm{mex}(\{2,0\})=1$, тобто серед варіантів, куди є хід, була позиція, SG(якої)=2, й треба подивитися, яка це була позиція. Подивившись у все той же процес обчислення SG(6) ще трохи детальніше, бачимо $2=2\oplus 0=SG(5)\oplus SG(1)$.

(Тут з процесу обчислень вибрано лише потрібне й записано у зворотньому відносно обчислень порядку, який зáраз якрáз зручний.)

Тобто, можна «розділити одну з купок з 6 паличок на 5+1», й це буде «виграшний хід», який віддає суперникові програшну позицію. Переконаймося в цьому: $SG(7,6,5,5,3,2,2,1)=SG(7)\oplus\oplus SG(6)\oplus SG(5)\oplus SG(5)\oplus SG(3)\oplus SG(2)\oplus SG(2)\oplus SG(1)=0\oplus 1\oplus 2\oplus 2\oplus 1\oplus 0\oplus 0\oplus 0=2\oplus\oplus 2\oplus 1\oplus 1\oplus =0\oplus 0=0$. Причому, якби міркували «як для Німа», то цього способу не знайшли б, бо $1\oplus 3=2>1$ і був би висновок «такий хід неможливий, бо палички можна лише забирати, а не додавати». Але такий висновок хибний, бо зміна чисел Шпрага–Ґранді лише частково нагадує забирання паличок.

Продовжуємо.

Чи можна з позиції «5 паличок», де $SG(5)=2$, перейти до позиції, SG(якої) $=2\oplus 3=1$? Так, бо при обчисленні SG(5) останньою дією було $\mathrm{mex}(\{0,1\})=2$, тобто серед варіантів, куди є хід, була позиція, SG(якої)=1, й треба подивитися, яка це була позиція. Подивившись у той сáмий процес обчислення SG(5) ще детальніше, бачимо $1=1\oplus 0=SG(3)\oplus SG(2)$; значить, слід розбити 5 як 3+2.

В цьому випадку (на відміну від попереднього) $2\oplus 3=1<2$, тому міркування «як для Німа» теж скажуть, що тут є «виграшний хід». Але не дуже допоможуть його знайти: вкажуть, що потрібен хід у ту позицію, SG(якої)=1, що такий хід точно-точно існує, але щоб з'ясувати, що це хід «розбити 5 як 3+2» все'дно або треба було запам'ятати це, знаходячи SG(5), або тепер треба шукати це заново.

(На мою думку (це лише думка, а не абсолютна істина):

- якщо потрібно знайти *всі* ходи́, то нема смислу робити «німівську перевірку» $(m_i\oplus r)<r$, бо і коли вона істинна, то хід існує (але треба ще шукати, який), і коли хибна, то може існувати (й тому теж треба перевіряти й шукати); і що тоді дає така перевірка?
- якщо досить знайти *якийсь один* із ходів (причому, байдуже який), то «німівська перевірка» (легка й швидка) корисна тим, що підігри, де $(m_i\oplus r)<r$, «перспективніші» інших: у них «виграшний хід» гарантовано є, й можна шукати лише там, не витрачаючи час на інші підігри, де може й не бути жодного «виграшного ходу».

Звісно, «німівська перевірка» є «легкою і швидкою» лише коли є готовий масив SG для всіх потрібних розімів купок. Але ж ми виходимо з того, що він є; коли його нема, то взагалі неясно, як грати в цю гру осмислено й стратегічно…)

Усе ще продовжуємо.

Чи можна з позиції «3 палички», де $SG(3)=1$, перейти до позиції, SG(якої) $=1\oplus 3=2$? Ні, бо при обчисленні SG(3) останньою дією було mex$(\{0\})=0$, тобто серед варіантів, куди є хід, не було позиції, SG(якої)=2.

Чи можна з позиції «2 палички», де $SG(3)=1$, перейти до позиції, SG(якої) $=0\oplus 3=3$? Ні, бо коли з позиції «2 палички» нема взагалі ніяких ходів, то нема й ходів до позиції, SG(якої)=3.

Підсумуємо всі можливі варіанти «виграшного ходу» з позиції 7+6+6+5+3+2+2:

- розбити якусь із купок з 6 паличок як 5+1;
  (окремий сумний факт, що я навіть затрудняюся сказати, чи це два різні ходи (бо таких купок дві), чи один хід (бо вище заявлялося, що в цій грі розглядаються лише незростаючі послідовності розмірів, тож неясно, в чому відмінність між ходами, які обидва ведуть із позиції 7+6+6+5+3+2+2 у позицію 7+6+5+5+3+2+2+1); якби була задача про цю гру, де треба було б писати програму, яка має вивести всі такі ходи, це мало б бути додатково прописано в умові, а поки що просто залишмо це питання недоз'ясованим)
- розбити купку з 5 паличок як 3+2.

Оскільки раніше було обіцяно один раз зробити пошук усіх «виграшних ходів», а потім довести гру до переможного кінця, надалі щоразу вибираючи якийсь один «переможний хід», займімося цим.

Нехай був обраний варіант ходу «розбити 6 як 5+1», тобто отримали позицію 7+6+5+5+3+2+2+1.

Нехай, наприклад, суперник відповів ходом «розбити 7 як 5+2» (ми ж мусимо розглядати і якісь ходи суперника, щоб довести приклад гри до кінця). Значить, отримали позицію 6+5+5+5+3+2+2+2+1; формально це означає, що треба обчислити $SG(6,5,5,5,3,2,2,2,1)=$ $=SG(6)\oplus SG(5)\oplus SG(5)\oplus SG(5)\oplus SG(3)\oplus SG(2)\oplus SG(2)\oplus SG(2)\oplus SG(1)\ldots$ але ми знаємо, що:

1. $SG(2)=SG(1)=0$;
2. $a\oplus 0=a$ для абсолютно будь-якого a;
3. купки «2 палички» та «1 паличка» по суті не беруть участі у грі (їх вже не можна розбивати).

Тому, щоб і не забути, якими насправді є позиції, і хоч трохи скоротити записи, надалі розділятиму позиції на частину, де ще можуть відбуватися ходи, й частину, де лише двійки й одиничці; зокрема, цю позицію запишу як $(6+5+5+5+3)+(2+2+2+1)$ й шукатиму $SG(6,5,5,5,3)=$ $=SG(6)\oplus SG(5)\oplus SG(5)\oplus SG(5)\oplus SG(3)=1\oplus 2\oplus 2\oplus 2\oplus 1=(1\oplus 1)\oplus(2\oplus 2)\oplus 2=2$.

(Чи можливий і чи доречний якийсь аналогічний прийом в якійсь іншій грі (не грі Ґранді, а іншій) — невідомо, залежить від гри; але тут це доречно.)

Оскільки тепер шукаємо якийсь один «виграшний хід», можна користуватися «німівською перевіркою», отже шукати хід лише в тих підіграх, SG(якої) $\oplus 2<$ SG(якої). Са́ме для щойно згаданих розмірів купок і значень SG це означає лише SG=2 (бо $1\oplus 2=3>1$, і лише $2\oplus 2=0<2$) й лише в якійсь із купок «5 паличок» (бо серед наявних у позиції купок, лише вони дають SG=2). А єдиним ходом у купці «5 паличок», який дає SG, рівний $SG(5)\oplus 2=2\oplus 2=0$, є «розбити 5 як 4+1».

(Са́ме так. Минулого кроку пропонувалося «розбити 5 як 3+2» (замість цього був узятий інший хід «розбити 6 як 5+1», але ж варіант «розбити 5 як 3+2» теж пропонувався як «виграшний»), а тепер таке ж 5 слід розбивати вже як 4+1. При тому, що розбиття 3+2 і 4+1 ні в якому разі *не(!)* взаємозамінні. Тому що хоч і маємо ту саму глобальну мету «виграти», яка однаково реалізується через однакову трохи локальнішу мету «підсунути супернику програшну позицію, SG(якої)=0», але, з причини різних SG глобально всієї позиції-з-багатьох-купок, для приведення до нуля SG всієї гри (сукупності підігор) конкретно у підгрі «купка з 5 паличок» тоді була зовсім локальна мета перейти до SG=1, а тепер є зовсім локальна мета перейти до SG=0. А раз різна зовсім локальна мета, то різні й розбиття купки з 5 паличок.)

Якщо повернутися від підгри до всієї гри, це означає перехід до позиції $(6+5+5+4+3)+$ $+(2+2+2+1+1)$.

Нехай, наприклад, суперник відповів таким са́мим ходом «розбити 5 як 4+1» (має право, бо інші купки «5 паличок» ще є).

Значить, отримали позицію $(6+5+4+4+3)+(2+2+2+1+1+1)$; для неї $SG(6,5,4,4,3)=SG(6)\oplus SG(5)\oplus SG(4)\oplus SG(4)\oplus SG(3)=1\oplus 2\oplus 0\oplus 0\oplus 1=1\oplus 1\oplus 2=$ $=2$. Знову користуючись «німівською перевіркою», знов отримуємо, що варто походити у підгрі,

SG(якої)=2, тобто знову «розбити 5 як 4+1» й перейти до позиції всієї гри $(6+4+4+4+3)\ +\ +\ (2+2+2+1+1+1+1)$.

Нехай цього разу суперник вирішив, наприклад, «розбити 4 як 3+1».

Отримали позицію $(6+4+4+3+3)\ +\ (2+2+2+1+1+1+1+1)$; для неї $SG(6,4,4,3,3) = SG(6)\oplus SG(4)\oplus SG(4)\oplus SG(3)\oplus SG(3) = 1\oplus 0\oplus 0\oplus 1\oplus 1 = 1\oplus 1\oplus 1 = =1$. Взагалі кажучи, тут є різні «виграшні ходи» (хоча б тому, що $1\oplus 1=0<1$, і є різні купки (6, 3, 3), де SG=1), але обмежимося першим з варіантів (купкою «6 паличок»). Потрібно перейти в цій підгрі від SG=1 до SG=0. Аналізуючи процес обчислення $SG(6)=1$ (звідки взявся результат 1?), бачимо $1=\mathrm{mex}(\{2,0\})$, причому варіант, який має SG=0, відповідає ходу «розбити 6 як 4+2». Отже, са́ме його (треба/можна) використати, й тим перейти до позиції $(4+4+4+3+3)\ +\ +\ (2+2+2+2+1+1+1+1+1)$.

(То все-таки «треба» чи все-таки «можна»? Якщо говорити про розбиття купки «6 паличок» — треба, це єдиний для цієї купки «виграшний хід», який перетворює SG до потрібного значення 0; але є інші «виграшні ходи» в купках «3 палички», тож у цьому смислі він лише один з кількох можливих.

Також зауважу (дуже схоже зауваження вже було раніше, але повторюся, бо це важливо), що щойно зроблено хід «розбити 6 як 4+2» (й це єдиний «виграшний» спосіб розбити 6), а набагато раніше було зроблено хід «розбити 6 як 5+1» (й тоді то був єдиний «виграшний» спосіб розбити 6). Й це нормально, бо з точки зору SG всієї гри (суми всіх підігор) за́раз треба переходити від підгри «купка “6 паличок”» до підгри, SG(якої)=0, а тоді треба було переходити до підгри, SG(якої)=2. Щоб SG *під*гри стало 2, й са́ме це й забезпечить, щоб SG всієї гри стало 0.)

Щоб довести гру до кінця, слід було би далі продовжувати аналогічні обчислення SG чергових поточних сум підігор. Якби йшлося про комп’ютерну програму, це був би найдоречніший спосіб. Але оскільки це текст для людей, а не виконання чи написання програми, заміню це продовження (яке було б довгуватим, ще нуднішим, і при цьому не містило би нічого ідейно нового) на таке міркування. Далі можливі *тільки* ходи «розбити 4 як 3+1» та «розбити 3 як 2+1». Причому, кожен з цих ходів — єдиний можливий для купки з відповідної кількості паличок. Тому, можна сказати, що безальтернативно лишається три ходи «розбити 3 як 2+1» та два ходи «розбити 4 як 3+1» (кожен з яких породжує купку «3 палички», а отже й кожен по ще одному ходу «розбити 3 як 2+1»). Отже, сумарно лишається $3+2+2=7$ ходів (у термінах розд. 1.6.4, півходів, тобто ходів однієї зі сторін), тобто непарна кількість, тобто останнім ходитиме той, хто має ходити за́раз, тобто той, для кого ми знаходили попередні ходи. Отже, він і виграє.

Задача 1.4:D. «Викреслювання клітинок — 1»

Є стрічка шириною в одну клітинку і довжиною в N клітинок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може закреслити кілька клітинок. Якщо закреслення відбувається скраю смужки, або безпосередньо поруч з уже закресленою клітинкою, то закреслити можна або 1, або 2 клітинки підряд. Якщо закреслення відбувається не згідно попереднього речення, а десь всере́дині досі суцільного фрагмента стрічки (так, щоб по обидва боки від закресленого були незакреслені фрагменти), то закреслити можна або 2, або 4 клітинки підряд. Ніяких інших варіантів ходу нема. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто закреслює останню клітинку (можливо, разом із ще кількома).

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців (хто може забезпечити собі виграш, хоч би як не грав інший). Якщо виграє 1-й, то програма повинна знайти також сукупність усіх його виграшних перших ходів.

Пару прикладів суто для пояснення правил гри наведено внизу стор. 57.

Вхідні дані. Єдине ціле число N $(1\leqslant N\leqslant 2000)$ — початкова кількість клітинок у стрічці (спочатку — єдиному неперервному фрагменті).

Результати. Єдине ціле число, або `1` (якщо 1-й гравець може забезпечити собі виграш), або `2` (якщо 2-й). Якщо відповідь з першого рядка `2`, то на цьому виведення слід припинити. А якщо відповідь з першого рядка `1`, то далі треба вивести також перелік всіх можливих перших ходів 1-го гравця, після яких 2-й (при правильній грі 1-го) вже ніяк не зможе виграти. Цей перелік виводити в такому форматі: кожен такий хід в окремому рядку; кожен хід записується як пара чисел через

пропуск: номер початкової клітинки закреслення та кількість клітинок, що закреслюються; якщо є багато різних виграшних перших ходів, вони повинні бути відсортовані за зростанням номера початкової клітинки, а якщо є багато різних виграшних перших ходів, де закреслення починаються в одній клітинці, то за зростанням кількості клітинок, що закреслюються.

Кількість перших виграшних ходів виводити не треба. Вважати, що клітинки занумеровані від 1 до N.

Приклади:

Вхід	Рез-ти
2	1 1 2
3	2
4	1 1 1 2 2 4 1

Вхід	Рез-ти
17	1 6 4 7 2 9 4 10 2

Примітки. У першому прикладі ($N=2$), виграшним є лише один хід: закреслити зразу обидві клітинки й виграти за один півхід.

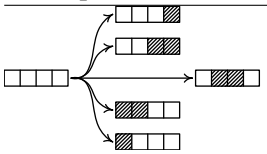
У дру́гому прикладі ($N=3$), виграшних ходів нема взагалі, бо виграш може забезпечити 2-й гравець. 1-й гравець не має іншого вибору, крім як креслити чи то одну, чи то дві клітинки біля одного з країв, і в будь-якому з цих випадків 2-му гравцеві дістанеться один неперервний фрагмент або з однієї клітинки, або з двох; в будь-якому разі, його можна увесь закреслити й виграти.

У третьому прикладі ($N=4$), виграшними є формально три різні ходи, два з яких (`1 1` та `4 1`) симетричні: закреслити якусь одну з крайніх клітинок, після чого супернику дістанеться один неперервний фрагмент із трьох клітинок, що призведе до програшу 2-го гравця згідно міркувань попереднього абзацу. Інший виграшний перший хід — закреслити дві клітинки рівно посере́дині, тоді від стрічки лишається два окремих квадратики 1×1, 2-й не має іншого вибору, крім як закреслити один з них, після чого 1-й закреслює останній з цих квадратиків і виграє. Зверніть увагу, що у відповіді ходи *мусять* бути в порядку `1 1`, `2 2`, `4 1`.

Розбір задачі. Спочатку розглянемо процес побудови SG для цієї задачі, а вже потім розглянемо деякі інші питання, теж важливі та/або специфічні для цієї задачі.

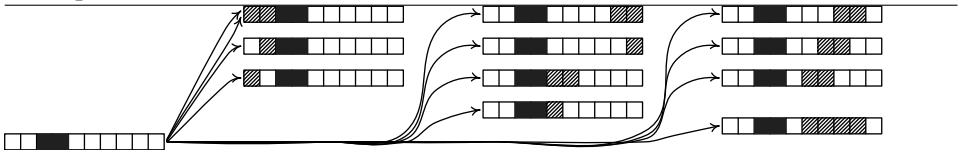
В цій грі є як ходи, що викреслюють клітинки скраю стрічки, так і ходи, які викреслюють клітинки всере́дині стрічки. У першому випадку зручніше вважати, що перехід між позиціями виражається самими лише формулами (6)–(7), без застосування суми ігор; у дру́гому, необхідно вважати, що відбувається розбиття поточної гри (стрічки) на дві непорожні підгри, тож використовується спочатку формула (9), результати якої потім теж підставляються у формули (6)–(7).

Приклади можливих ходів для позиції «суцільна стрічка з чотирьох клітинок»:



З кожного з країв можна закреслити 1 або 2 клітинки; посередині можна закреслити 2 клітинки, але не 4, бо «всере́дині» вимагає, щоб з кожного з країв хоч щось лишилося.

Приклади можливих ходів для позиції «10 клітинок, 3-тя та 4-та вже закреслені раніше»:



(Суцільно-чорні — закреслені раніше; заштриховані — закреслені на поточному ході.) У лівому фрагменті можна закреслити будь-яку одну крайню, або двома різними способами (дві крайні зліва чи дві крайні справа) пояснити один і той самий хід «закреслити обидві». Якщо креслити з країв правого фрагмента, виходить чотири варіанти: закреслити одну зліва, або дві зліва, або одну справа, або дві справа. Якщо креслити всере́дині правого фрагмента, є один варіант закреслити зразу чотири, і три варіанти закреслити дві.

Ходи, що викреслюють 1 чи 2 клітинки скраю стрічки, переводять її з «позиції s клітинок» до «позиції $(s-1)$ клітинок» чи «позиції $(s-2)$ клітинок» відповідно. Тому, при обчисленні $SG(s)$ у правій частині (7) будуть, зокрема, $SG(s-1)$ («практично завжди» при $s \geqslant 1$) та $SG(s-2)$ (лише при $s \geqslant 2$).

Якщо $s \geqslant 4$, то додатково з'являється можливість викреслити «десь всере́дині» 2 клітинки, тим розбивши одну стрічку на дві, тобто перейти до суми ігор. При досить великих s можуть бути різні розміри шматків. (Наприклад, навіть з урахуванням роз'яснених в примітці наприкінці умові симетрій, при задіюванні лише одного з симетричних варіантів, «позицію 10 клітинок» можна розбити такими способами: $1+\not2+7$; $2+\not2+6$; $3+\not2+5$; $4+\not2+4$. В усіх випадках, "$\not2$" позначає дві закреслені клітинки, які вже не є частинами позицій подальших ігор-доданків. Тобто, 10 може перетворитися в (1, 7), або (2, 6), або (3, 5), або (4, 4).)

З викреслюваннями «десь всере́дині» 4 клітинок ситуація цілком аналогічна.

Все це можна підсумувати, наприклад, так:

$$SG(s) = \operatorname{mex}\left(\left\{ \begin{array}{c} SG(s-1), \quad \underbrace{SG(s-2)}_{\text{при } s\geqslant 2}, \\ \underbrace{SG(i) \oplus SG(s-2-i)}_{\text{при } s\geqslant 4,\ \text{для всіх } 1\leqslant i<\lfloor \frac{s}{2} \rfloor}, \\ \underbrace{SG(i) \oplus SG(s-4-i)}_{\text{при } s\geqslant 6,\ \text{для всіх } 1\leqslant i<\lfloor \frac{s}{2} \rfloor -1} \end{array} \right\} \right). \tag{11}$$

А ось процеси обчислень кількох $SG(s)$:

$SG(0) = \operatorname{mex}(\varnothing) = 0$, бо коли є лише 0 клітинок, нема чого закреслювати (нема ходів).

$SG(1) = \operatorname{mex}(\{SG(0)\}) = \operatorname{mex}(\{0\}) = 1$, бо коли є лише 1 клітинка, то існує лиш один хід (закреслити її), який веде у позицію «0 клітинок»; значення $SG(0) = 0$ з'ясоване в попередньому абзаці.

Далі варіантів вибору стає більше й більше, тому в прикладах обчислення кількох подальших $SG(s)$ будемо пояснювати окремі проміжні кроки.

$SG(2) =$

Ходів, по суті, два (якщо вважати симетричні ходи однаковими): (1) викреслити обидві клітинки (заявивши, що вони «дві крайні»), залишиться 0; (2) викреслити одну крайню, залишиться 1.

$= \operatorname{mex}(\{SG(0), SG(1)\}) =$

значення $SG(0)=0$, $SG(1)=1$ вже знайдені раніше

$= \operatorname{mex}(\{0, 1\}) =$

ф-ла (6)

$= 2.$

$SG(3) =$

Ходів два, якщо вважати симетричні однаковими: (1) викреслити дві крайні клітинки, залишиться 1; (2) викреслити одну крайню, залишиться 2.

$= \operatorname{mex}(\{SG(1), SG(2)\}) =$

значення $SG(1)=1$, $SG(2)=2$ вже знайдені раніше

$= \operatorname{mex}(\{1, 2\}) =$

ф-ла (6)

$= 0.$

Якраз відповідає прикладу введення-виведення, що при $N=3$ виграє 2-й.

$SG(4) =$

Ходів три, якщо вважати симетричні однаковими:
(1) викреслити дві крайні клітинки, залишиться 2;
(2) викреслити одну крайню, залишиться 3;
(3) викреслити дві всере́дині,
залишиться два окремі шматки 1 і 1.

$= \text{mex}(\{SG(2), SG(3), SG(1,1)\}) =$

розкриваємо $SG(1,1)$ згідно (9)

$= \text{mex}(\{SG(2),\ SG(3),\ SG(1) \oplus SG(1)\}) =$

значення $SG(1)$=1, $SG(2)$=2,
$SG(3)$=0 вже знайдені раніше

$= \text{mex}(\{2, 0, 1 \oplus 1\}) =$

$1 \oplus 1 = 0$

$= \text{mex}(\{2, 0, 0\}) =$

ф-ла (6)

$= 1.$

Якраз відповідає прикладу введення-виведення,
що при N=4 є істотно різні виграшні ходи 1-го,
і в нас якраз у аргументах mex було зразу два нулі
(один з них відповідає «закреслити одну крайню»,
тому двом різним рядкам відповіді `1 1` і `4 1`»;
інший нуль відповідає ходу «викреслити дві клітинки
якраз посередині», тому лише одному рядку відповіді `2 2`).

$SG(5) =$

Ходів три, якщо вважати симетричні однаковими:
(1) викреслити дві крайні клітинки, залишиться 3;
(2) викреслити одну крайню, залишиться 4;
(3) викреслити дві всере́дині,
залишиться два окремі шматки 2 і 1.

$= \text{mex}(\{SG(3), SG(4), SG(2,1)\}) =$

розкриваємо $SG(2,1)$ згідно (9)

$= \text{mex}(\{SG(3),\ SG(4),\ SG(2) \oplus SG(1)\}) =$

значення $SG(1)$=1, $SG(2)$=2,
$SG(3)$=0, $SG(4)$=1 вже знайдені раніше

$= \text{mex}(\{0, 1, 2 \oplus 1\}) =$

$1 \oplus 1 = 0$

$= \text{mex}(\{0, 1, 3\}) =$

ф-ла (6)

$= 2.$

(Конкретні значення SG для цієї задачі не такі, як для гри Ґранді. Са́ме тому, що то були SG для гри Ґранді, а це SG для гри «Викреслювання клітинок — 1», і в цих іграх різні правила. Однак, загальна логіка вельми схожа, відрізняється *лише* те, з яких варіантів береться mex при уточненні під конкретну гру (по суті, під дозволені у грі ходи) загальної формули (7).)

Оскільки вивести всі можливі «виграшні ходи» потрібно лише для стартової позиції (де стрічка одна), в цій задачі досить лише шукати ходи, що ведуть у позиції, $SG(\text{яких}) = 0$ під час знаходженння $\text{mex}(\{SG(B_1), SG(B_2), \ldots, SG(B_q)\})$ згідно (7), а мати справу з правилом «спочатку знайшли $r = SG(s_1) \oplus SG(s_2) \oplus \ldots \oplus SG(s_k)$, потім намагаємося в окремих іграх перейти з $SG(s_i)$ до $SG(s_i) \oplus r$» са́ме в задачі 1.4:D *не* доводиться.

При цьому, формулу (7) доводиться використати багатократно (найімовірніше, знайти за нею всі-всі $SG(1)$, $SG(2)$, $SG(3)$, …, $SG(N)$), а формувати й виводити «виграшні ходи» треба лише при знаходженні останнього $SG(N)$. Це робить доцільним чи то додатковий параметр `bool doChoseMoves` (якщо оформлювати знаходження $SG(i)$ як метод `CalcSG(int i, bool doChoseMoves)`, чи то перевірку `if(i==n)`.

Також очевидно, що симетричність можливих ходів має водночас і приємні, й неприємні сто́рони. Головна приємність — ця симетрія помічна для скорочення перебору варіантів. Нам же

невигідно робити відверто зайву роботу, як-то спочатку окремо знаходити $SG(2) \oplus SG(5)$, потім окремо $SG(5) \oplus SG(2)$ — які, враховуючи асоціативність і комутативність ксора, гарантовано рівні, а за властивостями mex повтори рівних значень не змінюють його результат, але створюють зайву роботу. Головні неприємності — задача вимагає вивести всі ходи, і з точки зору формату виведення результатів симетричні ходи, розглянуті як один, зазвичай є двома окремими, але іноді все-таки одним (якщо він якраз посере́дині), й усе це треба акуратно врахувати.

Задача 1.4:E. «Викреслювання клітинок — інтерактив»

Правила гри повністю відповідають попередній задачі 1.4:D. Корисним для розуміння правил гри є також рисунок внизу стор. 57.

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Протокол взаємодії. На початку, один раз, Ваша програма повинна прочитати початкову позицію гри, яка задається єдиним числом N ($1 \leqslant N \leqslant 2000$) — кількість клітинок у єдиній неперервній смужці. Потім слід повторювати такий цикл:

1. Вивести два числá, розділені пропуском, у окремому рядку — свій хід, тобто починаючи з якої клітинки вона закреслює, і скільки клітинок, згідно з раніше описаними правилами. Клітинки занумеровані з одиниці (1, 2, …, N), й ці номери лишаються закріпленими за клітинками протягом гри (як би не викреслювали інші клітинки).
2. Якщо при цьому відбулося викреслення останньої клітинки, вивести окремим рядком фразу “`I won!`” (без лапок, символ-у-символ згідно зразку) й завершити роботу.
3. Інакше, прочитати хід програми-суперниці, в такому ж форматі (починаючи з якої клітинки вона закреслює, і скільки клітинок; ці числа подані в одному рядку, розділені одинарним пробілом). Гарантовано, що хід допустимий: відповідає описаним вище правилам, і жодна з нині закреслюваних клітинок не була закреслена раніше. Ця гарантія дійсна лише якщо Ваша програма правильно визначила, що гра ще не закінчилася.
4. Якщо при цьому відбулося викреслення останньої клітинки, вивести окремим рядком фразу “`You won...`” (без лапок, символ-у-символ згідно зразку) й завершити роботу.

Це слід повторювати, доки не будуть викреслені всі клітинки (тобто, доки якась із програм-гравців не ви́грає). Програма-суперниця не виводить фраз “`I won!`” / “`You won...`” чи якихось їх аналогів.

Приклад:

Вхід, суперник	Ваша програма
7 4 1 1 1	 2 2 6 2 5 1 I won!

Примітки. Нібито порожні рядки між різними ходами суто щоб краще було видно, хто коли ходить; вводити/виводити їх не треба.

Хід гри з прикладу можна прокоментувати так:

коментар	хід	стан стрічки
до початку гри жодна з 7 клітинок не закреслена		. . . . .
Ваша програма закреслює 2-у та 3-ю клітинки	2 2	. XX . . . .
Програма-суперниця закреслює 4-у клітинку	4 1	. XXO . . .
Ваша програма закреслює 6-у та 7-у клітинки	6 2	. XXO . XX
Програма-суперниця закреслює 1-у клітинки	1 1	OXXO . XX
Ваша програма закреслює 5-у клітинку й виграє	5 1	OXXOXXX

Оцінювання. У більшості тестів Ваша програма матиме справу з ідеальною програмою-суперницею, яка не робить помилок. Однак, невелика кількість тестів передбачатиме також і гру з різними

програмами-суперницями, які грати не вміють — роблять ходи, які дотримуються формальних правил гри, але можуть вибирати не найкращий з допустимих ходів, дотримуючись кожна своїх власних уявлень про те, як грати в цю гру. Буде оцінюватися як уміння Вашої програми виграти там, де це гарантовано можливо, так і вміння Вашої програми гідно, дотримуючись правил гри, програти, так і вміння Вашої програми скористатися (теж згідно правил) помилками чи іншими неадекватностями програми-суперниці, якщо такі будуть. Уміння перехоплювати ініціативу в неідеальної суперниці оцінюватиметься лише на досить великих тестах.

За будь-яке порушення правил гри з боку Вашої програми, відповідний тест оцінюватиметься як не пройдений.

Розбір задачі. Дещо дивно, але лиш цього (здається, шостого) разу написати інтерактивну програму-гравця стало *значно* важче, чим розв'язок неінтерактивної задачі про цю ж гру.

Головних причин цього дві.

Причина №1. Для визначення виграшності/програшності (й навіть для знаходження «виграшних» перших ходів) досить лише заповнювати згідно формули (11) масив значень SG, а для інтерактивної взаємодії потрібно ще (не замість, а *ще!*) переходити між поданнями «така-то підгра, у ній такі-то номери клітинок» і «такі-то номери клітинок за початковою нумерацією, коли був один неперервний фрагмент», причому в обох напрямках (щоб і виводити свої ходи, і розуміти, як хід суперника змінює поточну позицію та її SG).

Зберігати, де (за початковою нумерацією) зберігається який фрагмент-що-є-«доданком»-суми-ігор — ніби й не надто складно, але ж це треба зробити, причому акуратно і враховуючи, що такі фрагменти можуть і вкорочуватися з більшої довжини до меншої (коли закреслюють клітинк(у/и) скраю чи коло вже закресленої), і розділятися на два окремі (коли закреслюють клітинки «всере́дині»), і щезати, будучи викресленими повністю (лише за довжини фрагменту 1 або 2, але ж таке теж буде, й не обов'язково наприкінці), й такі дії можуть повторюватися багато разів у різних послідовностях.

(Тут можна шукати/розробляти якесь хитре поєднання структур даних, щоб робити все це якнайшвидше (я, ніби, вмію досягти оцінки «$O(\log N)$ на кожну операцію»), й це була б цікава задача зі структур даних… Але якщо мати *лише* мету розв'язати цю задачу — це досить марна робота, бо крім того все'дно потрібно і будувати масив значень SG, і визначати, коли в якій підгрі ходити; чи можна істотно прооптимізувати це (особливо, перебір варіантів згідно (11)) — складніше питання, я не вмію.

Тому, пропоную просто акуратно підтримувати `List`, елементами якого є фрагменти (кожен фрагмент — одна підгра, подається парою чисел: початком (за початковою нумерацією) й довжиною); змінювати цей `List` варто щоразу, вибравши свій хід, та щоразу, прочитавши хід суперника, але не частіше.)

Причина №2. Для визначення варіантів *першого* «виграшного ходу» досить дивитися, які аргументи mex дорівнюють нулю, а далі все ускладнюється. Як вже згадано і тут, і тут, маючи суму ігор, потрібно робити нулем не SG якоїсь окремої підгри, а ксор усіх SG.

(Наприклад, якщо стрічка вже розділена на три окремі фрагменти, й SG цих фрагментів 2, 3 і 5, то це виграшна позиція ($2 \oplus 3 \oplus 5 = 010_{Bin} \oplus 011_{Bin} \oplus 101_{Bin} = 100_{Bin} = 4 \neq 0$), і виграшний хід полягає в тому, щоб походити у підгрі, $SG(\text{якої}) = 5$, у позицію, $SG(\text{якої}) = 1$, бо тоді якраз вийде $2 \oplus 3 \oplus 1 = 0$. Не факт, чи це єдиний виграшний хід, але такий хід точно виграшний (чи то єдиний, чи то один із кількох).)

Тому, потрібно або при використанні (11) запам'ятовувати для кожного значення $SG(B_i)$, який хід (якщо їх кілька — хоча б один з них) дає са́ме таке значення SG, або на кожному ході перебирати заново варіанти згідно (11), шукаючи потрібне значення SG-після-ходу. Кожен з цих способів можна запрограмувати адекватно, щоб він добре вклався у відведені обмеження часу й пам'яті, але ж у кожному зі способів можна й напомилятися…

(Один з можливих варіантів реалізації першого способу — при переборі варіантів згідно (11) формувати `SortedDictionary`, де ключами є значення SG-після-ходу, а дру́гими значеннями пар — ходи. Це і дозволить потім швидко знаходити потрібний хід, і може бути використано для знаходження mex (не факт, чи завжди варто шукати mex са́ме так, але приємно отримати подвійну користь від однієї дії). Ще детальніших вказівок щодо реалізації не буде, бо це було б уже занадто про алгоритм, але не загальні засоби теорії ігор.)

Задача 1.4:F. «Лісові шахи»

(Автор задачі — Богдан Яковенко, історично перший (2003 р.) золотий призер Міжнародної олімпіади з інформатики (учнівської) від України. Подальший текст задачі (але не розбір) подається в точності у його формулюванні.)

Одного разу хлопець, що цікавиться олімпіадними задачами з програмування, пішов на прогулянку до лісу. Ходячи поміж деревами лісу, він зустрів Злого Мішку, лісового звіра, що не любить, коли хтось заходить до його лісу. Мішка хотів принести хлопчика в жертву, але дізнавшись, що хлопчик розуміється в програмуванні, вирішив зіграти з ним у гру “Лісові Шахи”.

Гра має наступні правила: на дошці розміром $3 \times N$ у першому рядку дошки знаходяться N чорних пішаків Злого Мішки, у третьому рядку знаходяться N білих пішаків хлопчика, відповідно другий рядок пустий.

Злий Мішка та хлопчик ходять по черзі, починає хлопчик. На кожному кроці гравець обирає пішак, яким або робить хід, або б’є ворожого пішака. Відповідно до правил пішаки ходять на одну клітину вперед, тобто якщо пішак білий, то він може піти з третього рядка на другий, а потім з другого на перший. Якщо пішак чорний, то все навпаки, він може піти з першого рядка на другий, а потім з другого на третій. Звісно, та клітинка, на яку ходить пішак, має бути пустою. Пішак б’є фігури по діагоналі на одну клітинку. В грі “Лісові Шахи” бити фігури є обов’язковим, тобто якщо можна бити фігуру, то її треба обов’язково побити на цьому кроці! Програє той, хто не зможе зробити хід. Ваша задача допомогти хлопчику вказати всі можливі перші ходи, що 100% призведуть до його виграшу, якщо він буде дотримуватись оптимальної стратегії.

Вхідні дані. Програма читає з клавіатури одне число N ($1 \leqslant N \leqslant 1000$).

Результати. Програма виводить на екран спочатку число K, що є кількістю перших ходів хлопчика, що 100% приведуть його до виграшу, якщо він буде дотримуватись оптимальної стратегії. Якщо таких немає, тоді треба вивести 0. Далі йдуть K чисел у порядку зростання, що показують, яким за номером білим пішаком повинен піти хлопчик.

Білі пішаки нумеруються від 1 до N зліва направо. Всі числа виводяться через пропуск.

Приклади:

Вхідні дані	Результати
3	1 2
2	2 1 2

Розбір задачі. Розв’язуючи цю задачу, треба спочатку прикласти певні зусилля, щоб побачити, що вона зводиться до викреслювань клітинок стрічки початкових розмірів $N \times 1$, приблизно схожих (але не в усіх деталях!) на попередню пару задач 1.4:D–1.4:E зі спільними правилами гри.

У поточній версії цього посібника утримаюся від опису всього процесу перетворення умови задачі від пішаків до викреслювань клітинок, але прошу повірити, що після цієї підказки це нескла́дно. Просто акуратно розгляньте, до яких наслідків призводить початковий хід пішака: (а) у крайньому стовпчику; (б) у не крайньому стовпчику.

Після цього, запишіть, яким виходить змінений (враховуючи зміни правил викреслювань) під цю задачу аналог формули (11), і запрограмуйте її. (Або, якщо вже є код, який розв’язує задачу 1.4:D, внесіть у нього відносно невеликі зміни.)

Наведу також кілька перших значень SG.

i	0	1	2	3	4	5	6	7	8	9	10
$SG(i)$	0	1	1	2	0	3	1	1	0	3	3

(Порахуйте, чи то програмою, чи то вручну за тією формулою, яку Ви вивели для «Лісових шахів» замість (11), і порівняйте.)

1.4.6 Причина, чому числа Шпрага–Ґранді не можна просто поєднати з упередженими іграми

Тут *не* «доводиться принципова неможливість», а лише пояснюється неправильність способу, який може легко спасти на думку, здається більш-менш природнім, але насправді не працює. Водночас, це

не означає, ніби не може бути якихось інших способів. Можуть. Але вони складніші, обмеженіші, й залишаться за межами нашої дисципліни.

Після того, як у розд. 1.2.4 досить просто (хай і ціною подвоєння кількості позицій, а отже й обсягу перебору) зуміли пристосувати виграшні/програшні позиції до упереджених ігор, може здаватися, ніби ця порада універсальна й може бути вжита скрізь.

Однак, ні.

Наведу загальні міркування та конкретний контрприклад (взяті з [24]), які показують несумісність розширення позиції інформацією «чий хід?» та повноцінної версії чисел Шпрага–Ґранді.

Міркування: якщо гра являє собою суму k підігор, і деякий гравець робить хід у деякій підгрі – у (розширеній) позиції конкретно цієї підгри інформація «чий хід?» правильно змінюється на протилежну, але ж у решті $k-1$ підігор цього не відбувається, й іншому гравцю дістаються чужі (розширені) позиції цих $k-1$ підігор, він не може там ходити так са́мо, як гравець, кому ці розширені позиції належать, і через це стандартне доведення теореми Шпрага–Ґранді порушується.

На це можна було б відповісти «А ми доведення й не вивчали, не знаємо, чи це так» та/або «Проблеми з доведенням твердження ще не гарантують хибності цього твердження; в історії математики траплялися випадки, коли після узагальнення деякої теорії класичні доведення деяких її теорем ставали неправильними, але самі́ теореми лишалися правильними й були правильно доведені іншим способом». Тому, далі наведено контрприклад, який явно показує хибність твердження, ніби завжди можна застосувати звичайні способи обчислення SG-оцінок позицій, розширивши позиції інформацією «чий хід?».

Розглянемо таку гру: «Є смужка, яка спочатку складається з N клітинок, розташованих як одна безперервна послідовність. 1-й гравець може на кожному своєму ході викреслити будь-які 2, або 4, або 16 клітинок, які йдуть поспіль. 2-й може викреслити на кожному своєму ході будь-які 1, або 2, або 3 клітинки, які йдуть поспіль. Програ́є той, кому нема як ходити (тобто, гра має нормальну умову завершення, що й треба для теореми Шпрага–Ґранді)». Спробуємо розширити поняття позиції, включивши, додатково до N булівських значень, які позначають, чи є відповідна клітинка досі не викресленою, ще число g (де $1 \leqslant g \leqslant 2$ — «чий хід?»). Нехай початкове значення $N = 8$. Тоді початковою позицією є $(1, 11111111)$, і один з можливих ходів з неї — у позицію $(2, 11100111)$, тобто викреслити дві клітинки рівно посере́дині. ***Якби*** узагальнення теореми Шпрага–Ґранді на позиції, розширені інформацією «чий хід?», було правильним, то позиція була б рівносильна сумі двох однакових позицій $(2, 111)$, і було б $SG((2, 11100111)) = SG((2, 111)) \oplus SG((2, 111)) = 0$ (бо $a \oplus a = 0$ завжди, зокрема і при $a = SG((2, 111))$, і для цього не обов'язково знати конкретне значення $SG((2, 111))$). Тобто, хід «викреслити дві клітинки рівно посередині» забезпечував би виграш 1-го гравця, бо супернику діставалася б програшна позиція, $SG(\text{якої}) = 0$.

Однак, насправді позиція $(2, 11100111)$ не є програшною. 2-й гравець може, наприклад, викреслити три крайні ліві клітинки й тим перейти до позиції $(1, 00000111)$. Після цього, 1-й гравець мусить викреслити дві підряд клітинки з трьох, що лишилися (інакше кажучи, перейти або до позиції $(2, 00000001)$, або до позиції $(2, 00000100)$). В будь-якому з цих випадків, на наступному ході 2-й гравець викреслює одну останню клітинку й виграє. Враховуючи стандартну аксіому теорії ігор, що обидва гравці бажають виграти, це означає, що початковий хід 1-го гравця «викреслити 2 клітинки рівно посере́дині» насправді унеможливлює його виграш. Тобто, маємо протиріччя, джерелом якого є припущення, ніби можна розширити позиції інформацією «чий хід?» і після цього надалі рахувати SG-оцінку суми ігор як побітовий ксор SG-оцінок підігор.

Насамкінець: деякі засоби часткового поєднання ідеї розбиття гри в суму підігор (не в точності розглянута сума ігор і не в точності теорема Шпрага–Ґранді, а деякі альтернативи) взагалі-то існують, але залишаться поза межами нашої навчальної дисципліни.

1.4.7 Причина, чому числа Шпрага–Ґранді не можна просто поєднати з ідеями розд. 1.3.3 і зробити «мі́зерні числа Шпрага–Ґранді»

Тут теж, як і в розд.1.4.6, *не* «доводиться принципова неможливість», а лише пояснюється неправильність способу, який може легко спасти на думку, здається більш-менш природнім, але насправді не працює. Водночас, це не означає, ніби не може бути якихось інших способів. Можуть. Можуть бути окремі доведення, що в деяких окремих іграх, які використовують SG, працює щось цілком аналогічне. Можуть бути окремі модифікації того підходу для окремих ігор, що використовують SG. Але «ідея» «просто поєднати числа Шпрага–Ґранді з ідеями розд. 1.3.3» не працює, й багато хто заявляє, що, ймовірно, нема універсального способу розв'язувати мі́зерні варіанти будь-яких ігор, «нормальні» варіанти яких розв'язують числами Шпрага–Ґранді.

У Німі палички лише забирають, не додаючи. А для значень SG це не зовсім так: формула (7) гарантує, що з позиції з деяким SG можна перейти у позицію з будь-яким меншим SG, але не заборонені ні переходи у позиції з більшим SG, ні ситуації, коли менше SG отримується як побітовий ксор більших, як-то $2\oplus 2=0$, $4\oplus 5=1$, тощо.

Припустимо, ніби, незважаючи на це, переносити ідею мі́зерного Німа на довільні SG можна завжди й буквально, використовуючи цілком аналогічну до (3) умову «позиція програшна $\Leftrightarrow$ $\Leftrightarrow$ $SG(s_1)\oplus SG(s_2)\oplus\ldots\oplus SG(s_k)\oplus u\big(SG(s_1),SG(s_2),\ldots,SG(s_k)\big)=0$», з такою ж $u(\ldots)$, як у (4).

Нехай до того ж спочатку гра не є сумою підігор (інакше кажучи, спочатку є лиш одна підгра, рівна всій грі). Тоді повинно бути правильно&вигідно «заганяти» суперника у таку позицію p, де $SG(p)=1$, «щоб він не мав інших варіантів, крім як піти до позиції, $SG(\text{якої})=0$, і тим зробити останній хід, отже програти»… тільки от, на відміну від Німа, SG=0 *не* гарантує кінця гри; можливий і альтернативний варіант, що SG=0, бо цим гра розпалася в суму двох підігор, таких, що $SG(s_1)=$ $=SG(s_2)=2$.

Тобто, ми «заганяли» суперника в позицію, де $1\oplus u(1)=1\oplus 1=0$, а в результаті самі отримали позицію, де $2\oplus 2\oplus u(2,2)=2\oplus 2\oplus 0=0$. Ніби як суперник походив із програшної позиції у програшну, що протирічить означенню програшної позиції. І біда не з означенням програшної позиції (з ним усе гаразд!), а з тим, що ми *безпідставно* вирішили, ніби (3) можна узагальнити з конкретної гри Нім на довільні ігри, де застосовні числа Шпрага–Ґранді.

(На це можна пробувати відповісти «але ж у нас було ще одне безпідставне припущення: що SG=0 пов'язане са́ме з $SG(s_1)=SG(s_2)=2$, а не з чимось іншим; може, причина протиріччя в ньому?». І так, і ні. Це справді теж не доведене припущення. Але я будую не доведення, а контрприклад (до твердження, ніби (3)–(4) можна узагальнювати на *будь-які* ігри, до «нормальної» версії яких застосовні числа Шпрага–Ґранді), тому можу пред'являти приклад, і казати: «глядіть, отут не вийшло, й цього вже досить, щоб казати, що Ваше твердження *не завжди* правильне, а Ви ж казали, ніби завжди». Важливо, звісно, щоб наш приклад був можливим. Але з цим усе гаразд: наприклад, якщо продовжити побудову чисел Шпрага–Ґранді для гри Ґранді (початок розгляду тут), виявиться, що $SG(28)=1$, можна розбити 28 як 17+11, і $SG(17)=$ $=SG(11)=2$.

Мені не відомі контрприклади, які не містять розбиття гри в суму ігор. Схоже (але це не точно), що якщо таких розбиттів нема або якщо вони настають лише до (а не після) переходу від $\max\big(SG(s_1),\ldots\big)\geqslant 2$ до $\max\big(SG(s_1),\ldots\big)\leqslant 1$, то узагальнювати (3)–(4) на числа Шпрага–Ґранді все-таки можна. Але, навіть якщо це правда, все'дно це стосується не всіх ігор, до «нормальних» версій яких застосовні числа Шпрага–Ґранді, а лише частини з них.

Є й інші ситуації, коли можна якось розв'язати мі́зерну версію задачі, використавши зокрема й числа Шпрага–Ґранді. Але це теж окремі випадки, а не універсальний спосіб.)

1.5 Що робити, якщо гра допускає нічиї́?

В межах цього розд. 1.5, розглядаємо лише послідовні дискретні детерміновані скінче́нні ациклічні ігри двох гравців з повною інформацією, і вважаємо, що можливих результатів є рівно три:

1. 1-й гравець виграв, 2-й програв;
2. нічия, ніхто не виграв і ніхто не програв;
3. 2-й гравець виграв, 1-й програв.

Для 1-го гравця варіант 1 найкращий, 2 — середній, 3 — найгірший; для 2-го все симетрично: 3 — найкращий, 2 — середній, 1 — найгірший.

В цих рамках, нічиї можуть настати лише якщо правилами гри передбачено, що потрапляння гри в деякі позиції означає нічию (в наступному розд. 1.6 з'явиться додаткова причина для нічиїх, але поки що так), і це означає, що гра *не може* мати «нормальну умову завершення», а отже незастосовні (принаймні, по-простому) всі прийоми розд. 1.4.

В цих рамках, ніщо не заважає мати хоч неупереджені, хоч упереджені ігри, відмінність між ними лишається такою ж, як у розд. 1.1.8 та 1.2.4.

Ніщо не заважає й узагальнити означення виграшних та програшних позицій, дані на самому початку розд. 1.2, таким чином.

1. Позиція *програшна*, коли або тим фактом, що гравцю дісталася ця позиція, він вже програв, або кожен хід з неї веде до виграшної позиції.
2. Позиція *виграшна*, коли або тим фактом, що гравцю дісталася ця позиція, він вже виграв, або існує хоча б один (можна й більше) хід з неї, що веде до програшної позиції; в які ще позиції ведуть інші ходи — байдуже.
3. Позиція *нічийна*, коли або тим фактом, що гравцю дісталася ця позиція, гра вже закінчилася внічию, або ходів з неї до програшних позицій не існує, зате існує хоча б один (можна й більше) хід з неї, що веде до нічийної позиції; чи є також (ходи/хід) у виграшн(і/у) позиці(ї/ю) — байдуже.

Все це зручно підсумувати (і, мабуть, легше запам'ятати) так:

1. якщо ходів нема, то дивимось результат за тим, яка це позиція;
2. якщо ходи є, то намагаємося загнати суперника в якнайгіршу позицію:
   (а) якщо вдається у програшну — найкращий варіант, ця позиція виграшна;
   (б) якщо у програшну не вдається, зате вдається у нічийну — so-so, ця позиція нічийна;
   (в) якщо нема інших варіантів, крім як у виграшну — найгірший варіант, ця позиція програшна.

Раніше твердження, що виграшність позиції гарантує виграш *лише ідеальному* гравцеві, теж перетворюється очевидним чином. Виграшність/програшність/нічийність гарантує са́ме цей результат, якщо обидва гравці ідеальні; якщо один гравець ідеальний, інший може допускати помилки — в ідеального гравця результат буде не гірший, чим випливає з оцінки виграшності/програшності/нічийності; якщо обидва гравці можуть допускати помилки — ніяких гарантій, і неясно, навіщо таким гравцям математична теорія ігор.

Чи лишаються правильними зворотна індукція та рекурсія з запам'ятовуваннями? В цілó-му — так. Відмінності є, але більш-менш очевидні. Як і раніше, знайшовши хід у програшну позицію, можна остато́чно оголошувати поточну позицію виграшною, не перебираючи подальші варіанти ходу з неї. Але що слід робити, знайшовши хід у нічийну? Запам'ятати це в якомусь прапорці — необхідно (ми щойно взнали, що поточна позиція не є програшною); однак, далі може бути (а може й не бути) хід у програшну, тому ще не відомо, якою (нічийною чи виграшною) кінець кінцем виявиться поточна позиція, а щоб це з'ясувати, треба продовжити цикл перебору можливих ходів з неї. Все це однаково стосується як ітеративної зворотної індукції *без* «оптимізаційного прийому», так і рекурсії з запам'ятовуваннями.

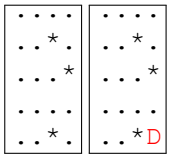
А що з «оптимізаційним прийомом»? У нього варто внести вельми *не*очевидні зміни. Найсуттєвішу дію «щоразу, знайшовши програшну позицію, відразу позначати як виграшні всі позиції, звідки є хід у поточну (яку ми щойно визначили як програшну)» варто лишити незмінною, бо це вона дає основне пришвидшення, і са́ме в ній нічого не змінюється від того, що з'явилися нічиї. Однак, слід з'ясувати нове питання «чи варто, знайшовши нічийну позицію, якось позначати всі позиції, звідки є хід у неї?» і заново розглянути старе питання «чи можна вважати, що коли позиція не позначена, то вона точно програшна?», і з обома все не так просто.

Один з можливих правильних способів такий. *Не* робити ніяких спеціальних дій щодо позначення позицій, звідки є хід у щойно визначену нічийну. Отже, ще не позначена позиція цілком може виявлятися хоч програшною, хоч нічийною, хоч виграшною-тому-що-таке-закінчення-гри. Але оце старе доведення (з урахуванням приміток до нього в розд. 1.2.3) все ще лишається правильним у тій частині, що ще не позначена позиція не може бути виграшною-бо-з-неї-є-ходи-у-програшн(у/і). А це дозволяє робити так: для кожної ще не позначеної позиції, з якої є ходи, запускати перебір цих ходів із неї, для визначення, чи вона програшна, чи нічийна, але варто обривати цикл перебору ходів, коли знайдено хід у нічийну позицію.

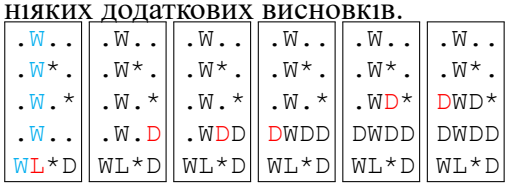
(Раніше обривати цикл було «не можна», а тепер «варто». Це взагалі нормально? Так, нормально, бо змінилися обставини: тоді перебір варіантів ходу з поточної позиції міг знайти спочатку варіант ходу в нічийну, а потім варіант ходу в програшну (який робить поточну позицію виграшною). А тепер це неможливо, й кільканадцять рядків тому було посилання на доведення, чому поточна позиція не може бути виграшною-бо-з-неї-є-ходи-у-програшні. Тож усе гаразд.)

Приклад заповнення поля позначками виграш/програш/нічия. Розглянемо «Фішку на мінному полі» з розд. 1.2.2, але змінимо правила гри так: нехай неможливість ходу не завжди означатиме програш гравця, якому дісталася ця позиція, а залежить від того, де це сталося: завершення гри у правому-нижньому кутку всього поля означає нічию, а в іншій клітинці, з якої нема ходів (бо знизу та/або праворуч міна), як і раніше, означає виграш того, хто походив у таку клітинку, й програш того, кому вона дісталася.

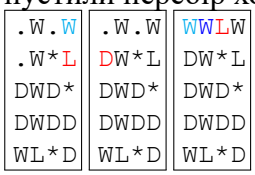
(Дуже поважних причин для са́ме такої зміни правил гри нема; просто, щоб показати описані зміни у зворотній індукції, потрібна *якась* гра з нічиїми. Втім, з кількох прикладів, які я підбирав навмання, цей виявився відносно кращим: нічийні позиції і з'явилися, й не поглинули собою геть усе.)



Заповнимо поле гри аналогічно цьому прикладу, але згідно щойно описаної зміни в оптимізованому варіанті зворотної індукції, і, звісно, користуючись як раніше оголошеними позначками "W" і "L", так і новою позначкою "D" (від **D**raw — нічия). На лівому рисунку зображене поле, як воно задане у вхідних даних. На правому права-нижня клітинка-позиція визначена нічийною, що *не* тягне за собою ніяких додаткових висновків.



На найлівішому з рисунків знайдено програшну клітинку-позицію; для неї, як і раніше, позначаємо виграшними всі, звідки є ходи у неї. На кожному з п'яти подальших рисунків чергова клітинка визначена нічийною (бо запустили перебір ходів ***з*** неї і знайшли хід у нічийну).

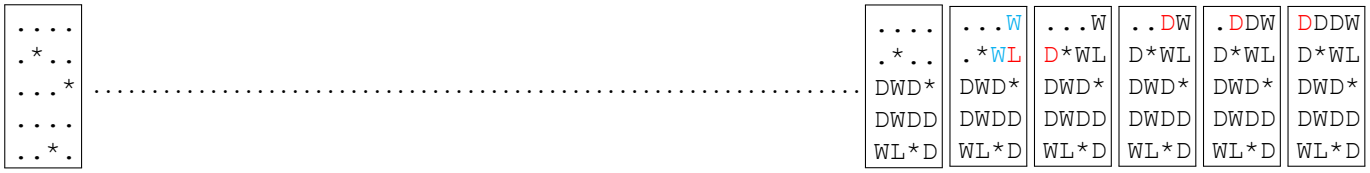


На лівому рисунку знайдено програшну клітинку-позицію й позначено виграшними всі (саме зараз — єдину), звідки є ходи у неї. На середньому чергова клітинка визначена нічийною (бо з неї є хід у нічийну). На правому знову знайдено програшну й позначено виграшними всі, звідки є ходи у неї.

Після цього непозначених клітинок більше нема.

Са́ме на цьому полі са́ме цієї гри початкова позиція виявилася виграшною. Але насправді це досить рідкісна ситуація, яку довелося штучно конструювати. Є така тенденція (на жаль, поки що не виміряна ніякими математичними засобами), що нічийні позиції частенько «витісняють» виграшні та програшні, за такою схемою: спочатку десь з'являються ходи не лише у виграшні позиції, а й у нічийні, і так щезають програшні позиції; потім нема ходів у програшні, й так щезають виграшні позиції, залишаючи самі нічийні.

(В цьому навіть можна віднайти щось «правильне» — особливо, якщо вважати, ніби «завершення гри внічию справедливіше, чим виграш одного гравця і програш іншого». Але етичні оцінки не є предметом розгляду математичних та технічних дисциплін, тому просто лишаю це як спостереження, а ставлення до цього формуйте кожен своє.)



Наприклад, розгляньмо поле, що відрізняється від попереднього прикладу *лише* розміщенням однієї (найближчої до старту) міни. Початок заповнення позначками цього пóля майже ідентичний попередньому прикладу (з очевидною зміною, що у 2-му зліва стовпчику позначки “W” не доходять до верху, бо впираються в міну). Але далі (3-й справа, він же передпередостанній, рисунок) бачимо, що передостання клітинка-позиція верхнього рядка стає нічийною замість програшної (бо тепер, на відміну від попереднього прикладу, з неї можна піти на 2 або на 3 вниз у нічийну клітинку-позицію), й далі, до початкової позиції (лівої-верхньої клітинки) включно, так і не з’являється ні програшних, ні виграшних позицій, лише нічийні.

1.6 А якщо гра має цикли?

З уже сказаного в розд. 1.1.5 випливає: цикли — послідовності ходів, які поверають до позицій, вже відвіданих у ході/сеансі/партії цієї гри. Розд. 1.2–1.5 розглядають випадок, коли цикли заборонені. Однак, ігри з циклами теж існують, тож тепер поговоримо про них.

1.6.1 Чому цикли — проблема?

І зворотна індукція, і рекурсія з запам’ятовуваннями працюють ***завдяки*** *відсутності циклів*, а коли цикли є — вони, переважно, непридатні.

Якщо застосовувати до гри, де є цикл(и), зворотну індукцію (наприклад, без «оптимізаційного прийому»), може виникати така проблема: для визначення виграшності/програшності поточної позиції потрібно знати виграшність/програшність позиції, яку ще не дослідили, тож не знаємо, виграшна вона чи програшна.

(Хтось може сказати, що в н(ього/еї) така проблема траплялася і з ациклічними іграми; однак, з ациклічними іграми таке може траплятися, *лише* якщо вибрати поганий порядок розгляду позицій; якщо ж гра має цикли, ця проблема може бути об’єктивно спричинена самóю грою, й це не справиш іншим порядком розгляду.)

При спробі застосовувати рекурсію із запам’ятовуваннями (яка, як сказано тут, «повинна самá розбиратися з порядком вирішення підзадач/позицій») чуда теж не стається: це з ациклічними залежностями така рекурсія правильно розбирається самá, а якщо є цикли — рекурсія може повторно викликати себе для тієї ж позиції, яка вже є у стеку рекурсії, що може давати (залежно від того, як написати код, та від особливостей гри) аварійне завершення із переповненням стека, або не завжди правильний результат.

1.6.2 Чи буває, що цикли є, але не створюють проблем?

Як окремі часткові випадки — іноді буває. Власне, тому в розд. 1.6.1 і написано, що проблеми *можуть* виникати, а досі розглянуті засоби *переважно* непридатні. Питання «За яких обставин цикли не створюють описаних проблем?» важке, розглянути його повністю нереально. Але розглянемо хоча б частково.

Приклад «Баше + хід “3→9”». Розгляньмо гру, яка має рівно одну відмінність від класичної гри Баше з розд. 1.2.1: додатково до правил, що на кожному кроці можна брати або 1, або 2, або 3 паличк(и/у), з’являється спеціальний хід: коли лишається рівно 3 палички, не більше й не менше, то чарівний «вжух», замість забирати палички, додає ще 6 паличок, після чого хід переходить до суперника; застосовувати «вжух» можна скільки завгодно разів (зокрема й не застосовувати зовсім), але лише коли в купці рівно 3 палички.

У так модифікованій грі Баше, теоретично можлива нескінче́нна послідовність «було 9 паличок; 1-й гравець забрав 3, лишилось 6; 2-й гравець забрав 3, лишилось 3; 1-й гравець зробив “вжух”, лишилось 9; 2-й гравець забрав 3, лишилось 6; 1-й гравець забрав 3, лишилось 3; 2-й гравець зробив “вжух”, лишилось 9; і так до нескінче́нності». Однак, хоч така послідовність і *можлива*, але *чи варто* гравцям справді так грати? В са́ме так модифікованій грі — однозначно, *ні*. Хоча б тому, що хід «3→9» робиться з позиції «3 палички», а з цієї ж позиції можна походити «3→0» (забрати всі 3 палички) й тим негайно виграти; хід «3→9» точно не кращий (бо взагалі ніщо не краще, чим негайно виграти). Отже, гра Баше, розширена *са́ме конкретно ходом 3→9*, за умови правильної гри обох гравців нічим не відрізняється від стандартної гри Баше. От і виходить: у са́ме так модифікованій грі цикл є, але не створює проблем.

Приклад «Баше + хід “5→7”». Міркування й висновок виходять ті са́мі: хід створює можливість циклу, але фактично використовувати цю можливість не варто, бо замість “5→7” краще піти “5→4” й цим забезпечити свій виграш на наступному своєму ході. Більш того: це правда і якщо хід “5→7” єдиний, яким розширили звичайну гру Баше, і якщо її розширили обома ходами “5→7” та “3→9”.

1.6.3 Додаткова аксіома «краще цикл, чим програш»

Приклад «Баше + хід “4→5”». Цей приклад докорінно відрізняється від двох попередніх. Нехай нам дісталася позиція «4 палички». Чого можна досягти з цієї позиції звичайними ходами гри Баше? Лише піти “4→3” або “4→2” або “4→1”, після чого програти (для “4→3” та “4→2” — якщо суперник не помилиться, але ж неможливо гарантувати, що він помилиться). Тут альтернатива “4→5” вже видається значно привабливішою. Звісно, неможливо гарантувати, що суперник, якому дістанеться позиція «5 паличок», піде “5→3” або “5→2” і тим дозволить своєму супернику (нам) виграти; він, мабуть, захоче піти “5→4”. Тобто, ***гравцям вигідно затягнути гру в цикл, щоб не програти***.

Стандартна аксіома ігор з циклами са́ме така́: завести гру в цикл вигідніше, чим програти (але менш вигідно, чим виграти).

Питання «то треба, щоб завести гру в цикл було вигідно обом гравцям, чи досить, щоб лише одному?» не має чіткої відповіді, бо тут залежність трохи інша. Якщо хтось із гравців може чи то вийти з циклу, чи то не заходити в нього, й після цього виграти (гарантовано, що б не робив у межах правил гри інший гравець) — тоді згода чи незгода іншого гравця з таким ходом гри неважлива, це все’дно зроблять. Але якщо вийти з циклу можна лише у виграшну позицію (яка дістається супернику, тобто вийти з циклу можна лише ціною свого програшу) — тоді повтори циклу будуть вигідні, й неважливо, чи мова про упереджену гру й невигідність покидати цикл кожному з суперників, чи про неупереджену, де позиції однаковісінько стосуються кожного з гравців.

1.6.4 Ретроаналіз

Слова́, що починаються на «ретро-», часто позначають щось несучасне та/або застаріле. Але є й трохи інші смисли (які зараз і потрібні) — «назад», «у зворотньому порядку».

Термін «*ретроаналіз*», він же «*ретроспективний аналіз*», він же «*ретроградний аналіз*» часто використовується у прив’язці до шахів, як у задачі 1.6:B. Однак, головні ідеї можна застосувати й до інших ігор. Відповідно, ці ідеї називають ретроаналізом також і для інших ігор.

(Ем… то тут заявляється, ніби є спосіб розв’язати шахи?

І так, і ні. Для «шахів у цілóму» — навіть якщо якось (невідомо, як) обійти (згадані тут і тут) проблеми потенційної нескінченності та потенційно надто великої додаткової до поточного розташування фігур інформації — все’дно і пам’яті, і швидкодії комп’ютерів не вистачає настільки кардинально, що мабуть не вистачатиме взагалі нікóли в майбутньому. Тому, довести ці ідеї до програми ідеальної гри в шахи практично не вийде.

Але для сильно зменшених випадків це працює, практично й на нині існуючих комп’ютерах. Зокрема, для вже згаданої задачі 1.6:B, де розглядається поле, на якому лишилися тільки білий король, білий ферзь і чорний король, я писав такий код, і для наших студентів він складни́й, але в принципі «підйомний». А інші люди писали аналогічний код для ситуацій, коли лишалося не 3 фігури, а 4–6, і це, хоч і складніше в написанні та довше у виконанні, теж працює. Практично.)

Умови застосовності ретроаналізу (якою повинна бути гра, щоб його можна було застосувати) в різних джерелах описують дещо по-різному, але особисто я виходжу з таких припущень/аксіом.

1. Грають рівно два гравці.
2. Виграш не має числового вираження.
3. Гра може містити цикли.
   (Ретроаналіз *можна* застосувати також і до ациклічних ігор, але для них, як правило, доречніший якийсь інший спосіб. Хоча, якщо раптом важливо дотриматися вимоги 5 цього переліку — ретроаналіз стає доречним також і для ациклічних ігор.)
4. Кожному з гравців завести гру в цикл вигідніше, чим програти (але менш вигідно, чим виграти).
   (Це в точності аксіома з розд. 1.6.3.)
5. Якщо гравець може виграти, йому вигідно виграти за якнайменшу кількість ходів; якщо гравець може лише програти (не може ні виграти, ні завести гру в нескінче́нний цикл), йому вигідно затягти гру, щоб було якнайбільше ходів.
   (Може здатися, ніби ця вимога виконується дуже рідко, бо зазвичай важливий лише результат, але неважливо, більше чи менше буде ходів. Однак, якщо їм це неважливо (не так, як кажуть «неважно, на каком язьіке», а потім таки наполягають на російській, а *справді* неважливо), то нехай погодяться на цю додаткову умову. Їм неважливо, а ретроаналізу легше, й усі задоволені.
   Але якщо у гравців є якісь інші наміри щодо кількості ходів — описана тут версія ретроаналізу діятиме всупереч тим намірам, і не факт, чи вдасться модифікувати ретроаналіз під ті наміри.)
6. Ретроаналізу більш-менш байдуже, чи гра неупереджена, чи упереджена (партійна).
   (У смислі розд. 1.1.8. На деталі реалізації це, звісно, впливає; але якщо пам'ятати й правильно (див. також розд. 1.2.4) враховувати, чи гра неупереджена й байдуже, якому гравцю дістається позиція, чи різні гравці ходять по-різному — реалізувати ретроаналіз вийде і там, і там.)
7. Гра має «нормальну умову завершення» (у смислі розд. 1.1.7, тобто «хто не має куди ходити, програ́є»).
   (Але це більше для того, щоб хоч трохи спростити й так громіздке викладення. Ретроаналіз *можна* модифікувати хоч під «хто не має куди ходити, виграє», хоч під урахування нічиїх у смислі розд. 1.5, хоч під ще деякі варіанти. Але деякі кроки все-таки треба змінювати, й це залишиться за межами цього посібника.)

Тут і далі, будемо вживати термін «*півхід*», у смислі ходу лише одного з двох гравців. Наприклад, хід лише білих чи лише чорних у шахах. А якщо і не в шахах, ми все'дно розглядаємо гру двох гравців.

(Англійською, відповідні терміни «ply» та «half-move» досить поширені (див., зокрема, тут і тут, в описі поняття «move»), однак знайти поважних україномовних джерел з терміном «півхід» (чи таким же поняттям, названим іншим словом) мені не вдалося.)

Тож, якщо перелічені умови дотримані, можна діяти згідно з ось цим алгоритмом ретроаналізу.

0. Почнемо з пошуку всіх *позицій-кінців*, у яких гра щойно завершилася виграшем однієї зі сторін.
   (В іграх, аналогічних грі Баше — паличок нема (тут така позиція єдина). В іграх, аналогічних «фішці на мінному полі» — з поточної клітинки-позиції нема ходів (тут єдиність чи не єдиність такої позиції залежить від розміщення мін). У шахах — щойно поставлено мат (тут таких позицій дуже багато, бо сам король може перебувати в різних клітинках, інші фігури обох гравців теж можуть перебувати в різних клітинках, і набори цих інших фігур можуть відрізнятися). І так далі.
   Враховуючи припущення 7, очевидно, що всі позиції-кінці є програшними, але не всі програшні позиції є позиціями-кінцями.)
1. Спираючись почергово на кожну зі знайдених у попередньому пункті позицій-кінців, знайдемо всі *1-півходові позиції*; це такі позиції, ***хоча б один*** хід (одного гравця, інакше кажучи півхід) з яких веде у якусь із позицій-кінців.
   (В іграх, аналогічних грі Баше — палички є, і якраз у такій кількості, що можна забрати зразу всі. В іграх, аналогічних «фішці на мінному полі» — можна одним кроком перемістити фішку в таку клітинку, з якої нема куди йти. У шахах — існує можливість так перемістити одну свою фігуру, щоб негайно поставити супернику мат. І так далі.
   Усі 1-півходові позиції є виграшними. Тому, природньо, що тут мова про хоча б один (пів)хід, аналогічно тому, як виграшною є позиція, з якої існує хоча б один хід у програшну.
   І так само, як виграшна позиція не винувата, якщо неідеальний гравець зробить поганий хід із неї й кінець кінцем програ́є, так і тут: це вже проблема гравця правильно вибирати свій хід, і можливість інших ходів (крім ходів у позиці(ю/ї)-кін(ець/ці)) не заважає позиції бути 1-півходовою.

Також проговорю очевидне, що назва «1-півходова» виражає «за один півхід до кінця гри», й аналогічно 2-півходова, 3-півходова, …)

2. Спираючись на знайдені в попередньому пункті 1-півходові позиції, знайдемо всі *2-півходові*; це такі позиції, ***всі*** ходи (одного гравця, інакше кажучи півходи) з яких ведуть у якісь із 1-півходових позицій.
   (Навести переконливі приклади з різних ігор тут уже трохи важче, хіба що «у класичній грі Баше це позиція “4 палички”: всі варіанти ходу з неї ведуть до тих позицій, у яких суперник зможе забрати всі палички й виграти».
   Очевидно, що всі 2-півходові позиції є програшними (але не всі програшні є 2-півходовими). Тому тут мова про всі (пів)ходи, аналогічно тому, як всі ходи із програшної позиції мають вести у виграшні. Щоб супернику діставалася виграшна позиція, і гравець (байдуже, ідеальний чи ні) нічого не міг із цим вдіяти (у рамках правил гри). А станом на цей момент виконання ретроаналізу, відомими виграшними позиціями є лише 1-півходові, тому й усі (пів)ходи мусять бути лише до них)

Далі продовжуємо приблизно аналогічно будувати 3-півходові, 4-півходові, 5-півходові, … позиції, й усі k-півходові позиції є виграшними при непáрних k і програшними при пáрних. Але деяка відмінність від досі розглянутих усе-таки є, тому запишу ще дві ітерації.

3. Спираючись почергово на кожну зі знайдених у попередньому пункті 2-півходових позицій, знайдемо всі *3-півходові*; це такі позиції, ***хоча б один*** (пів)хід з яких веде у якусь із 2-півходових позицій; але, якщо позиція вже позначена як 1-півходова (бо з неї є і (пів)хід у позицію-кінець, і (пів)хід у 2-півходову) — її слід пропускати, не оголошуючи 3-півходовою.
   (Це і враховує вищезгадане припущення 5, і корисно для кращого розуміння, коли треба ще продовжувати такі кроки з побудови сукупностей 1-, 2-, 3-, …, k-, …-півходових позицій, а коли варто вже припинити.
   Можливість ще інших (пів)ходів (і не в 2-півходову, і не в позицію-кінець) не заважає позиції бути 3-півходовою, аби був хоч один (пів)хід у якусь 2-півходову й не було жодного (пів)ходу ні в яку позицію-кінець.)
4. Спираючись на все раніше знайдене, знайдемо всі *4-півходові позиції*; це такі позиції, ***всі*** (пів)ходи з яких ведуть лише в 1-півходові та/або 3-півходові позиції, й хоча б один з них — у 3-півходову.
   (На відміну від позаминулого пункту, тепер ми знаємо про виграшність не лише 1-півходових, а ще й 3-півходових позицій. А 4-півходові повинні бути програшними. Отже, правило «всі ходи з програшних позицій повинні вести у виграшні» дотримане.
   Чому для оголошення позиції 4-півходовою треба, щоб був хід з неї у 3-півходов(у/і) (а не лише в 1-півходов(у/і)), очевидно.
   Чому позиції, з яких є ходи як у 3-півходов(у/і), так і в 1-півходов(у/і), варто оголосити 4-півходовими, а не 2-півходовими? Якщо оголосити їх 2-півходовими, доведеться вертатися й заново знаходити 3-півходові, отже знову знаходити 4-півходові, причому тоді може знову знадобитися заново знаходити 3-півходові, і так далі. Так можна взагалі не дійти навіть до 5-півходових… А треба, бо очевидно, що бувають ігри, для проходження яких треба десятки чи сотні ходів, а швидше неможливо. Натомість, можна просто завжди залишати сукупність k-півходових позицій такими, як їх вперше побудували. Тим паче, що під це є не ідеальне, але зазвичай прийнятне обґрунтування з припущення 5 «якщо гравець може лише програти, йому вигідно затягти гру, щоб було якнайбільше ходів».)

Далі все продовжується цілком аналогічно.

непарні k Спираючись почергово на кожну з уже знайдених $(k-1)$-півходових позицій, знайдемо всі *k-півходові* (при непарному k); це такі позиції, ***хоча б один*** (пів)хід з яких веде у якусь із $(k-1)$-півходових позицій; але, якщо позиція вже позначена як 1-півходова, чи 3-півходова, …, чи $(k-2)$-півходова (перелічені всі непарні, менші k) — її слід пропускати, не оголошуючи k-півходовою. Всі k-півходові (при непарному k) позиції є виграшними.

парні k Спираючись на все раніше знайдене, знайдемо всі *k-півходові позиції* (при парному k); це такі позиції, ***всі*** (пів)ходи з яких ведуть лише в 1-півходові та/або 3-півходові та/або … та/або $(k-2)$-півходові позиції (перелічені всі непарні, менші k), й хоча б один з цих (пів)ходів — у $(k-1)$-півходову. Всі k-півходові (при парному k) позиції є програшними.

А коли все це діло зупиняти? Наприклад, коли чергова сукупність k-півходових виявляється порожньою. Або, наприклад, коли початкова позиція гри нарешті потрапила в перелік k-півходових при деякому k. Або, наприклад, коли виконалася будь-яка одна з цих двох умов. Що з цих варіантів вибрати, може залежати від особливостей гри та/або того, що просять з’ясувати про гру.

При цьому, якщо кількість позицій гри скінче́нна, чергова сукупність k-півходових позицій обов'язково рано чи пізно виявиться порожньою. Хоча б тому, що одна й та ж позиція не може бути водночас і k_1-півходовою, і k_2-півходовою при $k_1 \neq k_2$, тож колись мусять закінчитися позиції.

Легко бачити, що якщо нема $k^\star$-півходових позицій, то не буде й $(k^\star{+}1)$-півходових, $(k^\star{+}2)$-півходових, …, бо (незалежно від парності чи непарності $k^\star$) кожна $(k^\star{+}1)$-півходова позиція мусить мати (пів)хід у $k^\star$-півходову, тож «обірвана» послідовність не може «відновитися» при більших k.

Чи може позиція так і не потрапити до переліку k-півходових ні при якому k? Якщо може, то що це значить? Звісно, може.

Чи можливо, щоб такою позицією, яка нікуди не потрапила, була позиція, з якої нема (пів)ходів? У рамках припущення 7 про «нормальну умову завершення гри» — ні, така позиція програшна (са́ме тому, що нема ходів), її слід віднести до позицій-кінців (які можна для зручності вважати 0-півходовими, але це вже деталі). А випадки порушення припущення 7 я обіцяв залишити за межами цього посібника.

Лишається ситуація, коли з позиції є (пів)ходи, а вона все'дно не потрапила до жодної з сукупностей k-півходових. Сукупності 1-, 2-, 3-, …-півходових вже скінчилися, а вона так і не потрапила. ***Це і є критерій того, що з цієї позиції не можна виграти*** (принаймні, якщо суперник грає оптимально для себе, то не можна)***, але можна завести гру в нескінче́нний цикл*** (навіть якщо суперникові, хоч ідеальному хоч ні, це не подобається, він або взагалі нічого не може з цим вдіяти, або має лише вибір «або погоджуватися з нескінче́нним циклом, або програти»).

(Чому це правда? Якби позиція була виграшною, вона обов'язково зустрілася б серед 1-, 3-, 5-, …-півходових; якби програшною — серед 2-, 4-, 6-, …-півходових. Звісно, щоб ця аргументація була цілком строгою, ці допоміжні твердження теж треба строго довести. Але нехай ці доведення залишаться за межами цього посібника.)

Задача 1.6:A. «Баше з додатковими ходами і циклами»

Двоє грають у таку гру. Спочатку є N ($1 \leqslant N \leqslant 10^5$) паличок. На кожному ході кожен гравець може забирати або одну, або дві, або три палички, але не більше, чим їх є всього. Описані досі ходи (тотожні класичній грі Баше з розд. 1.2.1) будемо називати *традиційними*. Крім них, існують *спеціальні* ходи: задається кілька пар цілих чисел (a_1, b_1), (a_2, b_2), …, (a_t, b_t), які означають: якщо кількість паличок дорівнює якомусь із a_i, то гравець може замінити a_i паличок на b_i. Якщо гравець може зробити спеціальний хід, він сам вирішує, чи робити його, чи якийсь із традиційних, але зробити якийсь один хід треба. Ситуація, коли відразу кілька пар мають однакове значення a_i, можлива; в такій ситуації гравець теж сам вирішує, яким із доступних ходів (спеціальних чи традиційних) скористатися. В будь-якому разі, після кожного ходу черга ходити переходить до іншого гравця, пропускати хід не можна. (Щоправда, користуватися ходом, де $b_i = a_i$, можна, і це в деякому смислі відповідає пропуску ходу; але ж це можливо, лише якщо поточна кількість паличок якраз рівна таким b_i та a_i, для яких існує такий спеціальний хід.) Закінчується гра тоді, коли лишається 0 паличок (це може статися хоч після традиційного ходу, хоч, якщо існує пара, де $b_i = 0$, після спеціального), і той, хто походив у цю позицію, виграв, а той, кому така позиція дісталася, програв. Однак, ця гра може й не завершуватися, бо завдяки спеціальним ходам з $b_i > a_i$ кількість паличок може так ніколи й не стати 0. Такий результат кожен з гравців розцінює як гірший, чим виграш, але кращий, чим програш.

Напишіть програму, яка визначатиме, хто виграє при ідеальній грі обох гравців. Щоб відповідь не так легко було вгадати, задачу слід розв'язати, при одній і тій самій сукупності пар (a_1, b_1), (a_2, b_2), …, (a_t, b_t), для різних початкових кількостей паличок.

Вхідні дані. В першому рядку записане єдине ціле число t ($0 \leqslant t \leqslant 12345$), яке задає кількість пар, що утворюють спеціальні ходи. Кожен з подальших t рядків задає один спеціальний хід, у вигляді $a_i\ b_i$ (два числá, розділені пробілом; дотримано обмежень $1 \leqslant a_i \leqslant 10^5$, $0 \leqslant b_i \leqslant 10^5$). Наступний ($(t{+}2)$-й) рядок містить єдине ціле число k ($1 \leqslant k \leqslant 12345$) задає кількість варіантів початкової кількості паличок, а ще наступний ($(t{+}3)$-й) рядок містить k натуральних чисел N_1, N_2, …, N_k (кожне в межах $1 \leqslant N_i \leqslant 10^5$) – різні початкові кількості паличок, для яких слід розв'язати задачу.

Результати. Виведіть у один рядок без пропусків рівно k великих латинських букв `W` та/або `L` та/або `D`, де `W` позначає, що при відповідній початковій кількості паличок гарантувати собі виграш може 1-й гравець, `L` – що 2-й, а `D` – що жоден з гравців не може гарантувати собі виграшу, але 1-й гравець може гарантувати, що або гра триватиме нескінче́нно довго, або якщо 2-й поступиться, то виграє 1-й.

Приклади:

Вхідні дані	Результати
2 1 1 8 8 12 1 2 3 4 5 6 7 8 9 10 11 12	WWWLWWWDDDDD

Вхідні дані
12 3 5 5 0 8 9 11 4 14 4 12 0 13 0 18 9 19 9 13 18 18 13 1 4 22 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Результати
WWWLWWWDDDWWWWLWWWDDDDD

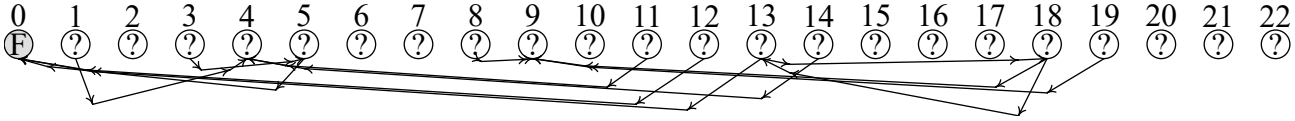
Примітки. У першому прикладі додаткові ходи дозволяють не забирати палички, коли їх у купці або 1, або 8. Якщо в купці 1 паличка, її вигідніше забрати й виграти. Якщо в купці 8 паличок, будь-який зі традиційних ходів веде до програшу, й вигідніше нічого не забирати, що потім повторюється обома гравцями до нескінче́нності. Й так виходить, що з усіх подальших позицій (9, 10, … паличок) теж вигідніше якось (за один хід чи за кілька) прийти до позиції «8 паличок» й повторювати її до нескінче́нності.

У другому прикладі все набагато складніше.

Розбір задачі. Потрібно застосувати до цієї гри ретроаналіз, описаний на кількох попередніх сторінках (68–71). Теоретично до цього додавати майже нічого, хіба сказати, що ця гра неупереджена й тому позицією варто вважати просто кількість паличок. А для демонстрації тієї теорії на практиці, наведемо тут процес ретроаналізу другого (більшого) з прикладів з умови.

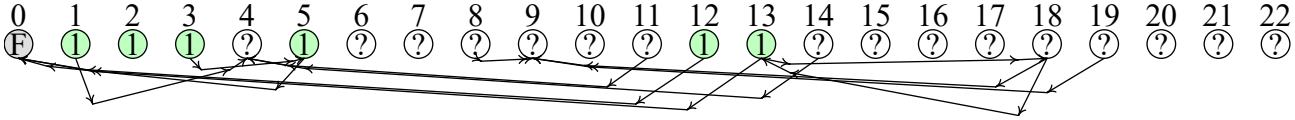
Зобразимо «майже граф гри» (майже як описано тут і тут, але «традиційні» ходи «забрати (1/2/3) паличк(у/и)» пропустимо, щоб не загромаджувати рисунок зайвими лініями), і на ньому будемо позначати 1-, 2-, 3-, …-півходові позиції.

З правил гри випливає, що 0 — єдина позиція-кінець. Ось граф, де вона позначена буквою F, решта позначені як «поки що невідомо».

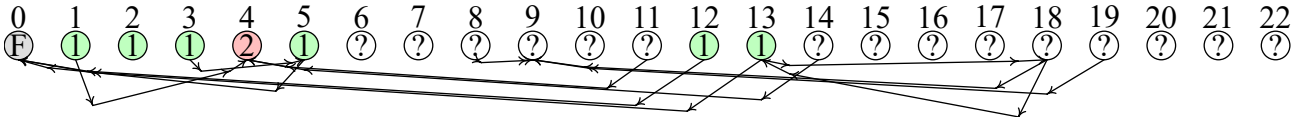


Позначимо 1-півходовими позиції, з яких можна за один хід одного гравця (один півхід) потрапити в єдину позицію-кінець 0; це позиції 1, 2 та 3 (з них є традиційні ходи в 0), а також позиції 5, 12 та 13 (з них є спеціальні ходи в 0).

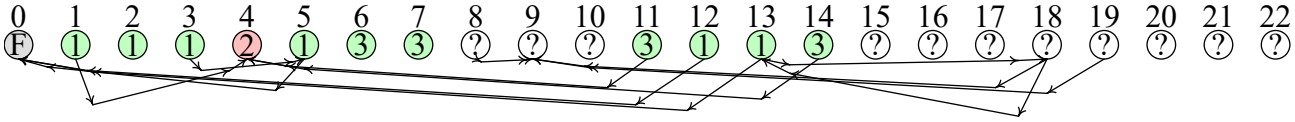
(Як бачимо, 1-півходовими виявляються, зокрема, позиції 3 і 5, при тому, що є хід з 3 у 5. Але це неважливо; головне, що з кожної з них є хід у 0. Більш того, це не єдина така ситуація, є ще чимало пар 1-півходових позицій, між якими є традиційні (а не спеціальні) ходи: з 2 в 1, з 3 в 1, з 3 у 2, з 5 у 2, з 5 у 3, з 13 у 12.)



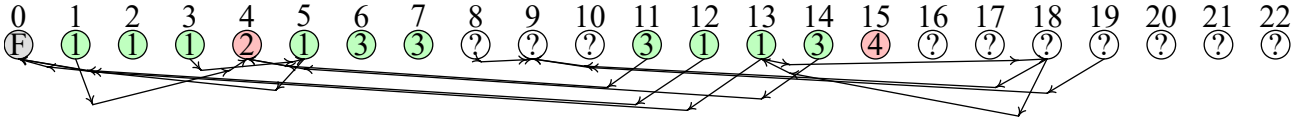
Позначимо 2-півходовими позиції, з яких усі (пів)ходи ведуть лише до 1-півходових. Такою виявляється лише одна позиція 4: з неї є ходи лише в 1, у 2 та у 3. Позиції 6, 7, 8, 14, 15, 16 та 18 не позначаємо, бо з них лише деякі з (пів)ходів ведуть до як(ої/ихо)сь із 1-півходових, але є також і (пів)ходи у позиції, поки що ніяк не позначені. Ще не відомо, де ті інші (пів)ходи дозволять виграти, де лише завести гру в цикл і тим уникнути програшу, а де всього лиш відтягти програш на пізніший момент. Але кожна з цих альтернатив краща, чим іти туди, звідки суперник може виграти за один (пів)хід.



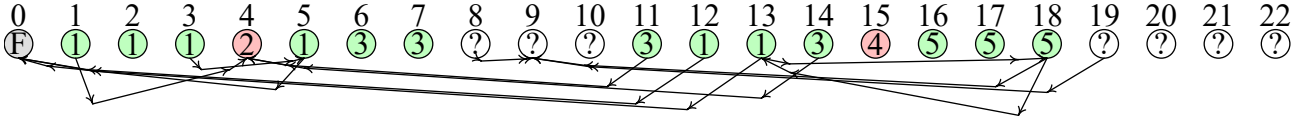
Позначимо 3-півходовими позиції, з яких можна за один (пів)хід перейти до 2-півходової. Це позиції 6, 7, 11 та 14. Позицію 1 при цьому пропускаємо, бо вона ще раніше оголошена 1-півходовою, й невигідно відкладати виграш на 3 півходи, коли можна виграти одним півходом.



Позначимо 4-півходовими позиції, з яких усі (пів)ходи ведуть лише до 3-півходових та/або 1-півходових, і хоч один з них до 3-півходової. Такою знову виявляється лише одна позиція, але не 8 (як можна помилково подумати, дивлячись послідовно зліва направо), а 15: з неї є ходи лише в 12, у 13 та в 14, де 14 є 3-півходовою, решта 1-півходовими. А позицію 8 (так само, як і 9, 10, 16 та 17) не позначаємо, бо з кожної з них, крім (пів)ходу до як(ої/ихо)сь із 3-півходових, є також хоча б один (пів)хід до позиції, яка не є ні 3-півходовою, ні 1-півходовою.

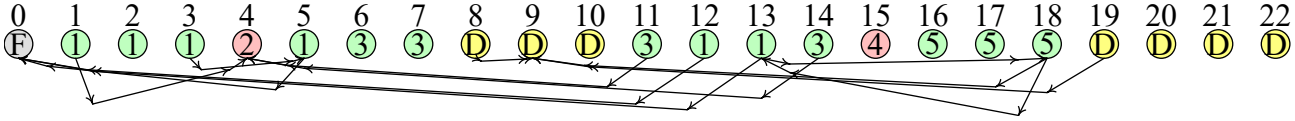


Позначимо 5-півходовими позиції 16, 17 та 18, з яких є (пів)хід до 4-півходової.



Спроба позначити 6-півходові позиції завершується тим, що нічого не позначено. Зокрема, 19 не позначена, бо, крім (пів)ходів до 5-півходових позицій 16, 17 та 18, є ще (пів)хід до досі не позначеної позиції 9.

Це є причиною завершити основний цикл, і оголосити, що з кожної з досі не позначених позицій 8, 9, 10, 19, 20, 21 та 22 гра буде зациклюватися. Звісно, за умов, що для обох гравців краще нескінче́нний цикл, чим програш, і що гравці не помиляться при намаганні дотриматися цього.



Так виходить, що в цьому прикладі є чимало ходів «назад» (у бік збільшення кількості паличок), але при цьому той нескінче́нний цикл, який краще, чим програш, забезпечується лише зацикленими

переходами в обидва боки між позиціями 8 і 9. Звісно, так буде далеко не завжди, але ось тут виявилося так…

А як щодо дуже великих кількостей паличок? Поки що в нас була сама́ гра Баше (в якій обчислювати `n%4` цілком можна для дуже великих `n`), її одне узагальнення в розд. 1.2.4 (в якому перехід до дуже великих кількостей паличок був складнішим, бо цикл доводилося шукати досить незручним способом, але можливим), та інше узагальнення в розд. 1.2.5 (в якому са́ме циклів не було взагалі, але кінець кінцем був заявлений без доведення зв'язок із «фібоначчієвою системою числення», що знов-таки дає можливість відповідати на питання про (ви/про)грашність позицій для вельми великих кількостей паличок).

Тож: *чи існує спосіб ефективно розв'язати цю задачу «Баше з додатковими ходами і циклами», але для значно більших кількостей паличок?* Відповідь залежить від того, (чи/як) буде при зростанні кількості паличок зростати кількість «спеціальних ходів».

Розгляньмо модифікацію цієї задачі, де кількість спеціальних ходів лишається $0 \leqslant t \leqslant 12345$, кількість позицій, для яких треба вивести W/L/D, лишається $1 \leqslant k \leqslant 12345$, але значення тепер бути значно більшими, в межах $1 \leqslant a_i \leqslant 10^{18}$, $0 \leqslant b_i \leqslant 10^{18}$, $1 \leqslant N_i \leqslant 10^{18}$. Замість 10^5, що для сучасного комп'ютера небагато. Було. А тепер 10^{18}. Що нереально багато, якщо хотіти будувати масив від 0 до N.

У нас вже є програма-розв'язок основної версії цієї задачі, тож використаємо її для моделювання більшої. Візьмемо замість 10^{18}, наприклад, 100 (це мало для комп'ютера, але не для людини) і 5 спеціальних ходів $12 \to 4, 41 \to 42, 67 \to 25$, $68 \to 25$ та $69 \to 25$, і подивимося на (ви/про)грашність усіх позицій від 0 до 100. Маємо:

```
LWWWLWWWLWWWWLWWWLWWWLWWWLWWWLWWWLWWWLWWWDDDDDDDDDDDDDDDDDDDDDDDDDDWWWLWWWLWWWLWWWLWWWLWWWLWWWLWW
```

Тобто, на проміжках, де нема спеціальних ходів, маємо один з двох повторюваних шаблонів: або багато разів повторюється лише `LWWW`, або лише `D`. Всі відхилення від цього (наприклад, чотири підряд `W` замість трьох, зміна з повторів `LWWW` на повтори `D`, тощо) спричинені якраз спеціальними ходами.

«Заднім умом» (коли на це вже вказали) воно й очевидно. Коли в деякому проміжку нема спеціальних ходів і є деяка `L`, наступні три будуть `W` (бо є хід в цю `L`), наступна одна знову `L` (бо всі ходи в щойно згадані `W`), і так далі — маємо повтори `LWWW`. Коли десь нема можливості піти у програшну позиції, але є можливість піти в «нічийну» (цикл) — вигідніше йти у цикл, на трьох наступних (раз це проміжок, де нема спеціальних ходів, і щойно не було способу піти у програшну) найвигідніше йти в цю нічийну, й далі повторюється, що найвигідніше йти в якусь нічийну — маємо повтори `D`.

Після чого можна заявити: от є в нас деякі a_i та b_i, пов'язані зі спеціальними ходами. Відсортуємо їх. Одним списком чисел, куди включимо як a_i, так і b_i, а ще долучимо до них число 0 (бо воно має спеціальний смисл кінця гри й єдиної позиції-кінця) та $\max(\max N_i,\ \max a_i,\ \max b_i)$ (бо ще більші значення нас точно не цікавлять). (Скажімо, для щойно розглянутого прикладу це будуть 0, 4, 12, 25 (кілька разів), 41, 42, 67, 68, 69, 100). Говорячи не про цей приклад, а взагалі, у списку буде $2t + 2$ чисел, якщо всі a_i, всі b_i, 0 та максимум всіх значень всі різні; якщо є однакові, кількість буде дещо меншою, бо повтори можна й варто поєднати в одне входження. І подивимося на різниці сусідніх (у відсортованому переліку) кількостей. З маленькими різницями (як-то $4 - 0 = 4$, $12 - 4 = 8$, $42 - 41 = 1$, $68 - 67 = 1$, $69 - 68 = 1$) нічого не робимо. Але там, де різниці помітно більші (як-то $25 - 12 = 13$, $41 - 25 = 16$, $67 - 41 = 26$, $100 - 69 = 31$) можна і варто сказати: вони значно довші, чим повторювані шаблони `LWWW` та/або `D`, їх можна й варто вкоротити, вирізавши цілу кількість цих шаблонів. Наприклад, замість «WWWLWWWLWWWLWWWLWWWLWWWLWWWLWWWLWW» лишити тільки «WWWLWWWLWW», бо початок і кінець у них все'дно однакові, а від того, що в довшому варіанті ще 12 разів повторився стандартний шаблон `LWWW`, тупо нема користі.

Це може викликати реакцію «так ми не знаємо, де там повторювані шаблони і які, бо не розв'язали задачу, а розв'язати для обмежень "до 10^{18}" не можемо, тож яка з цього користь?». Але треба трохи інакше поглянути на ситуацію, й користь з'явиться. *Не варто* намагатися «спочатку знайти конкретний повторюваний шаблон, а потім вилучати зайві його повторення». Натомість, варто ще до основного етапу (ретроаналізу) «повиділяти й повирізати такі фрагменти, які точно утворять повторення якогось із шаблонів». Неважливо, чи вилучаються «повтори самого лиш LWWW», чи «повтори самого лиш WWWL», чи «повтори самого лиш WWLW», чи «повтори самого лиш WLWW», чи «повтори самого лиш DDDD». Важливо, що нема іншого випадку, крім цих п'яти, і що в кожному з цих п'яти випадків можна вилучити шматки довжини, кратної 4, «безпечно», в тому смислі, що із результатів ретроаналізу, застосованого до послідовності з вилученими шматками, можна відновити той результат. який отримали б застосуванням ретроаналізу до початкового графу гри без вилучень шматків. (Принаймні, на рівні позначок W/L/D, забувши про ідею, що хто може виграти бажає виграти якнайшвидше, хто може лише програти бажає відтермінувати програш.)

Ось є у нас сусідні у відсортованій послідовності числа a_i та a_{i+1}, такі, що $13 \leqslant a_{i+1} - a_i \leqslant (10^{18} - 20)$. Ще раз повторю: раз a_i та a_{i+1} сусідні, то в межах $a_i + 1$, $a_i + 2$, …, $a_{i+1} - 1$ нема жодного спеціального ходу (ні початку, ні кінця). З цього проміжку, обов'язково залишмо $a_i + 1$, $a_i + 2$, $a_i + 3$ (бо з них є традиційні ходи в a_i), та $a_i + 4$ (бо довший з повторюваних шаблонів має довжину 4, то нехай він матиме місце поміститися, не зачіпаючи a_i), а також, симетрично, обов'язково залишмо $a_{i+1} - 1$, $a_{i+1} - 2$, $a_{i+1} - 3$ (бо до них є традиційні ходи з a_{i+1}), та, з аналогічних

міркувань, $a_{i+1}-4$. Ці 8 позицій (вони ж кількості паличок) залишаємо як окремі позиції, які будуть окремими елементами масиву при проведенні ретроаналізу. А між a_i+4 та $a_{i+1}-4$ (обидві межі́ не включно) є ще хоча б 4 числа́ від a_i+5 до $a_{i+1}-5$ (тепер обидві межі́ включно), й ото з цих позицій видалимо максимально можливу кратну 4 кількість позицій (якщо їх 4, або 8, або 12, … — не залишимо жодної; якщо їх 5, або 9, або 13, … — залишимо одну; якщо їх 6, або 10, або 14, … — залишимо дві; якщо їх 7, або 11, або 15, … — залишимо три).

Скажімо, для вже розглянутого прикладу вийде приблизно так:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | 21 22 23 24 25 26 27 28 29 30 31 32 | 37 38 39 40 41 42 43 44 45 46 | 63 64 65 66 67 68 69 70 71 72 73 74 75 | 96 97 98 99 100
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | 17 18 19 20 21 22 23 24 25 26 27 28 | 29 30 31 32 33 34 35 36 37 38 | 39 40 41 42 43 44 45 46 47 48 49 50 51 | 52 53 54 55 56

(червоним виділено елементи, наявні у відсортованому списку; вертикальні риски позначають місця видалень (вирізань); залишені та вирізані фрагменти виходять такі: від 0 до 16 залишені; 4 штуки 17, 18, 19, 20 вирізані; від 21 до 32 залишені; 4 штуки 33, 34, 35, 36 вирізані; від 37 до 46 залишені; 16 штук від 47 до 62 вирізані; від 63 до 75 залишені; 20 штук від 76 до 95 вирізані; від 96 до 100 залишені). В цьому прикладі зменшили кількість позицій, які потрібно розглядати, всього зі 101 до 57. Зате аналогічним чином залишити всього кілька десятків позицій при 5 спеціальних ходах можна для будь-яких цілих невід'ємних значень a_i, b_i, аби поміщалися в хоч який-небудь цілочисельний тип.

Легко бачити, що при $t\leqslant 12345$ після таких вирізань може лишитися десь приблизно до 250 тис позицій; це для комп'ютера адекватний розмір.

Так, ця ідея розказана без доведень. І я навіть не певен, чи вирізати справді варто лише для сусідів $a_{i+1}-a_i\geqslant 13$, чи можна й при ще трохи менших відстанях. І потрібно ще помарудитися, щоб усе це реалізувати. (Причому, це ми ще навіть не обговорили, що робити з тим, що N_i задані в початковому, а не зменшеному, вигляді. Кажучи коротко: запам'ятовувати, що ми навирізали, так, щоб можна було потім перетворювати початкові номери у зменшені чи то бінарним пошуком, чи то `SortedMap`-ом з пошуком найближчих.) Але ідея ось така, й вона працює.

Задача 1.6:B. «Ендшпіль (шахи)»

Нагадаємо деякі істотні для цієї задачі стандартні правила гри в шахи. Грають два гравці, один грає білими, інший чорними. Гра відбувається на шахівниці, тобто дошці 8×8, стовпчики позначаються буквами від "a" до "h" зліва направо, рядки — цифрами від 1 до 8 знизу догори. Кожна клітинка дошки або порожня, або містить одну фігуру. Якщо фігура A (не пішак) може походити згідно з правилами у клітинку, зайняту чужою фігурою B, то внаслідок такого ходу фігуру B б'ють, тобто знімають з дошки. Тому про всі клітинки, куди деяка фігура може походити, кажуть, що вони знаходяться «під боєм» цієї фігури.

Королю заборонено ходити у клітинки, які перебувають під боєм будь-якої чужої фігури (чи чужого пішака). Якщо один з гравців зробив такий хід, що король суперника опинився під боєм (це називають «шах»), суперник зобов'язаний відповісти таким ходом, щоб його король вже не був під боєм чужої фігури (чи чужого пішака). Якщо такого ходу не існує, це називають «мат».

Король може ходити на одну клітинку в будь-якому з 8-ми напрямків (ліворуч, праворуч, вперед, назад, в будь-якому напрямку за будь-якою діагоналлю). Ферзь може ходити в будь-якому з цих самих 8-ми напрямків на будь-яку кількість клітинок, але не перетинаючи клітинок, зайнятих фігурами (в тому числі й пішаками; байдуже, своїми чи чужими).

Нехай на шахівниці є три фігури: білий король, білий ферзь і чорний король. Зараз хід білих. За яку мінімальну кількість ходів вони гарантовано зможуть поставити мат? Чорні робитимуть усе, допустиме правилами гри, щоб уникнути мату чи відтермінувати його.

Вхідні дані. Програма має прочитати спочатку кількість тестових блоків $TEST$ ($1\leqslant TEST\leqslant$ $\leqslant 70000$), потім самі́ блоки. Кожен блок є окремим рядком, у якому записані позначки трьох клітинок, де розміщені білий король, білий ферзь і чорний король. Позначки клітинки складається із записаних разом букви вертикалі і номера горизонталі, позначки клітинок всередині рядка розділені одиничними пропусками (пробілами).

Усі задані позиції допустимі з точки зору шахових правил (зокрема, чорний король не під боєм).

Результати. Ваша програма повинна вивести для кожного тесту єдине число — мінімальну кількість ходів. Рахується лише кількість ходів білих (кількість ходів-відповідей чорних не додається).

Приклади:

Вхідні дані	Результати
2 a3 b3 a1 a3 e3 b1	1 2

Розбір задачі. Так склалося, що цю задачу й цей розбір (див. також [16, 94–97]) я писав раніше за написання цього посібника, тож тепер виходить, що цей розбір у значній мірі повторює по-іншому вже наведені на сторінках 68–71 пояснення суті ретроаналізу. Але хай вже так і буде…

Можливо, існують (а може, й ні) ефективніші розв'язки, які спираються на знання теорії шахів. Але ми розглянемо підхід, що не потребує вузькоспеціалізованих знань.

(До того ж, методи, традиційні для шахових програм, і складні́, й використовують спільно з деревом ходів евристичні (отже, не завжди точні) оцінки позицій, тож нема впевненості, чи знайдуть вони са́ме мінімальну кількість ходів.)

Закодуємо позиції 19-бітовими числами: один біт — чий хід (чорних чи білих), решта $3 \cdot 2 \cdot 3 = 18$ — координати на дошці кожної з трьох фігур (бо три фігури, а розмір шахівниці становить два виміри по 8 клітинок, де якраз $\log_2 8 = 3$). Таких кодів, включаючи і коди неможливих позицій (зокрема, більше однієї фігури в одній клітинці), 2^{19} (приблизно пів мільйона), що для комп'ютера відносно небагато.

(Те, що розмір шахівниці виявився степенем двійки — вельми випадковий збіг; якби було не так — стало б неможливо виділяти окремі біти, відповідні за розміщення окремих фігур, але це принесло б лише технічні зміни, а не принципові. Якби, наприклад, розміри шахівниці були 10×9 — було б 10×9 варіантів розміщення однієї фігури, $(10 \times 9) \times (10 \times 9) \times (10 \times 9)$ варіантів розміщення трьох фігур, і ще множник 2, щоб позначати, чий хід, загальна кількість $(10 \times 9)^3 \times 2 = 1\,458\,000$.)

Розсортуємо їх за трьома групами: мати (хід чорних, король чорних під боєм, ходити нікуди), неможливі (більше однієї фігури в одній клітинці; королі на сусідніх клітинках; хід білих, а чорний король під боєм білого ферзя) і «решта позицій» (їхні статуси поки що невідомі, для більшості пізніше будуть уточнені).

(Ми тут спираємося також і на те, що самим лише королем неможливо поставити мат супернику; якби задача не мала настільки очевидних причин, чому виграти може лише одна зі сторін — варто було б починати аналогічний процес і від усіх можливих виграшів білих, і від усіх можливих виграшів чорних. Мабуть, від виграшів білих окремо, від виграшів чорних окремо, але це вже деталі, які можна робити по-різному.)

Переглянувши перелік усіх позицій-матів, знайдемо всі *1-півходові* позиції, тобто такі, з яких за один півхід (білих) *можна* прийти у мат. Потім, знаючи про кожну позицію, чи є вона 1-півходовою, переглянемо «решту» позицій і виберемо з них *2-півходові*, тобто позиції, з яких *будь-який* дозволений правилами хід (чорних) веде в 1-півходову. Таким чином, якщо позиція 2-півходова, то після двох півходів (спочатку чорних, потім білих) чорним поставлять мат (при правильній грі білих, для будь-якої допустимої гри чорних).

Далі повторимо аналогічні дії з невеликою відмінністю. Переглянувши перелік усіх 2-півходових позицій, знайдемо всі *3-півходові* позиції, тобто такі, з яких за один півхід (білих) можна прийти у 2-півходову, але при цьому 1-півходові позиції слід пропускати — якщо білі можуть поставити мат за один півхід, нема чого розтягувати гру на три півходи. Потім, переглянемо «решту» позицій і виберемо з них *4-півходові*, тобто такі, будь-який дозволений правилами хід (чорних) з яких веде або в 1-, або в 3-півходову позиції, причому обов'язково існує хоча б один хід (чорних), який приводить у 3-півходову позицію. Отже, якщо позиція 4-півходова, то при правильній грі обох сторін після чотирьох півходів (чорних, білих, чорних, білих) чорним поставлять мат. Якщо білі гратимуть неправильно, мат може настати пізніше чи не настати взагалі; якщо чорні — мат може настати вже через два півходи.

Подальші дії (знаючи $2k$-півходові позиції, будуємо $(2k+1)$- та $(2k+2)$-півходові) цілком аналогічні. Процес завершується, коли чергових …-півходових позицій вже не з'являється (експериментально, 20-півходові ще є, а 21-півходових вже нема).

Умовою кінця циклу *не можна* ставити «перелік "решти" позицій став порожнім», бо порожнім він так і не стає. Зокрема, за рахунок патових позицій, де *пат* — позиція, в якій повинен ходити гравець, жодна фігура якого не може нікуди походити (не порушуючи правил гри), але, на відміну від мату, король цього гравця не під боєм. За сучасними шаховими правилами, пат є не програшем, а нічиєю. В умові задачі це не описано, але, трактуючи її суто формально, можна побачити, що такі позиції не є метою білих. Чорні ж, у рамках цієї задачі, не можуть привести гру в таку позицію.

Ще один момент, який не згаданий в умові чітко, але випливає хоч з неї, хоч з повних шахових правил — за неправильної гри, білі можуть втратити ферзя (підставивши його під бій чорного

короля у клітинці, не сусідній зі своїм королем), після чого вже не зможуть поставити мат. Це треба врахувати при відборі позицій-матів, а також при визначенні переліків ходів з кожної позиції та/або при визначенні неможливих позицій.

Щоб швидко визначати, якою є позиція (неможливою, матом, 1-півходовою, тощо) варто підтримувати масив з діапазоном індексів від 0 до $2^{19}-1$, значення якого якось кодуватимуть ситуації «неможлива», «мат», «1-півходова», «2-півходова», …, «ще не відомо».

Цей алгоритм виконує чимало дій ($\approx P \cdot D \cdot M$, де P — кількість позицій ($\approx 2^{19}$), D — максимальна кількість півходів (20), M — середня кількість ходів з позиції (≈ 10)). Але ці дії майже не залежать від кількості позицій у вхідних даних (задача розв'язується для всіх позицій, і лишається тільки бра́ти готові відповіді).

1.6.5 Чи є альтернативи ретроаналізу?

Так, є. На мою думку, вони не простіші (хоча комусь може бути й інакше), тому утримаюся від детального їх розгляду. Але посилання на одну відносно відому альтернативу наведу: [25](link).

1.7 Динамічне програмування в ациклічних послідовних іграх з числовими виграшами

Динамічне програмування (скорочено «ДП», напівскорочено «динпрог») розглядається в тому смислі, як у дисципліні «Алгоритми та структури даних»; детальніше можна бачити, зокрема, у [16, с. 50–66], тут, та в багатьох інших джерелах.

Розв'язування ігор з числовими виграшами динпрогом зазвичай відбувається приблизно так:

1. визначити позиції гри;
2. приписати кожній позиції A_i підзадачу «яку максимальну суму $p(A_i)$ може забрати собі гравець, якому дісталася ця позиція?»;
3. організувати переходи/зв'язки́ між позиціями, щоб визначати відповіді «більших» (ближчих до старту, дальших від можливих фінішів) позицій, базуючись на відповідях «менших».

Звісно, це лише дуже загальний напрям, деталі в різних ситуаціях можуть істотно відрізнятися.

1.7.1 Випадок двох гравців та сталої суми виграшів

Нехай гравців рівно двоє, і є певна стала (інакше кажучи, постійна чи константна) сума виграшу S, яка за результатами гри буде поділена між цими гравцями, й результати гри можуть відрізнятися *лише* тим, *як* ця сума буде *розділена* між ними. Також можна вважати, що всі можливі результати визначаються одним числом w, й 1-й гравець отримує виграш w, а 2-й гравець отримує виграш $S-w$. Як правило, при цьому *не* вимагають ні умови $0 \leqslant w \leqslant S$, ні умови $S \geqslant 0$, тобто виграші можуть бути від'ємними. Більш того, цей різновид задач нерідко переформульовують, вимагаючи $S=0$ й кажучи про *гру з нульовою сумою*; тоді знаки виграшів завжди (крім випадку $w=0$) протилежні. Теоретично «стала сума» й «нульова сума» рівноцінні (теоретично можна сказати, що при $S \neq 0$ відбувається гра з нульовою сумою, просто коли 1-й отримує виграш w і 2-й виграш $(-w)$, то 2-му ще додатково (незалежно від результатів гри) доплачують S).

Звісно, далеко не всі ігри-як-взаємодії-сторін зводяться са́ме до таких. Хоча б тому, що буває взаємодопомога, буває взаємна шкода, а ігри зі сталою сумою неспроможні навіть помітити відмінність між цими моделями поведінки, й тим паче непридатні для досліджень подальших деталей. Однак, ігри зі сталою сумою все ж існують, і їх можна досліджувати, усвідомлюючи їх обмеженість.

Якщо гра зі сталою сумою виявляється ще й неупередженою у смислі розд. 1.1.8 (тобто, гравці мають однакові ходи та однакові правила нарахування числових виграшів), зручно продовжити міркування, вперше згадане на початку розд. 1.2 і продовжене в розд. 1.5. Спочатку, поки йшлося

лише про виграшні й програшні позиції, гравець шукав такий «виграшний хід», який «підсуне» супернику програшну позицію; коли з'явилася ще можливість нічийних позицій — ідея змінилася на намагання «підсунути» супернику, якщо вийде, програшну позицію; якщо не виходить програшну, то, якщо вийде, нічийну; лише якщо нема ні програшних, ні нічийних, тоді «віддавати» виграшну. Тепер час для подальшого узагальнення все тієї ж ідеї: враховуючи, що сума стала і кожен гравець забирає все, що не може забрати інший, збільшення свого виграшу рівносильне зменшенню виграшу суперника, й тому ***максимальний виграш досягається на тому варіанті ходу, який «віддає» супернику позицію з мінімальним виграшем***.

(Якщо гра має сталу суму, але упереджена, застосування цієї ж ідеї «віддавати супернику позицію з мінімальним виграшем» теж може виявлятися доречним (після введення інформації «чий хід?» у позицію, приблизно аналогічно розд. 1.2.4). Але тут вже може виявитися, а може й ні, залежно від особливостей гри.)

Задача 1.7:A. «Гра на максимум суми (1/2/3)»

N карток викладені в ряд зліва направо. На кожній картці написане ціле число. Два гравці по черзі забирають карткú, причому забирати можна лише з правого боку, або 1 картку, або 2 картки, або 3 картки (але, звісно, не більше карток, чим їх є). Закінчується гра, коли забрано всі карткú (поки хоч одна картка є, гравець зобов'язаний робити один із можливих ходів). Мета гри — отримати якнайбільшу суму (чисел, записаних на забраних картках).

Яку максимальну суму гарантовано зможе набрати перший гравець?

Вхідні дані. У першому рядку вказано кількість карток N ($1 \leqslant N \leqslant 1234567$). У другому рядку через пробіли задані N цілих чисел (що не перевищують за модулем 10^3; інакше кажучи, з проміжку $[-1000; +1000]$) — значення, записані на картках.

Результати. Виведіть єдине ціле число — максимальну суму, яку гарантовано зможе набрати перший гравець.

Приклади:

Вхідні дані	Результати
7 11 11 11 11 11 11 11	44
8 3 2 999 4 6 7 -5 1	1000
9 23 -17 2 -13 5 3 11 -7 19	30

Примітки. У першому прикладі, числа додатні й однакові, тому гравцям вигідно забирати якнайбільше карток: 1-й забирає три штуки, потім 2-й забирає три штуки, потім 1-й забирає останню і цим завершує гру з сумою $(11 + 11 + 11) + (11) = 44$.

У дру́гому прикладі, значення 999 перевищує всю решту, тож варто дбати в першу чергу про те, щоб узяти це найбільше число. І перший хід «узяти дві картки 1 та (–5)» це забезпечує: як би на нього не відповів 2-й (чи забере лише 7, чи дві картки 7 і 6, чи три картки 7, 6 і 4), все'дно на наступному ході 1-й зможе взяти собі зокрема й картку 999. А інші можливі перші ходи 1-го гравця не гарантують йому взяття картки 999: якщо забрати на першому ході три картки, 2-й зможе забрати 999 наступним же ходом; якщо забрати на першому ході одну картку, 2-й у відповідь може взяти собі лише одну картку, тобто (згідно з уже наведеними міркуваннями) забезпечити, що картка 999 дістанеться йому. Звісно, для остато́чної відповіді слід ще знайти точне значення суми; це $(1 + (-5)) + (999 + 2 + 3) = 1000$, за такого ходу гри: 1-й забирає 1 та (–5); 2-й забирає 7, 6 та 4 (це найкраще, що йому лишається після втрати надії взяти 999); 1-й забирає 999, 2 і 3.

Зверніть увагу: «виграшним» виявився контрінтуїтивний перший хід. Узяти від'ємне значення (–5) (хоча його можна й не бра́ти) і не взяти після нього додатне значення 7 (хоч його й можна взяти разом з 1 і –5) на перший поверховий погляд якось ніби протирічить меті набрати якнайбільше. Однак, усе це нівелюється тим, що лише так 1-й гравець може гарантувати, що потім візьме 999.

Очевидно, що при «середніх» значеннях (і різних, і нема одного, значно більшого решти) згадані підходи не працюють, а динпрог працює.

У третьому прикладі, 30 досягається так: 1-й гравець забирає одну картку 19; 2-й гравець забирає три картки (–7), 11, 3; 1-й гравець забирає одну картку 5; 2-й гравець забирає дві картки (–13), 2; 1-й гравець забирає дві картки (–17), 23, на чому гра закінчується; $(19)+(5)+(-17+23)=30$.

Розбір задачі. Легко бачити, що гра неупереджена. Оскільки гравці забирають картки лише з правого боку, можна вважати позицією кількість карток, що лишилися.

Діючи згідно зі згаданою тут загальною ідеєю, оголосимо серію підзадач «Яку найбільшу суму $p(i)$ зможе назбирати гравець, якому дісталася позиція "лишилося i карток"?», осмислену при $0\leqslant i\leqslant N$. (Випадок $p(0)=0$ виражає «коли карток нема, то й сума=0» і є тривіальною підзадачею ДП.)

Застосувавши вищезгадане міркування для ігор зі сталою сумою «віддавати супернику позицію з мінімальним виграшем», отримаємо рівняння ДП

$$p(i)=s(i)-\min\Big(p(i-1),\underbrace{p(i-2),\,p(i-3)}_{\text{рахуємо лише існуючі}}\Big), \tag{12}$$

де $s(i)$ — сума значень перших i карток, а примітка «рахуємо лише існуючі» позначає, що при $i=1$ в аргументах min лишається тільки один варіант $p(i-1)=p(0)=0$, при $i=2$ лишаються два варіанти $p(i-1)=p(1)$ та $p(i-2)=p(0)=0$, а далі (при $i\geqslant 3$) задіяні всі три варіанти.

Щоб шукати $s(i)$, ніби й можна щоразу запускати цикл, що додає елементи від крайнього лівого до поточного; однак, це неефективно, а обмеження «$1\leqslant N\leqslant 1234567$» натякає (точніше кажучи, волає), що потрібен підхід, де $s(i)$ знаходять значно швидше. І такий підхід існує — *префіксні суми*. Це допоміжний масив, елементи якого зберігають суми елементів початкового масиву, починаючи від початку й до потрібного місця. Звісно, щоб це давало виграш, потрібно знаходити елементи масиву сум не вкладеними циклами, а на основі попередніх; зазвичай, це роблять одним з таких способів:

1-й спосіб	`s[0] = data[0];`
	`for(int i=1; i<n; i++)`
	`  s[i] = s[i-1] + data[i];`

2-й спосіб	`s[0] = 0;`
	`for(int i=0; i<n; i++)`
	`  s[i+1] = s[i] + data[i];`

Легко бачити, що вони «зсунуті» один відносно іншого (2-й містить на початку 0, відповідний сумі 0 штук елементів, тоді як 1-й починається відразу з суми одного елемента № 0).

`i`	0	1	2	3	4	5	6	7
`data[i]`	3	1	4	1	5	9	2	
`s[i]` (1-й спосіб)	3	4	8	9	14	23	25	
`s[i]` (2-й спосіб)	0	3	4	8	9	14	23	25

Який з цих варіантів вибрати — дрібне питання, яке можна вирішити як завгодно. Але від цього вибору залежить, як програмувати формулу (12).

Спираючись на все сказане, допишіть програму самостійно.

Задача 1.7:B. «Гра на максимум суми (1/2/3) — інтерактив»

(Гра ідентична попередній задачі 1.7:A.)

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Протокол взаємодії. На початку, один раз, Ваша програма повинна прочитати початкову позицію гри, яка задається у двох рядках. Перший рядок містить єдине число N ($1\leqslant N\leqslant 1000$) — кількість карток. У другому рядку через пробіли задані N цілих чисел (що не перевищують за модулем 10^3; інакше кажучи, належать проміжку $[-1000;+1000]$) — значення, записані на картках.

Потім слід повторювати такий цикл:

1. Вивести окремий рядок, що містить одне ціле число від 1 до 3 (не більше, чим поточна кількість карток) — свій хід, який позначає, скільки карток з правого боку забирає Ваша програма.

2. Якщо було забрано останню картку, вивести окремим рядком фразу "`FINISH` <*myScore*> <*yourScore*>" (без лапок; слово `FINISH` так і вивести, символ-у-символ згідно зі зразком, замість <*myScore*> повинна бути сума чисел на картках, забраних Вашою програмою, замість <*yourScore*> сума на картках, забраних програмою-суперницею) й завершити роботу.
3. Інакше, прочитати хід програми-суперниці, в такому ж форматі, як у позаминулому пункті 1. Гарантовано, що хід допустимий: виведене програмою-суперницею одне число перебуває в межах від 1 до 3 і не перевищує поточної кількості карток. Ця гарантія дійсна лише якщо Ваша програма правильно визначила, що гра ще не закінчилася.
4. Якщо було забрано останню картку, вивести фразу в абсолютно такому ж форматі, як у позаминулому пункті 2, й завершити роботу.

Все це слід повторювати, доки не будуть забрані всі картки (тобто, доки гра не завершиться). Програма-суперниця не виводить фраз "`FINISH` <*myScore*> <*yourScore*>" чи якихось аналогічних.

Приклади:

Вхід, суперник	Ваша програма
8 3 2 999 4 6 7 -5 1 3	 2 3 FINISH 1000 17
8 3 2 999 4 6 7 -5 1 1 1	 2 3 1 FINISH 1008 9

Примітки. Нібито порожні рядки між різними ходами у прикладі зроблені щоб краще було видно, хто коли ходить; вводити/виводити їх не треба.

Перший приклад є грою ідеальних суперників, які обидва завжди вибирають найкращі ходи: спочатку 1-й гравець забирає `2` картки (1 та (-5)); потім 2-й забирає `3` картки (7, 6 та 4); потім 1-й забирає `3` картки (999, 2 та 3). Протягом цієї гри 1-й гравець набрав $(1+(-5))+(999+2+3)=1000$, а 2-й гравець набрав $(7+6+4)=17$.

У дру́гому прикладі послідовність карток така са́ма, але програма-суперниця грає неправильно. Внаслідок цього, Вашій програмі вдається набрати більше: $(1+(-5))+(6+4+999)+(3)=$ $=1008$. Але Ваша програма зарані не знає, чи оптимально грає програма-суперниця, й повинна бути готова до будь-яких дозволених правилами гри варіантів розвитку подій. Також зверніть увагу, що в цьому прикладі Ваша програма повинна вивести наприкінці `FINISH 1008 9`, згідно фактичної гри (а не попередню теоретичну оцінку цих сум, яка така само `1000 17`, як і в попередньому прикладі).

Оцінювання. Оцінювання потестове (кожен тест запускається й оцінюється незалежно від результатів проходження інших тестів; оцінка за розв'язок є сумою оцінок за проходження тестів).

Вашій програмі треба буде грати як з ідеальними програмами-суперницями, так і іншими. Вашій програмі при цьому невідомо, чи конкретно в цьому запуску йдеться про ідеального суперника, чи ні. Незалежно від того, якою є програма-суперниця, Ваша програма зобов'язана набрати суму, більшу або рівну сумі, обчисленій за правилами попередньої (не інтерактивної) задачі (якщо Ваша програма набере строго менше, оцінка за відповідний тест буде 0; якщо Ваша програма набере більше, ніяких підвищень балів за це не буде).

Решта вимог до Вашої програми: довести гру до кінця; вчасно помітити, що гра закінчилася, після чого негайно вивести фінальне повідомлення й завершити роботу; правильно порахувати, хто яку суму набрав (фактично).

Розбір задачі. Майже все вже сказано в розборі неінтерактивної версії 1.7:A та розборах інших інтерактивних задач. Але наголошу, що зáраз здається особливо доречним використати згаданий наприкінці розбору задачі 1.2:E (або в [16, с. 54–55], або ще в багатьох джерелах) масив виборів, де для кожної підзадачі-позиції зберігатиметься готова відповідь, скільки (1, 2 чи 3) карток найвигідніше забирати, потрапивши в цю позицію. Динпрог знаходить оптимальні значення (отже, може знаходити й вибори) для всіх підзадач, а не лише тих, які відповідають ідеальній грі обох гравців (див. також [16, с. 57]).

Втім, це не позбавляє від необхідності рахувати суми, фактично набрані гравцями.

(При великому бажанні, можна обійтися й без масиву виборів, якщо є масив із відповідями $p(i)$ (при $0 \leqslant i \leqslant N$) згідно (12): можна на кожному ході заново визначати, який з трьох (чи, якщо карток вже мало, меншої кількості) варіантів дає мінімум у (12). Але це складно й мутно, а на чи не єдиний аргумент на користь такого способу «а раптом пам'яті дуже мало, не вистачає на масив виборів?» можна відповісти «Тоді краще зекономити пам'ять, не тримаючи вхідні дані в пам'яті, а читаючи й відразу обробляючи, замінивши масив s префіксних сум на всього 1–2 наразі останніх елементи, масив p головних відповідей (12) на всього 3–4 наразі останніх елементи, а масив виборів усе-таки тримати повністю». Тобто, безпосередньо при $N \leqslant 1000$ обсяг пам'яті взагалі не є проблемою; мова про те, що *якби* умову задачі *змінили* так, щоб обсяг пам'яті став би вузьким місцем, то, на мою думку, вигідно було б прооптимізувати купу інших речей, а масив виборів зберегти.

Само собою, важливо також не намагатися «спростити» все до жадібних алгоритмів замість динпрога, бо навряд чи існує жадібність, яка вміє розібратися з проблемами, окресленими у примітках до умови задачі 1.7:A.)

Задача 1.7:C. «Гра на максимум суми (L/R)»

N карток викладені в ряд зліва направо. На кожній картці написане ціле число. Два гравці по черзі забирають по одній картці, причому забирати можна або крайню ліву, або крайню праву. Закінчується гра, коли забрано всі картки (поки картки є, гравець зобов'язаний робити один із можливих ходів). Мета гри — отримати якомога більшу суму (чисел, записаних на забраних картках).

Яку максимальну суму гарантовано зможе набрати перший гравець?

Вхідні дані. У першому рядку вказано кількість карток N ($1 \leqslant N \leqslant 2013$). У другому рядку через пробіли задані N цілих чисел (що не перевищують за модулем 10^3; інакше кажучи, належать проміжку $[-1000; +1000]$) — значення, записані на картках.

Результати. Виведіть єдине ціле число — максимальну суму, яку гарантовано зможе набрати перший гравець.

Приклад:

Вхідні дані	Результати
4 1 2 9 3	10

Примітка. Якщо на першому ході забрати 1, то суперник у відповідь буде змушений забрати або 3, або 2; у будь-якому з цих випадків, перший гравець зможе забрати собі 9, і, таким чином, гарантовано отримати суму 10 (після чого другий гравець забирає останню картку, і гра закінчується).

Якби перший гравець на першому ходу забирав 3, то другий міг би у відповідь забрати 9, і в результаті перший отримав би лише $3+2=5$. При великій дурості, другий гравець міг би відповісти на хід "3" і ходом "1"; у цьому випадку перший гравець міг би отримати суму $3+9=12$. Але перший гравець не може гарантувати, що другий зробить такий дурний хід, тому й відповідь не 12, а 10.

Розбір задачі. Тут деяка частина переноситься з двох попередніх задач 1.7:A–1.7:B абсолютно без змін, дещо зазнає вельми істотних змін, але тісний зв'язок задач все'дно очевидний.

Оскільки тут картки можна забирати і зліва, і справа (за один хід — «або …, або …», але за кілька ходів це стає «і …, і …»), не виходить зручно задати позицію одним числом, лише двома. Які це будуть 2 числа — можливі варіанти. Як мінімум, такі:

(а) (i,j)-підзадача означає «залишилися всі картки з номерами від i до j, обидві межі включно», осмислена при $0 \leqslant i \leqslant j \leqslant N-1$, випадок «карток не лишилося» або подавати окремо, або вважати таким випадком ситуацію $i > j$;

(б) (i,j)-підзадача означає «залишилися всі картки з номерами від i включно до j не включно», осмислена при $0 \leqslant i \leqslant j \leqslant N$;

(в) (i,k)-підзадача означає «лишилися картки, починаючи з номера i, всього k штук», осмислена при $0\leqslant i\leqslant N-1,\ \ 0\leqslant k\leqslant N-i$;

(г) (k,i)-підзадача означає «залишилося k карток, починаючи з номера i», осмислена при $0\leqslant k\leqslant N$, $0\leqslant i\leqslant N-k$.

Вибрати з цих (чи, можливо, ще якихось аналогічних) варіантів можна, в принципі, будь-який. Особисто мені здається відносно зручнішим варіант (г).

(Багато хто вибрав би варіант (а) як «традиційний», бо ≈так були вперше сформульовані рівняння динпрога для задачі мінімальної триангуляції опуклого многокутника. Та задача *не* має глибокого зв'язку з цією; *єдине* спільне — дрібніші підзадачі є проміжками (діапазонами) більшої, тобто вкладена підзадача відрізняється від зовнішньої тим, що могли забрати частину елементів хоч лише зліва, хоч лише справа, хоч з обох боків, але не зсере́дини. Оскільки навіть така схожість може бути корисною, наведу для охочих посилання [26] та [27, с. 5–11] (на жаль, ні першоджере́ла, ні книги, де мін. триангуляція пояснена детальніше, *не* є легко&офіційно доступними через Інтернет). Але абсолютно не наполягаю на їх вивченні.

Чому я вибрав серію підзадач всупереч традиції? В (а) є згадана наприкінці [27] проблема «а в якому порядку розв'язувати такі підзадачі?», а у (г) порядок розв'язання підзадач якнайприроднішій. (Більш того: і в [26], і в [27] спочатку ставлять серію підзадач за варіантом (а), а потім розв'язують підзадачі в порядку, найприроднішому якраз для (г)!). Ще, варіант (г) дозволяє трохи зекономити пам'ять, виділяючи в зубчастому масиві рядки різних довжин.)

Тобто, серія підзадач — «Яку максимальну суму $p(k,i)$ може гарантовано набрати гравець, якому дісталася позиція "лишилося k карток, починаючи з i-ї"?».

Тривіальними можна зробити, наприклад, підзадачі «лишилася 1 картка» $p(1,i)=c(i)$. (Або, наприклад, підзадачі «нічого не лишилося» $p(0,i)=0$.) Рівнянням динпрога можна зробити, наприклад,

$$p(k,i)=S(\underbrace{i,\ \ldots}_{k\text{ штук}})-\min\left(\left\{\begin{array}{c}p(k{-}1,i),\\ p(k{-}1,i{+}1)\end{array}\right\}\right), \tag{13}$$

де "$S(i,\ \ldots)$ (k штук)" є сумою k записаних на картках чисел $c_i\ +\ c_{i+1}+\ldots\ +\ c_{i+k-1}$; у термінах запровадженої після формули (12) префіксної суми, це буде (залежно від того, який з двох варіантів префіксної суми вибрали) або $s(i+k)-s(i)$, або $s(i+k-1)-\big(\texttt{i==0 ? 0 : }s(i-1)\big)$. Аргументація, чому з суми значень всіх карток слід віднімати мінімум, та ж, що й раніше: «віддавати супернику позицію з мінімальним виграшем». Мінімум вибирається са́ме з цих варіантів, бо при забиранні однієї картки завжди відбувається перехід від k до $k-1$, але можуть бути варіанти «забрати зліва» (тоді ліва межа діапазону зсувається й маємо меншу підзадачу $p(k-1,\ i+1)$) та «забрати справа» (тоді права межа діапазону зсувається, а ліва нерухома, й маємо меншу підзадачу $p(k-1,\ i)$).

Перші альтернативні розв'язки задач 1.7:C та 1.7:A. Рівняння (13) можна замінити, наприклад, на

$$p(k,i)=\max\left\{\begin{array}{l}c(i)+\min\left\{\begin{array}{l}p(k-2,i+2),\\ p(k-2,i+1)\end{array}\right\},\\ c(i{+}k{-}1)+\min\left\{\begin{array}{l}p(k{-}2,i{+}1),\\ p(k-2,\ i),\end{array}\right\}\end{array}\right. \tag{14}$$

Це рівняння (на відміну від усіх досі розглянутих) охоплює «повний хід», тобто «півходи» обох сторін. (До речі, через це тепер не можна вважати тривіальними самі́ лише підзадачі $(1,i)$, бо тепер можна «перескочити через них»; найпростіший спосіб це врахувати — і залишити відомі тривіальні $p(1,i)=c(i)$, і долучити до них нові $p(0,i)=0$.)

Спочатку розберемося, звідки в (14) варіанти $p(k-2,\ i+2)$, $p(k-2,\ i+1)$ та $p(k-2,\ i)$. Вони відповідають випадкам «один гравець забрав картку зліва, інший теж зліва» (найлівішою стає $i+2$); «один гравець забрав картку зліва, а інший справа» (найлівішою стає $i+1$; згадується двічі: спочатку як «поточний гравець забрав зліва, а його суперник справа», потім як «поточний гравець забрав справа, а його суперник зліва»); «один гравець забрав картку справа й інший теж справа» (найлівішою карткою лишається i).

Тепер, звідки в (14) і max, і min: перший «півхід» робить гравець, якому дісталася позиція $(k,\ i)$, і він зацікавлений у збільшенні свого виграшу $p(k,i)$, а наступний «півхід» робить його суперник;

оскільки гра все ще має сталу суму, де збільшення одного виграшу зменшує інший, цей суперник зацікавлений у зменшенні виграшу $p(k,i)$ свого суперника (для якого ми й проводимо аналіз).

Тепер, звідки в (14) доданки $c(i)$ та $c(i+k-1)$ й чому лише вони: це варіанти, з яких вибирає гравець, якому дісталася позиція $(k,\ i)$, і вибрана картка ($c(i)$, якщо ліва, чи $c(i+k-1)$, якщо права) додається до його виграшу. Інших доданків $c(\dots)$ нема, бо інші картки або взагалі не забирають са́ме на цьому «повному ході обох гравців», або їх забирає не той гравець, для якого рахуємо $p(k,i)$.

Побудований за аналогічним принципом альтернативний розв'язок задачі 1.7:A може бути, наприклад таким. Залишимо ту саму серію підзадач «Яку найбільшу суму $p(i)$ зможе назбирати гравець, якому дісталася позиція "лишилося i карток"?» та ту са́му єдину тривіальну підзадачу $p(0)=0$, а основне рівняння тоді набуде вигляду

$$p(i)=\max_{1\leqslant k\leqslant 3}\left\{\left(\sum_{j=i-k}^{i-1}c_j\right)+\min_{1\leqslant q\leqslant 3}\{p(i-k-q+1)\}\right\} \tag{15}$$

(лише існуючі варіанти).

Це вважаючи, що в $p(i)$ аргумент i є кількістю карток, тому $0\leqslant i\leqslant N$ (включно), але c_j є значенням j-ї при нумерації з 0 картки, тому $0\leqslant j<N$ (не включно). Примітка («лише існуючі варіанти») виражає, що діапазони $1\leqslant k\leqslant 3$ та $1\leqslant q\leqslant 3$ (де k виражає, скільки карток бере поточний гравець, q — скільки його суперник) при $i\leqslant 5$ *можуть* звужуватися (варіанти пропускатися), якщо поточної кількості карток не вистачає для відповідного варіанту. Однак, для всіх $i\geqslant 1$ хоча б один спосіб існує, тому, маючи в реалізації (15) відповідні перевірки на k та q, досить лише однієї тривіальної підзадачі $p(0)=0$.

(При бажанні можна переписати (15), «розгорнувши» як

$$p(i)=\max\left\{\begin{array}{l} c(i-1)+\\ \qquad \min\{p(i-1),\ p(i-2),\ p(i-3)\},\\ c(i-1)+c(i-2)+\\ \qquad \min\{p(i-2),\ p(i-3),\ p(i-4)\},\\ c(i-1)+c(i-2)+c(i-3)+\\ \qquad \min\{p(i-3),\ p(i-4),\ p(i-5)\} \end{array}\right\} \tag{15*}$$

(лише існуючі варіанти);

усі примітки при цьому лишаються в силі.)

Однак, (14) громіздкіше, чим (13); при порівнянні хоч (15), хоч (15*) з (12) це ще гостріше; а великих переваг хоч (14) над (13), хоч (15) чи (15*) над (12) якось ніби не видно.

(Я згоден вважати перевагою, що «(14)–(15*) можна запрограмувати, не знаючи часткових сум». Але я не згоден, що така перевага важлива й варта таких нагромаджень.)

Однак, у переході від (13) до (14) чи від (12) до (15) є інша приємна складова — спроба відійти від ідеї «вся сума мінус мінімум того, що дістанеться супернику». Просто, щоб користь від цього стала помітнішою, можна й варто діяти рішучіше.

(Що було «нерішучим»? Тут все ще використовують зацікавленість у зменшенні виграшу свого суперника.)

1.7.2 Відходимо від ідеї «всі сили на підсовування супернику якнайгіршої для нього позиції»

Мета́ — ***перейти до покращень свого виграшу, не пов'язуючи їх зі зменшеннями виграшу суперника***. Мова ні в якому разі *не* про «гравець починає дбати про чужі інтереси замість своїх» чи ще якісь бздури. В аксіомах теорії ігор як говорилося «Кожен гравець бажає виграти і прикладає (в межах, дозволених правилами гри) всіх зусиль для збільшення свого виграшу» та «Покращення результатів окремого гравця цілком може відбуватися за рахунок погіршення результатів іншого гравця», так і продовжує говоритися те са́ме. Мова про те, що коли сума виграшів *не* є сталою, то «дбати про свої інтереси» *не завжди пов'язане* з «робити супернику якнайгірше».

Нехай є деяка гра (не будемо уточнювати її правила, просто деяка), і в деякій її позиції в одного з гравців є три варіанти ходу, після будь-якого з них гра закінчується, а від того, який хід зробить цей гравець, залежать виграші обох сторін, і вони такі: або (–100; –200), або (+5; +40), або (+8; –2), де перше число — виграш того, хто робить цей хід, дру́ге — виграш його суперника. Зосередження на покращенні свого виграшу означає, що слід вибирати той хід, який дає виграші (+8; –2), бо $+8 = \max(-100,\ +5,\ +8)$. Гравець, зацікавлений у максимізації свого виграшу, не вибере (+5; +40) замість (+8; –2), хоч це й означає, що заради збільшення свого виграшу всього на $8-5=3$ він зменшує виграш суперника аж на $40-(-2)=42$. Але, водночас, такий гравець не буде й вибирати (–100; –200) просто лише тому, що так супернику дістанеться якнайгірший результат (–200). Вибір кращого для себе — це *лише* вибір кращого для себе, він не гарантує ні того, що це буде добре для суперника, ні того, що це буде для нього погано. Все залежить від правил гри.

Дру́гі альтернативні розв'язки задач 1.7:C та 1.7:A. Якого вигляду набуде ідея «дбати лише про свої інтереси», якщо застосувати її, для початку, до задачі 1.7:C? Один з варіантів такий: нехай позиції ті са́мі (k,i) «лишилося k карток, починаючи з i-ї», але для кожної такої позиції знаходимо пару (з програмістської точки зору — кортеж/структуру/клас з двох полів) (a,b), де $p(k,i).a$ виражає, яку суму набере при правильній грі обох гравців той, кому дісталася позиція (k,i), а $p(k,i).b$ виражає, яку суму набере при цьому його суперник. Тоді можна взяти тривіальні підзадачі $p(1,i)=(c(i),0)$, тобто «кому дісталася позиція "одна картка номер i", той отримує $c(i)$, а його суперник отримує 0», а рівняння динпрогу може бути таким:

$$p(k,i)=\boxed{\begin{array}{l} p1=c(i)+p(k-1,i+1).b; \\ p2=c(i{+}k{-}1)+p(k-1,i).b; \\ \texttt{if}\ (p1\geqslant p2) \\ \quad \texttt{return}\ \ (p1,\ \ p(k-1,i+1).a) \\ \texttt{else} \\ \quad \texttt{return}\ \ (p2,\ \ p(k-1,i).a) \end{array}} \tag{16}$$

(Це важко записати математично, тож вжито суміш математичної й програмістської нотацій; але це неважливо: ми й раніше не досліджували динпрог математично, а програмували).

Як і в (13), розглядаються два варіанти: або гравець бере собі картку зліва, або справа. Якщо зліва, то йому дістається сама́ картка $c(i)$ плюс оптимальна сума подальших ходів; якщо справа, то йому дістається сама́ картка $c(i{+}k{-}1)$ плюс оптимальна сума подальших ходів. Це очевидно.

Менш очевидно, чому в тексті словесних пояснень спочатку $p(k,i).a$, потім $p(k,i).b$, а в `return`-ах (16) спочатку $p1$ чи $p2$, отримані з $p(\ldots).b$, потім $p(\ldots).a$. Але це теж правильно, са́ме в тому смислі, що $p1$ чи $p2$, отримане з $p(\ldots).b$, йде у $p(k,i).a$, а $p(\ldots).a$ йде у $p(k,i).b$. «Навхрест»: b в a, a в b. Коли «наш» гравець робить хід з позиції (k,i), то *його супернику* дістається або $(k{-}1,i)$, або $(k{-}1,i{+}1)$. Рішення в тій позиції прийматиме суперник. Виграш, який суперник рахуватиме, як свій, для «нашого» гравця є виграшем суперника. А виграш, який суперник рахуватиме як виграш свого суперника, є виграшем «нашого» гравця. Са́ме тому й треба щокроку міняти місцями $p(\ldots).a$ з $p(\ldots).b$.

((1) Мені не подобається формулювання в термінах «"нашого" гравця», воно може схилити до хибної думки, ніби ми якось підсуджуємо тому гравцеві замість дивитися на гру безсторонньо. Але ніякого підсуджування тут нема, результати роботи алгоритму максимально безсторонні, я просто не зумів знайти кращого формулювання.

(2) Відповідь нової підзадачі $p(k,i)$ частину $p(k,i).a$ формує на основі змінених $p(\ldots).b$ (бо це хід того гравця, який якраз за́раз забирає картку й тому змінює свій виграш відносно ранішого $p(\ldots).b$), а частина $p(k,i).b$ береться як незмінна раніша $p(\ldots).a$. Принаймні, це так і в задачах 1.7:A–1.7:C, і взагалі є поширеним підходом до побудови аналогічних рівнянь динпрогу щодо багатьох інших задач; однак, якщо правила набирання сум виграшів будуть якісь дуже вже інші, ця властивість може й порушитися.)

Якого вигляду набуде ідея «дбати про свої інтереси»*, якщо застосувати її до задачі 1.7:A?* Один з варіантів такий: нехай позиції ті са́мі «лишилось i карток, де $0\leqslant i\leqslant N$ (включно)», для кожної такої позиції знаходимо пару (a,b), де $p(i).a$ виражає, яку суму набере при правильній грі обох гравців

той, кому дісталася позиція «лишилось i карток» а $p(i).b$ виражає, яку суму набере при цьому його суперник.

Тоді можна мати єдину тривіальну підзадачу $p(0) = (0,\ 0)$, а вибір динпрогом оптимального варіанту може набути такого, наприклад, вигляду:

```
for (int i=1; i<=n; i++) {
  int sumC = c[i-1];
  int pk = sumC + p[i-1].b;
  p[i] = (pk,  p[i-1].a);
  for (int k=2; k<=3 && i-k>=0; k++) {
    sumC += c[i-k];
    pk = sumC + p[i-k].b;
    if (pk > p[i].a)
      p[i] = (pk,  p[i-k].a);
  }
}
```
(17)

(Від аргументації утримаюся, бо в ній нема нічого дуже нового, а акуратний розгляд усіх етапів досить громіздкий. Охочі можуть провести цю аргументацію самостійно. Уточню лише, що в коді (17) мається на увазі, що масив `p` має діапазон індексів від 0 (оцінка для позиції «жодної ка́ртки») до N (оцінка для позиції «всі N карток»), а масив `c` — від 0 (значення найлівішої ка́ртки, яку забиратимуть останньою) до $N-1$ (значення найправішої ка́ртки, яку забиратимуть першою). За бажання, це можна переписати якось інакше, але в цьому коді так.)

Чи дозволяють ці модифікації розв'язати задачі 1.7:C та 1.7:A краще? Найперші розв'язки цих задач вже були правильними; тому, *не* слід хотіти ще більших набраних гравцем сум чи аналогічних покращень. І взагалі, якщо застосувати ідею «дбати лише про свої інтереси» чи то до гри зі сталою сумою, чи то до ще якоїсь гри, де властивості само́ї гри все-таки вимагають, щоб покращення результату одного гравця погіршувало результати іншого — для цих ігор отримаємо *той са́мий* результат, що й ранішим способом «віддавати супернику позицію з найменшим виграшем».

Перевага ідеї* «дбати лише про свої інтереси» *над ідеєю* «віддавати супернику позицію з найменшим виграшем» *в тому, що «дбати лише про свої інтереси» працює для ширшої сукупності ігор. У цьому смислі ця ідея краща. Водночас, вона може бути (а може й не бути) ***гіршою за якимись іншими ознаками.***

Дуже багато задач з *не* сталою сумою та/або більш чим двома гравцями (зокрема, але не тільки: 1.7:D, 1.7:F, 1.9:B), відносно неважко розв'язати аналогічно розглянутій «дру́гій модифікації», але чи то дуже важко, чи то геть неможливо розв'язати жодним з раніше розглянутих способів.

(Щоправда, в аналізі кожної з цих задач 1.7:D, 1.7:F, 1.9:B з'являються деякі додаткові проблеми, які породжують сумніви щодо цих розв'язків… Але це більше говорить про складність сами́х задач, чим про ідею «дру́гої модифікації».)

Задача 1.7:D. «Гра на максимум суми (L/R, два числа на картці) — 1»

N карток викладені в ряд зліва направо. На кожній картці написано два натуральні числа, одне вгорі й цианового (ось такого) кольору, інше внизу й фіолетового (ось такого) кольору. Два гравці по черзі забирають по одній картці, причому забирати можна або крайню ліву, або крайню праву. Закінчується гра, коли забрано всі картки, пропускати хід не можна. Мета гри — отримати якомога більшу суму, ***враховуючи, що 1-й гравець враховує й додає лише верхні цианові числа своїх карток, 2-й гравець враховує й додає лише нижні фіолетові числа своїх карток***.

Якими будуть результати гри при правильній грі обох гравців?

Вхідні дані. У першому рядку вказано кількість карток N ($1 \leqslant N \leqslant 30$). Далі йдуть N рядків, кожен з яких містить записані через пропуск (пробіл) рівно два натуральних числа, записані на відповідній картці, спочатку верхнє цинанове, потім нижнє фіолетове. Гарантовано, що всі $2N$ чисел різні, всі є цілими степенями двійки і перебувають у межах від 2^0 до 2^{60}, обидві межі включно.

Результати. Виведіть в одному рядку через пропуск два цілі числа — результати гри при правильній грі обох гравців, спочатку суму цианових чисел, яку набере 1-й гравець, потім суму фіолетових чисел, яку набере 2-й гравець.

Приклади:

Вхідні дані	Результати
4 1024 8 256 512 4 2 16 65536	1280 65538
4 1 2 8 64 1024 256 32 16	1025 80
4 1 2 8 256 1024 64 32 16	1056 258

Розбір задачі. Цей розбір неповний і трохи наївний. Цілком повний розбір цієї задачі включає і написане прямо тут, і помітну частину розд. 1.7.3. Але почнемо з трохи наївного розбору.

У першому тесті, хід правильної гри виявляється таким же, який дає «найжадібніший» алгоритм «на кожному кроці кожен гравець бере собі ту з карток, яка сама по собі дає більший виграш»: 1-й бере ліву $\frac{1024}{8}$, бо там 1024, а 1024>16; 2-й бере праву $\frac{16}{65536}$, бо там 65536, а 65536>512; 1-й бере ліву $\frac{256}{512}$, бо там 256, а 256>4; 2-й бере останню $\frac{256}{512}$, бо не має вибору. Підсумок:

1-й гравець			2-й гравець		
1024	256	$\Leftarrow \Sigma = 1280$	16	4	
8	512		65536	2	$\Leftarrow \Sigma = 65538$

Водночас, найвигідніші ходи в іграх дру́гого і третього тестів мають істотні (причому, різні) відмінності від щойно згаданого жадібного алгоритму.

У другому тесті, все досить схоже на тест з умови задачі 1.7:C: з суто локального міркування 32>1 на першому ході ніби локально вигідніше взяти праву $\frac{32}{16}$; але якщо так і зробити, то 2-й гравець забере $\frac{1024}{256}$, і, щоб цього не допустити, 1-му вигідніше почати з того, щоб забрати ліву $\frac{1}{2}$. Після цього 2-й забирає $\frac{8}{64}$, 1-й отримує можливість забрати $\frac{1024}{256}$, 2-й забирає останню $\frac{32}{16}$. Підсумок:

1-й гравець			2-й гравець		
1	1024	$\Leftarrow \Sigma = 1025$	8	32	
2	256		64	16	$\Leftarrow \Sigma = 80$

(2-й міг би й забрати $\frac{32}{16}$ на своєму першому ході, але це змінило б *лише* порядок, а набори забраних гравцями карток (а отже, й суми) не змінилися б).

Якщо ж розглянути третій тест, то для цих вхідних даних з «правильної гри обох гравців» випливає, що тепер 1-й може безболісно забирати $\frac{32}{16}$ на своєму першому ході, бо 2-му невигідно відповідати на це забиранням $\frac{1024}{64}$: якщо він це зробить, то втратить цінну для нього картку $\frac{8}{256}$ (яка 1-му менш цінна, чим $\frac{1024}{64}$, але цінніша, чим $\frac{1}{2}$). Тому, 2-й гравець на першому своєму ході забере $\frac{1}{2}$, потім 1-й забере $\frac{1024}{64}$, потім 2-й останню $\frac{8}{256}$. Підсумок:

1-й гравець			2-й гравець		
32	1024	$\Leftarrow \Sigma = 1056$	1	8	
16	64		2	256	$\Leftarrow \Sigma = 258$

Як бачимо, гра упереджена (див. розд. 1.1.8), причому не за ознакою «різні ходи» (вони якраз однакові: кожен щоразу може забирати або крайню ліву, або крайню праву картку), а за ознакою «різні функції виграшу». Тому здається доречним, як і в розд. 1.2.4, включити до позиції «хто ходить?», що в цій задачі означає «розглядати позиції, залежні від трьох аргументів (k, i, p) (лишилося k карток, крайня ліва i-а, ходить гравець № p)».

(Са́ме в цій грі на кожному кроці забирається рівно одна картка; тому, са́ме в цій грі з позицій, де $(N-k)$ па́рне, завжди ходитиме лише 1-й гравець, а з позицій, де $(N-k)$ непа́рне — лише 2-й. Однак, для більшості схожих ігор (зокрема, але не тільки, якщо ввести такі ж картки з двома числами, де 1-го гравця цікавить лише верхнє число, 2-го — лише нижнє, у гру із задачі 1.7:A) важливо все-таки мати позиції, що відрізняються лише ознакою «чий хід?».)

При цьому, нікуди не дівається і згадана тут ідея «для кожної позиції знаходимо пару (a,b), де $p(\dots).a$ виражає, яку суму набере при правильній грі обох гравців той, кому дісталася ця позиція, а $p(\dots).b$ виражає, яку суму набере при цьому його суперник». Це ні в якому разі *не* одне й те са́мо, що «чий хід?». Чотири питання «скільки набере 1-й, якщо за́раз ходитиме 1-й», «скільки набере 2-й, якщо за́раз ходитиме 1-й», «скільки набере 1-й, якщо за́раз ходитиме 2-й» та «скільки набере 2-й, якщо за́раз ходитиме 2-й» — це *дійсно чотири різні* питання. Навіть у цій задачі 1.7:D це все'дно чотири різні питання, правда половина з них не потрібні, раз відомо, скільки спочатку карток і що 1-й рахує суму верхніх чисел, а 2-й суму нижніх.

Як реалізацію програм, так і виведення рівняння дипрога (яке являє собою досить очевидне з урахуванням кількох попередніх абзаців узагальнення (16)) залишаємо в якості вправи, а для контролю наведемо повні таблиці динпрога для всіх трьох тестів. В кожній клітинці таблиці динпрога наведено по два різноколірні числа, бо це вищезгадані $p(\dots).a$ та $p(\dots).b$; пари, де обидва числá більші та обидва числá менші, виражають: більші — реально необхідні навіть з урахування згаданого зауваження про па́рність $(N-k)$, менші — ті, обчисленння яких завдяки цьому зауваженню не обов'язкові (але я їх понаводив, щоб можна було побачити суть відмінності на прикладах).

Для першого тесту:

хід 1-го гравця

	0	1	2	3
1	$\frac{1024}{0}$	$\frac{256}{0}$	$\frac{4}{0}$	$\frac{16}{0}$
2	$\frac{1024}{512}$	$\frac{256}{2}$	$\frac{16}{2}$	
3	$\frac{1028}{512}$	$\frac{260}{65536}$		
4	$\frac{1280}{65538}$			

хід 2-го гравця

	0	1	2	3
1	$\frac{0}{8}$	$\frac{0}{512}$	$\frac{0}{2}$	$\frac{0}{65536}$
2	$\frac{1024}{512}$	$\frac{4}{512}$	$\frac{4}{65536}$	
3	$\frac{1024}{514}$	$\frac{256}{65538}$		
4	$\frac{1028}{66048}$			

Для дру́гого тесту:

хід 1-го гравця

	0	1	2	3
1	$\frac{1}{0}$	$\frac{8}{0}$	$\frac{1024}{0}$	$\frac{32}{0}$
2	$\frac{8}{2}$	$\frac{1024}{64}$	$\frac{1024}{16}$	
3	$\frac{1025}{64}$	$\frac{40}{256}$		
4	$\frac{1025}{80}$			

хід 2-го гравця

	0	1	2	3
1	$\frac{0}{2}$	$\frac{0}{64}$	$\frac{0}{256}$	$\frac{0}{16}$
2	$\frac{1}{64}$	$\frac{8}{256}$	$\frac{32}{256}$	
3	$\frac{8}{258}$	$\frac{1024}{80}$		
4	$\frac{40}{258}$			

Для третього тесту:

хід 1-го гравця

	0	1	2	3
1	$\frac{1}{0}$	$\frac{8}{0}$	$\frac{1024}{0}$	$\frac{32}{0}$
2	$\frac{8}{2}$	$\frac{1024}{256}$	$\frac{1024}{16}$	
3	$\frac{1025}{256}$	$\frac{1056}{256}$		
4	$\frac{1056}{258}$			

хід 2-го гравця

	0	1	2	3
1	$\frac{0}{2}$	$\frac{0}{256}$	$\frac{0}{64}$	$\frac{0}{16}$
2	$\frac{1}{256}$	$\frac{1024}{256}$	$\frac{32}{64}$	
3	$\frac{1024}{258}$	$\frac{1024}{272}$		
4	$\frac{1025}{272}$			

1.7.3 Чому можливість довільних виграшів істотно все ускладнює?

Так уже склалося, що поточний розд. 1.7.3 по суті збігається з задачею 1.7:E (умовою + розбором).

Задача 1.7:E. «Гра на максимум суми (L/R, два числа на картці) — 2»

Загальні правила гри, нарахування виграшу та формат вхідних даних такі ж, як у задачі 1.7:D.

Якими будуть результати гри при правильній грі обох гравців? Яку максимальну суму гарантовано зможе набрати перший гравець?

Результати. Виведіть в першому рядку через пропуск два цілі числа — результати гри при правильній грі обох гравців, спочатку суму цианових чисел, яку набере 1-й гравець, потім суму фіолетових чисел, яку набере 2-й гравець. Потім виведіть у дру́гому рядку одне ціле число — максимальну суму, яку гарантовано зможе набрати 1-й гравець, хоч би як не грав 2-й.

Приклади:

Вхідні дані	Результати
4 1024 8 256 512 4 2 16 65536	1280 65538 1040
4 1 2 8 64 1024 256 32 16	1025 80 1025
4 1 2 8 256 1024 64 32 16	1056 258 1025

Розбір задачі. Чому в першому й третьому тестах перше й останнє чи́сла відповіді відрізняються (1040<1280 та 1025<1056)? Хіба «правильна гра 1-го гравця» відрізняється від «здобування 1-м гравцем якнайбільшої суми, хоч би як не грав 2-й?»

Останнє питання попереднього абзацу поставлене неточно: є різниця між правильною грою ***обох*** гравців і правильною грою ***лише 1-го*** гравця, коли 2-й гравець може грати ***як завгодно***. Наприклад, говорячи про перший тест: «а раптом 2-й гравець на першому своєму кроці візьме замість правої картки (яка приносить йому 65536) ліву (яка приносить йому 512)?». Він має право так ходити в рамках правил гри. Щоправда, якщо справді так і походить, то погіршить, порівнюючи зі способом з першого рядка відповіді, *обидва* результати: і результат 1-го гравця (який йому суперник) з 1280 до 1040, і свій власний результат 2-го гравця з 65538 до 514.

Такі дії 2-го гравця ***протирічать*** аксіомі теорії ігор «кожен гравець робить усе дозволене правилами гри, щоб збільшити свій виграш». Тобто, для теорії ігор типово розраховувати, що гравці ***не*** роблять, як у попередньому абзаці, і вважати адекватним аналіз, який на це спирається. Але для повнішого розуміння того, як співвідносяться результати, які теорія ігор вважає оптимальними, з рештою можливих результатів, варто розуміти таку властивість:

- ***Поки мова йшла про ігри, де програш одного означав виграш іншого, а також ігри зі сталою сумою — неідеальність гравця була виключно його проблемою, а супернику це було навіть вигідно*** (згадайте хоч би й задачі, де треба було інтерактивно грати, «перехоплюючи ініціативу» в неграмотного суперника).
- ***Однак, у іграх з не сталою сумою неграмотність (чи інша непередбачуваність) гравця вже є як його проблемою, так і проблемою його суперник(а/ів). Коли відомо, що суперник(и) послідовно дба(є/ють) про максимізацію сво(го/їх) виграш(у/ів) — можна оцінювати, як в(ін/они) поход(и/я)ть, і враховувати це; коли таких даних про наміри нема — неможливо.***
- ***Отже, для «правильної гри обох гравців» та для «правильної гри 1-го проти непередбачуваного 2-го» варто робити різні аналізи*** (які зазвичай дають різні результати)

(Звісно, ці результати можуть виявлятися й однаковими, але хіба випадково або за рахунок особливостей конкретних значень у вхідних даних. Якщо говорити про інші ігри, то можливі й такі ігри, де сума ніби й не стала, але по суті вигідно

робити супернику якнайгірше; тоді, результати цих різних аналізів будуть однаковими завдяки особливостям правил гри, а не вхідних даних. Однак, розглядати це детальніше не будемо.)

Тобто, в розборі попередньої задачі 1.7:E описано аналіз, адекватний, якщо обидва суперники дбають лише про свій виграш, не намагаючись ні підіграти супернику, ні зробити йому погано навіть за рахунок погіршення власного результату.

Якщо треба убезпечитися від суперника, який ігнорує свої власні виграші й зосереджений суто на завданні збитків гравцю, за якого робимо аналіз — тоді варто проігнорувати виграші суперника (він же сам їх по суті ігнорує), і застосувати, наприклад, щось схоже на згадані тут «перші альтернативні ро́зв'язки» з їхніми рівняннями, що враховують «півходи» обох сторін і містять як max, так і min.

(Тобто, говорячи конкретно про задачу 1.7:E, для отримання останньої з її відповідей можна розв'язати задачу 1.7:C (враховуючи лише верхні цианові числа й ігноруючи нижні фіолетові) будь-яким зі способів (13), (14) чи (16). У попередньому абзаці я виділяв са́ме «перші альтернативні ро́зв'язки» тому, що намагався сформулювати правило в ширшому вигляді, не лише конкретно про задачу 1.7:E.)

Однак, усе сказане від початку цього розд. 1.7.3 досі — *лише перша* з двох основних причин, які роблять ігри з не сталою сумою складнішими для аналізу. Перейдемо до розгляду дру́гої.

Як у задачі 1.7:D, так і в її продовженні 1.7:E є обмеження «всі $2N$ чисел різні, всі є цілими степенями двійки». Виявляється, коли нема обмежень на значення на картках, то, навіть у разі чіткого дотримання припущення «обидва гравці дбають лише про максимізацію кожен свого виграшу, не цікавлячись ні збільшенням, ні зменшення виграшу суперника, і при цьому спроможні розрахувати наслідки своїх дій»), розбір задачі 1.7:D неповний, а в деякому смислі навіть неправильний!

Розглянемо карткѝ $\frac{5}{4}$ $\frac{12}{2}$ $\frac{5}{2}$ $\frac{0}{2}$ (4 штуки, на крайній лівій верхнє число 5, нижнє 4, …, …, …). Спробуємо застосувати до них переходи динпрога, описані в наївному розборі задачі 1.7:D… і просто *не зможемо(!)* цього зробити. Коли 1-й гравець вибирає, чи вигідніше почати з забирання лівої $\frac{5}{4}$, чи правої $\frac{0}{2}$, йому потрібно знати, який розв'язок оптимальний для 2-го гравця для підзадачі «3 картки "$\frac{12}{2}$ $\frac{5}{2}$ $\frac{0}{2}$"», а який для підзадачі «3 картки "$\frac{5}{4}$ $\frac{12}{2}$ $\frac{5}{2}$"». І якщо для підзадачі "$\frac{5}{4}$ $\frac{12}{2}$ $\frac{5}{2}$" досить ясно, що відповіддю є суми $\frac{12}{6}$, то для підзадачі "$\frac{12}{2}$ $\frac{5}{2}$ $\frac{0}{2}$" 2-му гравцю просто байдуже, які з карток бра́ти, бо для нього вони однакові (будучи вельми різними для 1-го).

Для уникання цієї проблеми здається доречним домовитися про якесь додаткове припущення. Однак тут є щонайменше дві проблеми:

1. Навіть просто припущення, що гравець дбає лише про свій виграш, вже не завжди виконується на практиці (див. також гру «Ультиматум» та міркування про неї); якщо «поверх нього» нагромадити ще додаткові припущення — буде дуже важко розібратися, коли всі ці припущення виконуються, а коли ні.
2. Деякі очевидні додаткові припущення призводять не до тих наслідків, яких від них інтуїтивно очікуєш.

Зокрема, інтуїція може запропонувати «в разі вибору між варіантами, які дають однакові свої оцінки, вибирати той, який дає більшу оцінку суперника» — наприклад, сподіваючись, що він робитиме так са́мо, внаслідок чого *ніби* повинно зменшитися явне марнування ресурсів (наприклад: «якщо 2-му гравцю байдуже, чи взяти $\frac{12}{2}$, чи $\frac{0}{2}$» — хай бере $\frac{0}{2}$, йому однаково, а 1-му вигідніше додати собі 12 з $\frac{12}{2}$, чим додати 0 з $\frac{0}{2}$»). Однак, з цим є ціла купа проблем, від побоювань «а що, як я врахую інтереси суперника, а він мої інтереси не врахує?» до контрінтуїтивних прикладів, коли обидва гравці са́ме так і роблять, а в результаті виходить *гірше(!)*, чим якби вони робили інші вибори. Так буває досить рідко[6]. Але все ж буває.

У кожному з наступних двох прикладах розглянуто, як для однакових вхідних даних один раз застосовується в точності описана ідея, інший раз — протилежна ідея «кожен суперник в разі однаковості своїх оцінок вибирає меншу (а не більшу) чужу». Обидва ці приклади оформлені приблизно аналогічно прикладам з наївного розбору задачі 1.7:D, але для кожної підзадачі, де там

[6] з'ясувати, «наскільки рідко», в рамках цього посібника нереально, бо навіть якщо говорити не про універсальний аналіз відразу всіх ігор, а лише конкретно про цю задачу, все'дно ймовірність сильно й складно залежить від кількості карток та значень на них

вказувалася одна пара чисел, тепер вказується дві: перед двокрапкою — що виходить, якщо при однаковості своїх оцінок кожен гравець завжди вибирає більшу чужу оцінку, після — якщо завжди меншу.

Приклад 1 (карткú $\frac{4}{30}$ $\frac{2}{1}$ $\frac{25}{30}$ $\frac{20}{1}$) з інтуїтивно очікуваним результатом «намагання взаємно частково врахувати інтереси суперника збільшило виграші».

Хід 1-го гравця	0	1	2	3
1	$\frac{4}{0}:\frac{4}{0}$	$\frac{2}{0}:\frac{2}{0}$	$\frac{25}{0}:\frac{25}{0}$	$\frac{20}{0}:\frac{20}{0}$
2	$\frac{4}{1}:\frac{4}{1}$	$\frac{25}{1}:\frac{25}{1}$	$\frac{25}{1}:\frac{25}{1}$	
3	$\frac{27}{30}:\frac{27}{30}$	$\frac{22}{30}:\frac{22}{30}$		
4	$\frac{45}{31}:\frac{29}{2}$!			

Хід 2-го гравця	0	1	2	3
1	$\frac{0}{30}:\frac{0}{30}$	$\frac{0}{1}:\frac{0}{1}$	$\frac{0}{30}:\frac{0}{30}$	$\frac{0}{1}:\frac{0}{1}$
2	$\frac{2}{30}:\frac{2}{30}$	$\frac{2}{30}:\frac{2}{30}$	$\frac{20}{30}:\frac{20}{30}$	
3	$\frac{25}{31}:\frac{4}{31}$!	$\frac{25}{2}:\frac{25}{2}$		
4	$\frac{22}{60}:\frac{22}{60}$			

Тут маємо «розумну змагальність з елементами співпраці», коли при виборі «маючи карткú $\frac{4}{30}$ $\frac{2}{1}$ $\frac{25}{30}$ і хід 2-го гравця — чи забрати $\frac{4}{30}$, чи $\frac{25}{30}$?» цей 2-й гравець вибирає $\frac{4}{30}$, бо йому однаково, а 1-му є різниця, й *завдяки* цьому процес оптимальної гри справді проходить через цю підзадачу (1-й забирає $\frac{20}{1}$, лишаючи $\frac{4}{30}$ $\frac{2}{1}$ $\frac{25}{30}$; 2-й забирає $\frac{4}{30}$, лишаючи $\frac{2}{1}$ $\frac{25}{30}$; 1-й забирає $\frac{25}{30}$, лишаючи $\frac{2}{1}$, яку 2-й забирає, вже не маючи вибору; 1-й набирає сумарно $20+25=45$, 2-й набирає сумарно $30+1=31$). Якщо ж обидва гравці намагаються в разі однаковості своїх виграшів вибирати гірше для суперника, тобто в тій самій ситуації «маючи карткú $\frac{4}{30}$ $\frac{2}{1}$ $\frac{25}{30}$ і хід 2-го гравця» 2-й вибирає $\frac{25}{30}$, то 1-й приймає рішення, що йому на першому ході вигідно забрати не $\frac{20}{1}$ (що дасть йому $20+4=24$ на картках $\frac{20}{1}$ і $\frac{4}{30}$), а $\frac{4}{30}$ (що дасть йому більшу суму $4+25=29$ на картках $\frac{4}{30}$ і $\frac{25}{30}$). Тобто, тут таке співробітництво вигідне: і 1-й отримає $45>29$, і 2-й отримає $31>2$.

Само собою, якщо 1-й, сподіваючись на це, забере $\frac{20}{1}$, а 2-й, всупереч цим сподіванням, забере $\frac{25}{30}$ — 1-й отримає виграш $20+4=24$, що менше не лише за 45, а й за 29. Про що раніше й говорилося, як про одну з проблем («а що, як я врахую інтереси суперника, а він мої інтереси не врахує?»). Але «найвеселіше» далі, у прикладі 2.

Приклад 2 (карткú $\frac{24}{1}$ $\frac{30}{15}$ $\frac{4}{1}$ $\frac{4}{30}$) з контрінтуїтивним результатом «намагання взаємно частково врахувати інтереси суперника **зменшило(!)** виграші».

Хід 1-го гравця	0	1	2	3
1	$\frac{24}{0}:\frac{24}{0}$	$\frac{30}{0}:\frac{30}{0}$	$\frac{4}{0}:\frac{4}{0}$	$\frac{4}{0}:\frac{4}{0}$
2	$\frac{30}{1}:\frac{30}{1}$	$\frac{30}{1}:\frac{30}{1}$	$\frac{4}{30}:\frac{4}{1}$!	
3	$\frac{28}{15}:\frac{28}{15}$	$\frac{34}{30}:\frac{34}{30}$		
4	$\frac{34}{2}:\frac{54}{31}$!			

Хід 2-го гравця	0	1	2	3
1	$\frac{0}{1}:\frac{0}{1}$	$\frac{0}{15}:\frac{0}{15}$	$\frac{0}{1}:\frac{0}{1}$	$\frac{0}{30}:\frac{0}{30}$
2	$\frac{24}{15}:\frac{24}{15}$	$\frac{4}{15}:\frac{4}{15}$	$\frac{4}{30}:\frac{4}{30}$	
3	$\frac{30}{2}:\frac{30}{2}$	$\frac{4}{45}:\frac{30}{31}$!		
4	$\frac{28}{45}:\frac{28}{45}$			

Абсолютно ті ж припущення, які у прикладі 1 дали можливість збільшити виграші обох гравців (хоч і з ризиком для гравця, який дотримається цих припущень, а його суперник ні), тепер призводять до цілком контрпродуктивного результату: *зменшують обидва* результати. Коли при виборі «маючи карткú $\frac{4}{1}$ $\frac{4}{30}$ і хід 1-го гравця — чи забрати $\frac{4}{1}$, чи $\frac{4}{30}$?» 1-й гравець вибирає $\frac{4}{1}$, бо «йому однаково, а 2-му є різниця», *са́ме це* призводить до того, що 2-му стає вигідним забирати $\frac{30}{15}$ у позиції «карткú $\frac{30}{15}$ $\frac{4}{1}$ $\frac{4}{30}$, хід 2-го гравця». Якби 2-й не мав причин сподіватися, що 1-й у позиції «карткú $\frac{4}{1}$ $\frac{4}{30}$, хід 1-го» забере $\frac{4}{1}$ й залишить йому $\frac{4}{30}$, то 2-й у позиції «$\frac{30}{15}$ $\frac{4}{1}$ $\frac{4}{30}$, хід 2-го» не забирав би $\frac{30}{15}$, бо вважав би це висикоризикованим, а не вигідним. Насамкінець, коли 1-й гравець вибирає свій найперший хід, то бачить, що в початковій позиції «$\frac{24}{1}$ $\frac{30}{15}$ $\frac{4}{1}$ $\frac{4}{30}$, хід 1-го» хід «забрати $\frac{24}{1}$» дасть йому лише $24+4=28$ (на картках $\frac{24}{1}$ і $\frac{4}{1}$), а хід «забрати $\frac{4}{30}$» дасть більшу суму $4+30=34$ (на картках $\frac{4}{30}$ і $\frac{30}{15}$). Тобто, жоден з намірів «частково враховувати інтереси іншого» чи «уникати явного марнування ресурсів» у цьому випадку не спрацював: картку $\frac{4}{30}$, вигідну 2-му, забрав 1-й, а $\frac{24}{1}$, вигідну 1-му, забрав 2-й; гравці отримали загальні результати $4+30=34$ (1-й, на картках $\frac{4}{30}$ і $\frac{30}{15}$) та $1+1=2$ (2-й, на картках $\frac{24}{1}$ і $\frac{4}{1}$) — і все це при тому, що вони *можуть(!)* походити «1-й забирає $\frac{24}{1}$, 2-й забирає $\frac{4}{30}$, 1-й забирає $\frac{30}{15}$, 2-й забирає $\frac{4}{1}$» і тим набрати $24+30=54$ (1-й) та $30+1=31$ (2-й). Причому, для реалізації цього (кращого для кожного з гравців!) результату нікому з них *не* треба довірятися супернику і йти на

якийсь великий ризик (нема навіть ризику, співмірного з описаним наприкінці прикладу 1). Ось просто нехай 2-й не нахабніє, забираючи $\frac{30}{15}$ у позиції «картки $\frac{30}{15}$ $\frac{4}{1}$ $\frac{4}{30}$, хід 2-го», і гравці отримають свої 54 (1-й) та 31 (2-й). Але ж питання в тому, як оце «просто нехай 2-й не нахабніє» описати математично…

А наступне питання — нехай навіть це буде якось описано; це буде якесь досить складне́ припущення; які гарантії (чи, хоча б, причини для сподівань), що гравці будуть його дотримуватися?

Ще раз: описана у прикладі 2 ситуація, коли дотримання правила «при виборі між варіантами, які дають однакові свої оцінки, вибирати той, який дає більшу оцінку суперника» призвело до поганих результатів — виключення, а не правило. Щоб знайти цей приклад, я написав програму, яка перебрала кілька мільйонів випадкових прикладів, кожен розв'язала двома вищезгаданими способами і вибрала «найяскравіший». З більшою початковою кількістю карток, та сама програма дала такі результати:

кількість карток	20	100	500
обидва гірше	34297	41881	234
один гірше, інший так само	10573	9599	51
один гірше, інший краще	330622	539711	24196
один так само, інший краще	180899	36615	1421
обидва так само	293768	1121	1
обидва краще	149841	371073	74097
всього	1 млн	1 млн	100 тис

(Тобто, чим більше карток, тим *частіше* підхід «при однаковості своїх оцінок, вибирати **кращу** чужу» виграє у підходу «при однаковості своїх оцінок, вибирати **гіршу** чужу». Але можливість, що обом стане гірше, все'дно лишається. Плюс, усе це — результати порівнянь при конкретному розподілі значень на картках (`rnd.next(1,6)*rnd.next(1,6)`), де `rnd.next(1,6)` рівноймовірно повертає ціле число від 1 до 6 (обидві межі включно); при зміні розподілу результати можуть помітно змінюватися, і, як вже сказано, заплутані спроби аналізу всього цього не варто робити складовою цього посібника.)

Плюс, не обов'язково вибирати лише з цих двох підходів (однак, неясно, *які* слід узяти інші…).

Плюс, нікуди не дівся аргумент «а що, як я врахую інтереси суперника, а він мої інтереси не врахує?».

1.7.4 А як щодо трьох і більше суперників?

З одного боку, додається чимало проблем: і загострюється питання «чи правильно ми розуміємо, чого хочуть гравці?», і з'являється принципово нове питання «чи є коаліції?», і посилюються деякі з проблем розд. 1.7.3.

З іншого — попри все це, хоча б деякі з таких ігор цілком можна аналізувати, причому аналогічно розглянутому досі (зокрема, «дру́гим альтернативним ро́зв'язкам …»).

Коаліція (в теорії ігор) — це група з кількох гравців, які узгоджують між собою свої дії. Різновидів коаліцій може бути дуже багато; зокрема, ось *не(!)* повний перелік питань, на які можна відповісти по-різному (й від цього залежить, яка це коаліція й як її аналізувати):

- чи додаткові угоди всере́дині коаліції обов'язкові до виконання, чи члени коаліції можуть їх порушувати, коли вони стають невигідними?
- чи коаліція є таємною групою, члени якої приховують від інших свої зв'язки́, чи персональні склади коаліцій загальновідомі?
- чи всере́дині коаліції якось додатково до основних правил гри перерозподіляють виграші між її гравцями, чи ні?
- якщо перерозподіли виграшів у коаліціях є, а персональні склади коаліцій загальновідомі, то чи знають інші всі деталі такого перерозподілу?

Очевидно, що розгляд цих різновидів зайняв би багато часу, тож і не намагатимусь розглянути все це в рамках нашої дисципліни (це нереально).

Отже, розглянемо в цьому розд. 1.7.4 лише частковий випадок, коли гравців троє й ніяких коаліцій нема (кожен гравець все ще зацікавлений лише у збільшенні свого виграшу), а ходять гравці строго в порядку «1-й, 2-й, 3-й, 1-й, 2-й, 3-й, …». Очевидно, що цей випадок охоплює далеко не всі ігри. Але якщо гра задовольняє обмеженням цього випадку, то її можна аналізувати (чи, хоча б, пробувати аналізувати) аналогічно вже розглянутим «дру́гим альтернативним ро́зв'язкам …».

Задача 1.7:F. «Гра на максимум суми (1/2/3) для трьох гравців»

N карток викладені в ряд зліва направо. На кожній картці написане ціле число. ***Три*** гравці по черзі (строго в порядку «1-й, 2-й, 3-й, 1-й, 2-й, 3-й, …») забирають карткú, причому забирати можна лише з правого боку, або 1 картку, або 2 картки, або 3 картки (але, звісно, не більше карток, чим їх є). Закінчується гра, коли забрано всі карткú (поки хоч одна картка є, гравець зобов'язаний робити один із можливих ходів). Мета гри — отримати якнайбільшу суму (чисел, записаних на забраних картках).

Яку суму набере кожен з гравців, якщо всі вони гратимуть правильно, намагаючись лише максимізувати кожен свою суму?

Вхідні дані. У першому рядку вказано кількість карток N ($1 \leqslant N \leqslant 1234567$). У другому рядку через пробіли задані N цілих чисел (що не перевищують за модулем 10^3; інакше кажучи, з проміжку $[-1000; +1000]$) — значення, записані на картках.

Результати. Виведіть в один рядок три цілі числá, розділені пропусками — суми, які наберуть за правильної гри 1-й, 2-й і 3-й гравці відповідно.

Приклади:

Вхідні дані	Результати
7 11 11 11 11 11 11 11	33 33 11
8 3 2 999 4 6 7 -5 1	4 8 1005
9 23 -17 2 -13 5 3 11 -7 19	12 9 5

Примітки. У першому тесті, числа додатні й однакові, тому вигідно забирати якнайбільше карток: 1-й забирає три штуки, потім 2-й забирає три штуки, потім 3-й забирає останню й цим завершує гру; суми становлять $11 + 11 + 11 = 33$ для 1-го, $11 + 11 + 11 = 33$ для 2-го та 11 для 3-го.

Наведу лише структуру відповіді дру́гого тесту (не пояснюючи, чому вона така). 1-й забирає 1; 2-й забирає –5, 7 та 6; 3-й забирає 4, 999 та 2; 1-й забирає 3 й цим закінчує гру.

Розбір задачі. Спочаку наведу розв'язок, більш-менш схожий на «дру́гий альтернативний розв'язок задачі 1.7:A», але потім поясню, чому він (як і перший розв'язок задачі 1.7:D) дещо наївний.

Нехай позиції будуть ті сáмі «лишилось i карток, де $0 \leqslant i \leqslant N$ (включно)», для кожної такої позиції знаходимо трійку (a, b, c), де $p(i).a$ виражає, яку суму набере при правильній грі всіх трьох гравців той, кому дісталася позиція «лишилось i карток»; $p(i).b$ — яку суму набере при цьому гравець, який ходить наступним, а $p(i).c$ — яку суму набере наступний після наступного (тобто: 3-й, якщо за́раз хід 1-го; 1-й, якщо хід 2-го; 2-й, якщо хід 3-го). Знову можна мати єдину тривіальну підзадачу $p(0)$ (звісно, тепер $p(0) = (0,\ 0,\ 0)$) і цілком аналогічні (17) зв'язкú між підзадачами, в яких змінено, головним чином, те, що замість «щокроку міняти місцями $p(\ldots).a$ з $p(\ldots).b$» тепер буде «значення щокроку "по колу" переходять між трьома полями $.a$, $.b$, $.c$». Пропоную читачам визначити подальші деталі цих переходів (і взагалі довести цю ідею до стану готової програми) самостійно, спираючись на щойно наведені міркування та на наведені далі приклади.

Приклад 1 (відповідає тесту 2 з умови).

$c(i-1)$	$i-1$	i	$p(i).a$	$p(i).b$	$p(i).c$	скільки карток вигідно брати
		0	0	0	0	–
3	0	1	3	0	0	одну
2	1	2	5	0	0	дві
999	2	3	1004	0	0	три
4	3	4	1005	3	0	три
6	4	5	1009	5	0	три
7	5	6	17	1004	0	три
–5	6	7	8	1005	3	три
1	7	8	4	8	1005	одну

Приклад 2 (відповідає тесту 3 з умови).

$c(i-1)$	$i-1$	i	$p(i).a$	$p(i).b$	$p(i).c$	скільки карток вигідно брати
		0	0	0	0	—
23	0	1	23	0	0	одну
–17	1	2	6	0	0	дві
2	2	3	8	0	0	три
–13	3	4	–11	6	0	дві
5	4	5	5	–11	6	одну
3	5	6	9	5	–11	одну
11	6	7	20	5	–11	дві
–7	7	8	13	5	–11	три
19	8	9	12	9	5	три

Причини, чому цей алгоритм наївний. (Отже, ставитися до нього слід обережно.)

Причина перша. Хоч це й уже згадано загальними фразами, підкреслю ще раз на прикладі: ці результати (й узагалі цей підхід) — для ситуації, коли кожен гравець зацікавлений *лише* у збільшенні свого виграшу; для інших ситуацій він непридатний.

Наприклад, у дру́гому тесті перший хід 1-го гравця «взяти одну картку 1» призводить до того, що найбільша картка 999 дістається 3-му гравцю. Але якби 1-й гравець був сильно зацікавлений у збільшенні виграшу 2-го гравця за рахунок 3-го, то 1-й гравець міг би вибрати перший хід «взяти три картки 1, –5 та 7», після чого 2-й міг би походити «взяти три картки 6, 4 та 999». Аналогічно, якби 2-й був сильно зацікавлений у збільшенні виграшу 1-го за рахунок 3-го, то (в разі того ж першого ходу 1-го гравця «взяти одну картку 1») забирав би не три картки, а одну картку –5; яким би після того не був хід 3-го, після нього 1-й матиме можливість забрати якісь три картки, завжди включно з карткою 999.

На це можна відповісти «так в усих цих прикладах гравець ходить всупереч власним інтересам (що протирічить аксіомам теорії ігор)»: наприклад, коли 1-й гравець своїм першим ходом забирає три картки (а не одну), то зменшує свій виграш з $(1)+(3)=4$ до $(1-5+7)=3$; для 2-го гравця теж невигідно забирати одну картку замість трьох і отримувати, в найгіршому випадку, лише (-5) замість $(-5)+7+6=8$. Що ж, са́ме завдяки цій невигідності відхилень від нього розглянутий алгоритм чогось вартий. Але й на це теж є що відповісти: (1) у випадку таємних коаліцій (можете називати їх змовами, якщо від того на душі́ стане легше…) може бути (а може й не бути) таємний перерозподіл вирашів, який може приносити гравцю більше, чим він втрачає, ходячи неоптимально з точки зору офіційних правил; (2) див. причину другу.

Причина дру́га. Тут *можливі(!)* проблеми, схожі на згадані в розд. 1.7.3, коли водночас: (1) неясно, що слід вибирати в разі однакових виграшів; (2) від цього може залежати результат (істотно, тобто може змінитися не лише, хто які картки візьме, а й хто яку суму набере). (Й це для гри зі сталою сумою! Але ж трьох, а не двох, гравців…)

Розгляньмо вхідні дані «5 карток, значення –2 1 –1 –1 3» та спробуймо побудувати для них таблицю, аналогічну попереднім.

$c(i-1)$	$i-1$	i	$p(i).a$	$p(i).b$	$p(i).c$	скільки карток вигідно брати
		0	0	0	0	—
–2	0	1	–2	0	0	одну
1	1	2	1	–2	0	одну
–1	2	3	0	–2	0	дві
			(див. примітку 1)			
–1	3	4	–1	$?\begin{bmatrix}0\\-2\end{bmatrix}$	$?\begin{bmatrix}-2\\0\end{bmatrix}$	$?\begin{bmatrix}\text{одну}\\\text{три}\end{bmatrix}$
			(див. примітку 2)			
3	4	5	Мало того, що неясно, чи виходить $(2,0,-2)$, чи $(3,-1,-2)$; так ще й вибір кращого ходу залежить *не(!)* від гравця, який ходить зáраз.			

Примітка 1. До цього місця все гаразд; але там однакову суму –1 можна отримати двома способами: взявши або одну картку –1, або три –1, -1 і 1. І не варто казати «це неважливо, сума ж однакова», бо виграші інших гравців (поля́ $.b$ і $.c$) різні.

Примітка 2. Також *неправильно* казати, ніби «істотна невизначеність є лише для 2-го і 3-го гравців, а для 1-го можуть відрізнятися хіба лиш набори карток, а сума однакова, бо невизначеність буває лише за однакових сум». Це так, коли невизначеність вперше з'являється, але ж далі відбуваються «переходи "по колу" між полями $.a$, $.b$, $.c$». Наприклад, при спробі порахувати $p(5)$ неясно, чи слід вважати, що $p(4)=(-1,0,-2)$, чи $p(4)=(-1,-2,0)$; при цьому:

- у випадку $p(4)=(-1,0,-2)$
  - взяття однієї картки призводить до «3 зáраз, –2 потім, разом 1», сумарно $(1,-1,0)$;
  - взяття двох — до «$(3-1)=2$ зáраз, ніякого "потім" не буде», сумарно $(2,0,-2)$;
  - а взяття трьох — до «$(3-1-1)=1$ зáраз, ніякого "потім" (теж) не буде», сумарно $(1,1,-2)$;

  тобто найвигідніше брáти дві картки;
- натомість, у випадку $p(4)=(-1,-2,0)$
  - взяття однієї картки призводить до «3 зáраз, ніякого "потім" не буде», сумарно $(3,-1,-2)$;
  - взяття двох та взяття трьох тотожні попередньому, $(2,0,-2)$ і $(1,1,-2)$ відповідно,

  тобто найвигідніше брáти одну картку.

Враховуючи все це, задача 1.7:F насправді не має цілком чесного й чіткого розв'язку; скільки-небудь адекватно здати її можна *лише* тому, що там спеціально підібрані такі тести, де не виникає проблема вибору між однаковими максимумами (чесність чого сумнівна, бо це можна розцінити як ігнорування проблеми замість її вирішення). Однак, біда в тому, що тут справді неясно, як вирішити цю проблему чесно й універсально.

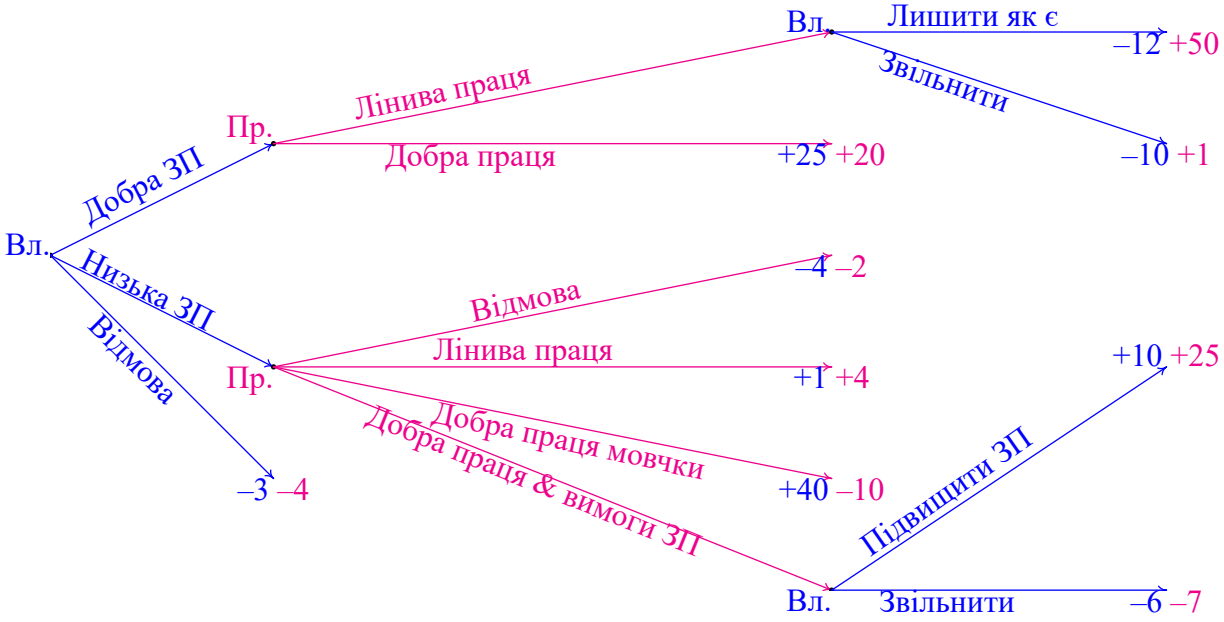
За бажання, можна пробувати врахувати невизначеність цієї задачі якимись із засобів розд. 1.9. Однак, безпосередньо в цьому посібнику варіанти й особливості таких поєднань не розглянуті.

1.8 Послідовні кількакрокові ігри на дереві рішень

Такі ігри часто розглядають зовсім без прив'язки до всього досі розглянутого в цьому посібнику. Але я не бачу нічого поганого в тому, щоб підкреслити зв'язок цього розд. 1.8 зі схожими ситуаціями (особливо, розд. 1.7.2–1.7.3), навіть якщо інші описи теорії ігор цього не роблять. Власне, матеріал цього розд. 1.8 має настільки багато спільного з розд. 1.7.2–1.7.3, що цілком можна було б зробити його не окремим підрозділом 1.8, а підпідрозділом 1.7.5 підрозділу 1.7.

1.8.1 Постановка задачі та приклад послідовної кількакрокової гри на дереві рішень

У цьому розд. 1.8 розглядаємо ігри, де є кілька (двоє чи більше) гравців, замість загального опису правил гри відразу задається сукупність унікальних ситуацій-позицій, де «гравець … приймає одне

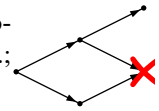


з кількох (від 2 і більше) рішень: або …, …, або …»; внаслідок кожного з цих рішень настає або одна з багатьох фінальних позицій, яка має числові значення виграшів усіх гравців, або інша ситуація-позиція, де якийсь гравець приймає одне з рішень.

Тобто, в більшості послідовних ігор граф гри потрібно ще будувати на основі правил гри, а тут цей граф гри відразу задається явно. Зазвичай вимагають, щоб цей граф гри був кореневим деревом у смислі [28, 29], звідки, зокрема, випливає його ациклічність у смислі [10] та розд. 1.1.5.

(Ациклічність необхідна для безпроблемної роботи розглянутих далі засобів чи то зворотної індукції, чи то динамічного програмування, з тих самих причин, що й у інших аналогічних ситуаціях: і це позначення позицій виграшними та програшними з розд. 1.2.2, і це знаходження числових виграшів з розд. 1.7, й інші схожі. Більш того, розглянутий далі алгоритм зворотної індукції на дереві рішень дещо схожий на, наприклад, «наївний» алгоритм розв'язування задачі 1.7:D «Гра на максимум суми (L/R, два числа на картці) — 1» (але, звісно, потреба застосовувати його до вузлів кореневого де́рева вносить свої зміни).

Кореневе дерево є особливим випадком ациклічного орграфа, коли, крім циклів, заборонені ще й ситуації, коли в одну вершину можна прийти різними шляхами (як на цьому рис.; ре́бра й вершини не підписані, бо це просто приклад орграфа, а не приклад графа гри).

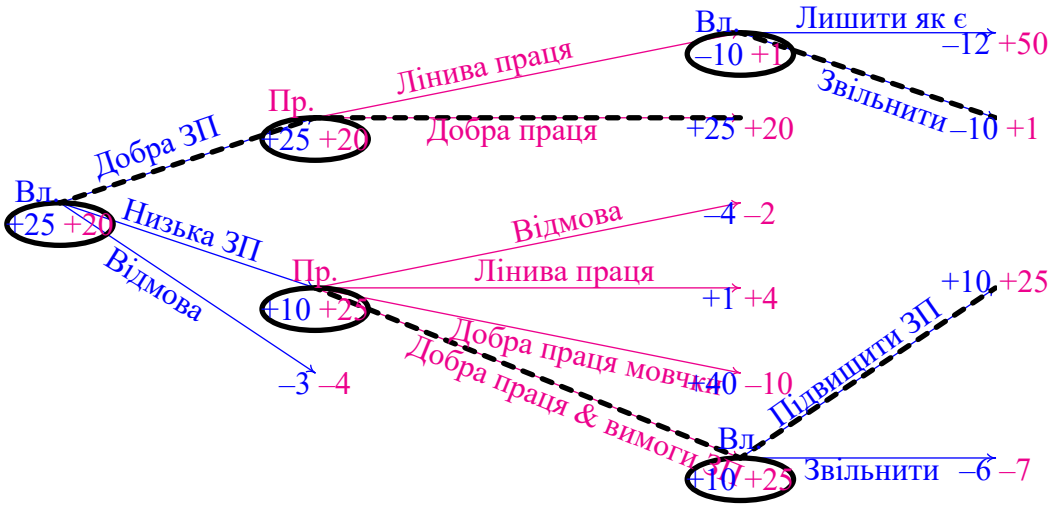


Наскільки добре й виправдано, що в цьому способі зазвичай вимагають розглядати са́ме дере́ва, а не довільні ациклічні орграфи — питання філософське; про це ще буде згадано тут.)

Такі ігри легко узагальнити на випадок довільної кількості гравців. Якщо гравців $k > 2$, *не* обов'язково, щоб вони ходили як 1-й, 2-й, …, k-й, знову 1-й, знову 2-й, …, знову k-й, …; можна робити ці послідовності виборів якимись іншими й навіть різними в різних гілках де́рева рішень; однак, якось дивно мати таке дерево рішень, де два чи більше разів підряд рішення приймає той са́мий гравець (чом би не об'єднати кілька підряд рішень одного гравця в одне рішення?), що й призводить до того, що коли гравців двоє, то вони ходять по черзі.

Тому, в найпростішому варіанті для двох гравців, кількакрокова послідовна гра на дереві рішень відбувається так: спочатку 1-й гравець приймає рішення, вибравши один з можливих варіантів свого ходу; деякі з цих варіантів ходу можуть відразу мати числові значення, скільки від того виграв 1-й гравець і скільки 2-й, решта передбачають, що тепер приймає рішення (вибирає варіант свого ходу) 2-й гравець, вже знаючи, яке рішення прийняв (який варіант свого ходу вибрав) 1-й гравець. Серед варіантів ходу 2-го гравця теж деякі можуть мати готові числові значення, скільки виграв 1-й гравець і скільки 2-й, а решта передбачати, що тепер приймає рішення (вибирає варіант свого ходу) знову 1-й гравець. І так далі.

Всі гілки такого де́рева мусять колись закінчитися «листками» з готовими числовими значеннями, скільки виграв 1-й гравець і скільки 2-й, але шляхи від кореня (початкового вибору 1-го гравця)



до різних «листків» є різними, й кількості проміжних вершин-рішень у цих різних шляхах можуть бути різними (однаковими теж можуть).

Наприклад, на рисунку вгорі стор. 95 зображено модель взаємодії між власником підприємства («Вл.») та охочим найнятися працівником («Пр.») після співбесіди між ними.

Звісно, це лише модель, яка враховує лише спрощену частину реальності. Чому розглянуті са́ме такі варіанти рішень і звідки взяті конкретні чи́сла у вузлах-«листках» — окреме складне́ питання, часто складіше за алгоритм розв'язання задачі, коли все це вже задано. З одного боку, «вони виражають індивідуальні особливості конкретного власника й конкретного працівника» та «слід просто подивитися на реальність і описати, хто що може зробити й кому що наскільки вигідно»; з іншого — це зовсім не «просто». Тим не менш, щоб застосовувати математичні та/або алгоритмічні засоби, модель потрібна, от і продовжмо її розглядати.

Власник може відразу відмовити у взятті на роботу, тоді сторони отримують помірно-від'ємні «виграші» (–3) та (–4) (час витратили, власник не отримав працівника, потенційний працівник не отримав робо́ти), на чому взаємодія між ними закінчується. Два альтернативні варіанти — власник може взяти на роботу, призначивши або низьку зарплату (ЗП), або добру ЗП. У кожному з цих випадків результат залежить від подальшого рішення працівника (який вже знає рівень призначеної ЗП).

Якщо ЗП добра і працівник працює добре — сторони отримують непогані виграші (+25 та +20). Якщо ж, при добрій ЗП, працівник працює ліниво — власнику слід вирішити, чи змиритися й лишити як є (це найкращий варіант для працівника (+50), але поганий для власника), чи звільнити працівника.

Якщо власник призначив низьку ЗП, працівник теж має варіанти «працювати добре» і «працювати ліниво» (але в них і числові вираження виграшів інші, і вибір працювати ліниво не потребує додаткового рішення власника, він з цим змирився ще призначаючи низьку ЗП), і ще варіанти «відмовитися від такої роботи» та «працювати добре, але вимагати підвищити ЗП» (після чого власник мусить або «підвищити ЗП», або «звільнити працівника»).

Такі задачі слід розв'язувати, починаючи від найдальших від кореня «листків».

(Що цілком відповідає хоч заповненню виграшності/програшності позицій «назустріч ходам» у розд. 1.2, хоч розв'язуванню підзадач динамічного програмування від менших (ближчих до кінця гри) до більших (ближчих до початкової позиції) у розд. 1.7.1–1.7.2. І в усьому цьому нема нічого дивного: майже все досі розглянуте (крім розд. 1.6) так чи інак експлуатує ту са́му ідею, чи ми називатимемо її зворотною індукцією (вона ж індукція назад), чи динамічним програмуванням. Са́ме у теорії ігор це частіше називають зворотною індукцією; але було б тупо вигадувати якісь неіснуючі відмінності між насправді схожими задачами, лише тому, що в одних із них більш прийнято казати «динпрог», а для інших «зворотна індукція».)

Наприклад, якщо власник уже призначив добру ЗП, а працівник уже працює ліниво, то власнику слід вибрати, змиритися чи звільнити працівника; відповідні вузли є «листками», і серед чисел (–12) та (–10) більшим є (–10), тому власник вибирає варіант «звільнити», отже — пара оцінок «(–10) власнику, (+1) працівникові» переноситься на проміжний вузол. Те, що у парах (–10, +1) та (–12, +50) дуже різний виграш працівника, ніяк не впливає, бо це хід власника, і він враховує лише свій виграш.

(Абсолютно аналогічно проголошеному в розд. 1.7.2).

Далі все відбувається аналогічно:

1. у ситуації «ЗП низька, працівник працює добре й вимагає підвищити ЗП» власник при виборі між (+10, +25) і (–6, –3) вибирає (+10, +25) (бо +10 = max(+10,–6));
2. у ситуації «ЗП добра», працівник при виборі між (–10, +1) і (+25, +20) вибирає (+25, +20) (бо +20 = = max(+1,+20));
3. у ситуації «ЗП низька», працівник при виборі між (–4, 0), (+1, +4), (+40, –10) і (+10, +25) вибирає (+10, +25) (бо +25 = max(0,+4,–10,+25));
4. у корені (початковій ситуації), власник при виборі між (+25, +20), (+10, +25) і (–3, –4) вибирає (+25, +20) (бо +25 = max(+25,+10,–3)).

Отже, відповідь всієї задачі: виграш власника 25, працівника 20. Вибори, наведені у списку вище, повторені на рисунку вгорі стор. 96.

Задача 1.8:A. «Кількакрокова послідовна гра на дереві рішень — 1»

Розв'яжіть базовий варіант кількакрокової послідовної гри на дереві рішень.

Вхідні дані. Єдиний рядок, який містить дерево, закодоване таким способом:

1. кожен окремий вузол-«листок» (фінальна позиція) подається у вигляді $(p_1\ \ p_2)$, тобто: відкривна кругла дужка "(", виграш 1-го гравця, пробіл, виграш 2-го гравця, закривна кругла дужка ")";
2. кожен вузол, що не є «листком» (отже, в ньому приймається рішення й різні варіанти ведуть до різних подальших вузлів) подається у вигляді: відкривна квадратна "[" (якщо хід 1-го гравця) чи кутова "<" (якщо 2-го) дужка, пробіл, код вузла, куди веде перший варіант рішення, пробіл, код вузла, куди веде дру́гий варіант рішення, пробіл, …, код вузла, куди веде останній варіант рішення, пробіл, відповідна закривна дужка ("]" для ходу 1-го, ">" для 2-го). При цьому вкладені коди вузлів можуть відповідати чи то попередньому пункту (якщо вони вже не мають розгалужень), чи то поточному (якщо мають).

(Наприклад, "`[ (2 3) (6 1) ]`" подає дерево, де початковий вузол має розгалуження на два варіанти, обидва є «листками», їхні виграші становлять 2 3 і 6 1. А на рисунку зі стор. 98 зображено, яким рядком кодується дерево, неоднократно зображене на стор. 95–96 (тепер без словесних позначок, зате частково вказана відповідність піддерев і підрядків).)

Гарантовано, що:

1. дерево містить щонайменше один вузол;
2. рядок, що кодує дерево, має не більше 10^5 символів (сумарно всіх, включно з цифрами, мінусами (якщо є), пробілами та всіма дужками);
3. для проміжних вузлів, кількість варіантів вибору перебуває в межах від 2 до 6;
4. всі виграші обох гравців (рахуючи по всім вузлам-«листкам») є різними числами, кожне з яких поміщається у 32-бітовий знаковий тип.

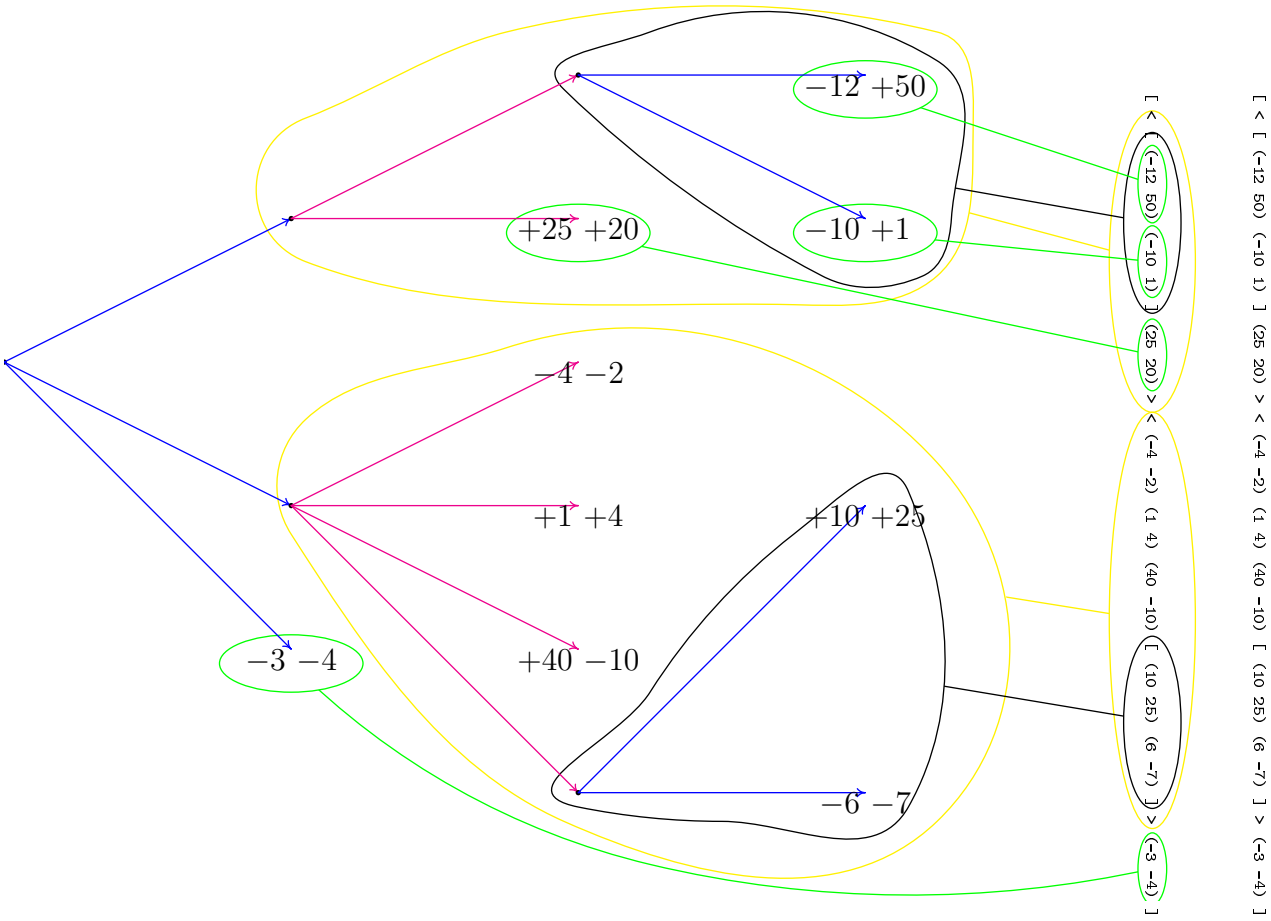
Результати. Програма виводить два числа в одному рядку, розділені пропуском: виграші, які мають отримати 1-й та 2-й гравці відповідно, якщо застосувати описаний алгоритм зворотної індукції.

Приклад:

Вхідні дані
`[ < [ (-12 50) (-10 1) ] (25 20) > < (-4 -2) (1 4) (40 -10) [ (10 25) (6 -7) ] > (-3 -4) ]`
Результати
`25 20`

Розбір задачі. На мою думку, тут головна складність — адекватно подати (кажуть також «представити», але мені більш подобається називати це «подати») дерево рішень у зручному для обробки вигляді, а помірна складність самóї зворотної індукції майже непомітна на цьому тлі.

Спосіб 1 — обробка з використанням стека. Можна переробити під цю задачу й такі вхідні дані відомий алгоритм обчислення виразу. Якось ≈так:



1. Завести клас/структуру/кортеж/…, полями як(ого/ої) будуть, наприклад, `List` пар виграшів двох гравців та делегат ме́тоду визначення максимуму в поточному вузлі (за 1-м числом чи за 2-м).
   (Опціонально, *якщо* хочемо контролювати правильність чи то нашого коду, чи то рядка з вхідними даними, можна додати ще поле «яким символом (“`]`” чи “`>`”) повинен закінчитися поточний фрагмент». Якщо виходити з припущення «вхідні дані коректні» й писати правильно, ця опція непотрібна.)
2. Завести стек (як тип даних) об’єктів пункту 1.
3. Поки вхідні дані не скінчилися, повторювати:

3.1. Якщо поточна лексема “`[`”, покласти у стек ще один об’єкт пункту 1, з порожнім `List`’ом, делегатом на метод «вибирати пару з більшим 1-м числом» і символом “`]`” (якщо зберігаємо).
3.2. Якщо поточна лексема “<”, покласти у стек ще один об’єкт пункту 1, з порожнім `List`’ом, делегатом на метод «вибирати пару з більшим 2-м числом» і символом “>” (якщо зберігаємо).
3.3. Якщо поточна лексема “`(`”, то прочитати число, ще число, дописати цю пару в `List` наразі верхнього елемента стека, прочитати “`)`”.
3.4. Якщо поточна лексема “`]`” або “>”, то застосувати до всіх пар з `List`’а наразі верхнього елементу стека той метод визначення кращої пари, делегат на який зберігається в цьому ж наразі верхньому елементі стека; результат цього метода поки що тримати в додатковій змінній, а верхній елемент стека прибрати; якщо якраз скінчилися і вхідні дані, й стек, то цей результат буде кінцевим результатом усієї задачі; якщо скінчилося тільки щось одне — щось не так, треба перевіряти правильність реалізації алгоритму та/або вхідних даних; якщо нічого не скінчилося — дописати цей результат у `List` того елемента, який щойно став верхнім, і продовжити.
   (Якщо зберігаємо й перевіряємо опціональне поле «яким символом повинен закінчитися поточний фрагмент», то у процесі також звірити, чи відповідають одне́ о́дному це поле й поточна лексема (якщо відповідають — зрадіти, що поки що все гаразд; якщо ні — заявити про проблему).)

Приклад виконання цього алгоритму:

Будемо щоразу вказувати щойно розглянуту частину виразу чорним кольором, розглянуту раніше — цияновим, ще не розглянуту — жовтим; при цьому будемо вказувати поточний стан стека.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

]	ε

Після “[”, згідно п. 3.1, у стеку з’явився один рівень, який слід буде закінчити символом “]”; значень цього рівню ще нема.

(Тут, умовно позначатимемо відсутність пар як ε; у програмі це `List`, який вже створений, але наразі не містить елементів.)

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

>	ε
]	ε

Після “<”, згідно п. 3.2, у стеку з’явився *ще* один рівень, який слід буде закінчити “>”; значень жодного з рівнів ще нема.

(Тут і далі зображаємо, що елементи стека додають і прибирають згори.)

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

]	ε
>	ε
]	ε

Знову п. 3.1, знову з’явився ще один рівень, значень досі нема.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

]	(-12 50)
>	ε
]	ε

Прочитавши “(”, слід виконати п. 3.3, тобто прочитати поточну пару й додати її у список поточного рівня.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

]	(-12 50) (-10 1)
>	ε
]	ε

Діючи аналогічно, додали у список поточного рівня наступну прочитану пару.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

>	(-10 1)
]	ε

Прочитавши (правильну) закривну дужку “]”, слід виконати п. 3.4, тобто застосувати до рівня, який перед цим був поточним, вибір кращої пари, дописати її у список попереднього рівня, а рівень, який щойно був поточним, знищити.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

>	(-10 1) (25 20)
]	ε

Прочитавши “(”, слід виконати п. 3.3, тобто прочитати поточну пару й додати її у список поточного рівня.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

]	(25 20)

Прочитавши (правильну) закривну дужку “>”, слід виконати п. 3.4; при цьому рівень, який щойно був поточним, знищується, а його краща пара дописується у список попереднього рівня.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

>	ε
]	(25 20)

До стека додався рівень з поки що порожнім списком, який слід буде закінчити символом “>”.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

>	(-4 -2)
]	(25 20)

Прочитавши “(”, слід виконати п. 3.3, тобто прочитати поточну пару й додати її у список поточного рівня.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

>	(-4 -2) (1 4)
]	(25 20)

Аналогічно попередньому.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

>	(-4 -2) (1 4) (40 -10)
]	(25 20)

Аналогічно попередньому.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

]	ε
>	(-4 -2) (1 4) (40 -10)
]	(25 20)

Після “[”, згідно п. 3.1, у стеку з’явився один рівень, який слід буде закінчити символом “]”; всі значення лишились, як були.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

]	(10 25)
>	(-4 -2) (1 4) (40 -10)
]	(25 20)

Прочитавши “(”, слід виконати п. 3.3, тобто прочитати поточну пару й додати її у список поточного рівня.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

]	(10 25) (6 -7)
>	(-4 -2) (1 4) (40 -10)
]	(25 20)

Прочитавши “(”, слід виконати п. 3.3, тобто прочитати поточну пару й додати її у список поточного рівня.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

>	(-4 -2) (1 4) (40 -10) (10 25)
]	(25 20)

Прочитавши (правильну) закривну дужку “]”, слід виконати п. 3.4; при цьому рівень, який щойно був поточним, знищується, а його краща пара дописується у список попереднього рівня.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

]	(25 20) (10 25)

Прочитавши (правильну) закривну дужку “>”, слід виконати п. 3.4; при цьому рівень, який щойно був поточним, знищується, а його краща пара дописується у список попереднього рівня.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

]	(25 20) (10 25) (-3 4)

Прочитавши “(”, слід виконати п. 3.3, тобто прочитати поточну пару й додати її у список поточного рівня.

[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]

стек порожній

Прочитавши (правильну) закривну дужку “]”, слід виконати п. 3.4; при цьому рівень, який щойно був поточним, знищується, а його краща пара, оскільки одночасно скінчилися і вхідні дані, і стек, є остаточним результатом усього алгоритму.

Спосіб 2 — рекурсивна обробка. Аналогічні дії можна робити не зі стеком як типом даних, а з рекурсією (отже, з програмним стеком). Доречними можуть бути такі, наприклад, методи:

- `ReadLeaf` — читає виграші фінальної («листкової») вершини і зберігає як пару; цей метод *не* є рекурсивним.
- `ProcessChoiceOne` та `ProcessChoiceTwo` — пара тісно пов’язаних, але різних методів, кожен з яких читає, обробляє і знаходить результат для піддéрева (`ProcessChoiceOne` — 1-го гравця, `ProcessChoiceTwo` — 2-го). Для обробки скільки-небудь нетривіальних вхідних даних вони мусять взаємно викликати один óдного, утворюючи тим непряму (indirect) рекурсію.
  (*Можна* зробити не два «тісно пов’язані, але різні» методи `ProcessChoiceOne` та `ProcessChoiceTwo`, а один, якому передавати більше аргументів, зокрема й делегат на метод вибору кращої пари; це той випадок, коли важко визначитися, чи краще один складніший метод, чи два значно простіші, але зі значним дублюванням коду.)

У формулі (18) наведено приклад, які частини виразу який метод обробляє.

Спосіб 3 — менш ефективний, але дещо простіший. (Хоча, це лише точка зору, що простіший. Я так думаю, що простіший — особливо, якщо вміти користуватися бібліотекою `Regex`, хоча

$$\underbrace{\underbrace{[\ <\ [\ \underbrace{\underbrace{(-12\ 50)}_{\texttt{ReadLeaf}}\ \underbrace{(-10\ 1)}_{\texttt{ReadLeaf}}}_{\texttt{ProcessChoiceOne}\text{ вертає }(-10\ 1)}\ ]\ \underbrace{(25\ 20)}_{\texttt{ReadLeaf}}}_{\texttt{ProcessChoiceTwo}\text{ вертає }(25\ 20)}\ >\ <\ \underbrace{\underbrace{(-4\ -2)}_{\texttt{ReadLeaf}}\ \underbrace{(1\ 4)}_{\texttt{ReadLeaf}}\ \underbrace{(40\ -10)}_{\texttt{ReadLeaf}}\ [\ \underbrace{\underbrace{(10\ 25)}_{\texttt{ReadLeaf}}\ \underbrace{(6\ -7)}_{\texttt{ReadLeaf}}}_{\texttt{ProcessChoiceOne}\text{ вертає }(10\ 25)}\ ]\ >\ \underbrace{(-3\ -4)}_{\texttt{ReadLeaf}}\ ]}_{\texttt{ProcessChoiceTwo}\text{ вертає }(10\ 25)}}_{\texttt{ProcessChoiceOne}\text{ вертає }(25\ 20)} \tag{18}$$

накрайняк можна й без неї.) Можна, взагалі не переймаючись ні стеком, ні деревовидною структурою гри, багатократно використати перетворення «знайшовши підрядок “[кілька штук ( )]” або підрядок “< кілька штук ( ) >”, замінити цей підрядок на його значення».

(Наприклад: почавши з усе того ж “[< [(-12 50) (-10 1)] (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]”, замінити “[(-12 50) (-10 1)]” на його значення “(-10 1)”, вийде “[< (-10 1) (25 20) > < (-4 -2) (1 4) (40 -10) [(10 25) (6 -7)] > (-3 -4)]”; потім замінити “[(10 25) (6 -7)]” на його значення “(10 25)”, вийде “[< (-10 1) (25 20) > < (-4 -2) (1 4) (40 -10) (10 25) > (-3 -4)]”; потім замінити “< (-10 1) (25 20) >” на його значення “(10 25)”, вийде “[(25 20) < (-4 -2) (1 4) (40 -10) (10 25) > (-3 -4)]”; потім замінити “< (-4 -2) (1 4) (40 -10) (10 25) >” на його значення “(10 25)”, вийде “[(25 20) (10 25) (-3 -4)]”; останнє перетворення створить вже остато́чне значення “(25 20)”).

Ефективність цього способу помітно менша, чим у раніше розглянутих, бо є чимало непродуктивних витрат на копіювання підрядків.

1.8.2 Ще деякі міркування про дерева рішень і зворотну індукцію на них.

Практичну цінність&застосовність цього алгоритму сильно знижує поєднання таких факторів.

Фактор 1. Для застосування зворотної індукції на дереві рішень кожен гравець повинен знати все дерево, включно з величинами виграшів у всіх «листках», причому як своїх, так і суперник(а/ів). Наприклад, коли працівник робить вибір у ситуації «власник призначив низьку ЗП», доцільність варіанту «працювати добре, вимагаючи підвищити ЗП» обґрунтовується тим, що на наступному ході власнику буде вигідніше все-таки підвищити ЗП, чим звільнити працівника, який почав працювати добре. Але якщо працівник думає, що це так, а насправді власник настільки не любить підвищувати ЗП, що справжні величи́ни виграшу в цьому «листку» не (+10, +25), а (–10, +25), тобто власнику краще звільнити такого «бунтівливого» працівника — той са́мий вибір працівника призвів би вже до його звільнення власником, що працівнику менш вигідно, чим варіант «лінива праця». Навіть якщо працівник знає сукупність варіантів де́рева ходів і правильно розраховує величи́ни власних виграшів у кожному вузлі — все це не рятує від невдалого рішення, спричиненого неправильною інформацією про чужий виграш. Ще одна схожа ситуація в цьому ж дереві: якщо виграш працівника у ситуації «ЗП добра, працівник працює добре» становить не (+20), а 0, то власнику насправді вигідно призначати низьку ЗП, а не високу (при високій ЗП такий працівник все’дно вибере «працювати ліниво», а неправильні (ті, що на рисунках) уявлення власника щодо виграшів працівника призводять до невдалого рішення власника).

(Ми не згадали са́ме цю проблему в розд. 1.7.3, бо там картки були об’єктивно видимими гравцям, тож ідеальний гравець мав усю цю інформацію (правильну). А в теперішній ситуації нема об’єктивних проміжних даних, на які можна покластися, є *лише* оцінки кінцевих («листкових») позицій, які треба «просто (насправді, ні) подивитися у реальному світі».)

Фактор 2. Застосування зворотної індукції на дереві рішень дає однозначний результат *лише* якщо варіанти, з яких треба обирати, завжди мають різні (неоднакові) оцінки. Нехай, наприклад, варіанти відповіді власника на ситуацію, коли він призначив низьку ЗП, а працівник почав працювати добре, але вимагати підвищення ЗП, мають виграші (–6, –7) («звільнити») та (–6, +25) («підвищити ЗП»). Тоді працівник взагалі ніяк не може оцінити, чи його вибір працювати добре, але вимагати підвищення ЗП призведе до підвищення ЗП (отже, виграшу (+25)), чи до звільнення (отже, виграшу (–7)). Через це, працівник ніяк не може визначити, чи вибір «працювати добре, але вимагати підвищення ЗП» для нього кращий за вибір «працювати ліниво», чи гірший. Що у свою чергу призводить до неможливості визначити оцінку для вузла «власник взяв на роботу, призначивши низьку ЗП, тепер хід працівника».

(В таких випадках іноді пропонують зберігати чи то мінімальне можливе й максимальне можливе значення, чи то всю множину можливих значень; *буває*, що це все-таки допомагає вибрати краще рішення. Наприклад, так *було б, якби* виграші були, наприклад, (–6, +4) («звільнити») та (–6, +25) («підвищити ЗП»); тоді можна *було б* міркувати так, що хоч і невідомо, який точно результат дасть вибір «працювати добре, але вимагати підвищення ЗП», але він точно не гірший за будь-який інший вибір ($\min(+25,+4) = 4 \geqslant 4 = \max(-2,+4,-10)$), і має шанси стати кращим за інші, тому й слід вибирати його. Однак, біда в тім, що ускладнення (збільшення громіздкості) є, а толк сумнівний: це при (–6, +4) та (–6, +25) вдалось *би* вибрати краще, а при раніше згаданих (–6, –7) та (–6, +25) не вдається.)

Тобто, маємо *ту ж проблему, що в розд. 1.7.3*, і, відповідно, ті ж контрінтуїтивні властивості спроб враховувати, крім свого виграшу, ще й чужий. Щоправда, тут оцінки проміжних позицій не додаються потихеньку картка за карткою, а переносяться з фінальних («листкових») вершин, що дає прихильникам дерев рішень можливість казати «Що значить “різні фінальні вершини мають однакові оцінки”? Так не буває! Щось мусить бути краще, а щось гірше; визначтеся, змініть оцінки так, щоб стали різними, і проблема невизначеності вибору при однакових оцінках зникне!». Що ж, якась частина правди в цьому справді є. Але проти цього все ж є пару заперечень. (1) А якщо змінювати оцінки все-таки не можна, от вони дані, й сказано їх не змінювати? (2) Не факт, чи зникне, чи «сховається на іншому рівні»; образно кажучи, це трохи схоже на «замітання сміття під килим»): проблеми ніби й не видно, але за деяких обставин вона може проявитися. Зокрема, коли доводиться штучно визначати, що краще і що гірше там, де в цьому нема впевненості, є великий ризик помилково визначити це неправильно, що загострює проблеми, описані вище як «фактор 1».

Фактор 3. Чим ближче до практичних застосувань теорії ігор, тим гострішим є питання «а чи правильними є прийняті в теорії ігор аксіоми та гіпотези?» та/або «а чи справді реальні люди діють так, як каже теорія ігор?». І тут стає вельми цікавим дослідження людської поведінки, описане далі, в аналізі гри «Ультиматум».

Ще, са́ме цьому варіанту зворотної індукції при виборі максимуму з можливих варіантів по суті байдуже, чи максимальний «трохи більше» за інші, чи «набагато більше». Тому, іноді вимагають, щоб для де́рева з n «листками» значення виграшу в листках були числами від 1 до n.

(Наприклад, якщо відсортовані значення виграшів працівника –10, –3, –2, 0, 1, 4, 20, 25, 50, то, на думку прихильників цієї вимоги, слід замінити –10 на 1, –3 на 2, –2 на 3, …, 50 на 9.)

Але це так лише конкретно для цієї версії алгоритма. Якщо ж, наприклад, намагаються працювати з імовірностями та математичними сподіваннями в іграх з неповною інформацією, щойно згадана ідея стає категорично неправильною, знову потрібні якнайточніші числові значення.

Величини виграшів, знайдені зворотною індукцією на дереві рішень, *не* є величинами, які кожен окремо гравець може забезпечити собі, хоч би як не грав інший. (Це цілком перегукується з розд. 1.7.3 і тому для уважних читачів повинно бути очевидним.) Зворотна індукція на дереві рішень розраховує, що суперник діятиме, намагаючись збільшити свій виграш, а не «як завгодно» і не «аби створити проблеми своєму супернику (нам)». З одного боку, це нормально, бо це одна з аксіом теорії ігор «Кожен гравець бажає виграти і прикладає (в межах, дозволених правилами гри) всіх зусиль для збільшення свого виграшу». Зі ще одного боку, це чудово, бо дає шанс вибрати *взаємо*вигідний варіант (якщо такий є).

Але з третього боку це означає, що навіть коли нам відоме все дерево рішень, і значення всіх оцінок (включаючи чужі), і ми вміємо застосувати зворотну індукцію — все’дно знайдена оцінка для нашого гравця не гарантована: якщо суперник помилиться чи зіграє безграмотно, він може погіршити не лише свій виграш, а й наш.

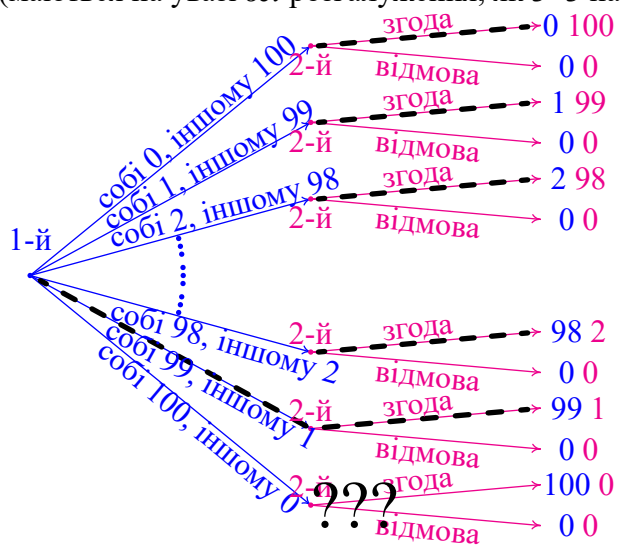
(Очевидно також, що, як і в задачі 1.7:E, можна розглянути окремо питання «якими будуть виграші при правильній грі обох гравців?» і окремо питання «який виграш може забезпечити собі такий-то гравець, хоч би як не грав суперник?». А от що робити з цими окремими ро́зв’язками далі — питання вже не математичної теорії ігор, а інших рівнів стосунків.)

1.8.3 Гра «Ультиматум»; чи описує дійсність зворотна індукція на дереві рішень?

Гра «Ультиматум» (див. [30, 31, 32]) має приблизно такий сюжет: «Експериментатор (який не є гравцем, бо не вибирає, як ходити) у присутності двох гравців кладе перед 1-м гравцем деякі матеріальні цінності (наприклад, 100 штук 10-гривневих монет), і повідомляє такі правила. Гравцям дістануться або всі ці монети, або нічого. 1-й гравець на свій розсуд вирішує, як розділити їх між собою і 2-м гравцем, кажучи “лишаю собі i монет” (де $0 \leqslant i \leqslant 100$, обидві межі́ включно), що за

правилами гри означає також “віддаю 2-му гравцю решту $100 - i$ монет”. 2-й гравець, вже після ходу 1-го, або погоджується з таким розділенням, або відмовляється. Якщо погоджується, то монети роздаються так, як сказав 1-й, а якщо відмовляється, то всі монети забирає назад експериментатор, і гравці нічого не отримують».

Якщо намалювати це у форматі, аналогічному попередньому, маємо приблизно таке дерево гри (маються на увазі *всі* розгалуження, як 3+3 намальовані, так і 95 пропущених).



Міркуючи за алгоритмом зворотної індукції на дереві рішень, маємо невизначеність, що має вибрати 2-й у позиції «1-й вибрав варіант “собі 100, іншому 0”»… Але то дрібниця. Справжня біда в іншому. В ось цьому протиріччі:

- цей алгоритм каже, що 2-й має погоджуватися з *будь-якою* пропозицією, де 1-й хоча б щось приділив 2-му, бо в разі згоди отримає деякий додатний виграш, тобто більший, чим 0 у разі відмови; спираючись на це, варіант «собі 99, іншому 1» повинен бути гарантовано вигідніший 1-му, чим будь-який з варіантів, де він пропонує 2-му будь-яку суму від 2 до 100;
- коли реально проводили психологічний експеримент, де випадковим людям пропонували зіграти рівно в цю гру (звісно, з дещо іншими матеріальними цінностями, бо проводили не в Україні), *не* розказуючи ні теорію ігор взагалі, ні про цей алгоритм зокрема, було *зовсім не так!*

Майже завжди 1-й пропонував від 30% до 50%, а в тих рідкісних випадках, коли 1-й усе-таки пропонував менше — 2-й, переважно, відмовлявся.

(Звісно, все це мало значну дисперсію (розкид) індивідуальних особливостей; також було з’ясовано статистично значимі кореляції з рівнем доходів піддослідної людини, фактом (не)знайомства гравців між собою, тощо; крім того, ці залежності сильно відрізнялися для різних культур (причому, як за протиставленням вестерни/Схід/первісне плем’я, так і за протиставленням мегаполіс/мале́ місто/село/рибальське селище/…).)

Усе це досить неприємно, й дехто робить з цього категоричні висновки, включно аж із «уся ваша теорія ігор ніц не варта, раз у ній таке твориться». Що ж, у цьому навіть є якась частина правди: математична теорія ігор, якою вона є наразі, справді не може відповісти на всі питання взаємодії сторін. Справді буває, що математична теорія ігор пропонує деякі засоби, які виявляються не дуже доречними.

Однак, попри цю обмеженість теорії ігор, її все ж можна акуратно й обмежено використовувати.

А ще, вертаючись до само́ї гри «Ультиматум», є деякі причини підозрювати, що са́ме в цьому випадку «чисті лабораторні умови експерименту» *не* сприяють природньому волевиявленню: аж занадто ясно, що це не є чесним заробітком (а коли є хоч якісь підстави так вважати, багатьом людям притаманно перебільшувати роль своєї праці, і, відповідно, щиро наполягати на збільшенні оплати). Відповідно, під час такого експерименту багато хто може сильніше, чим в реальних умовах, відчувати потребу ділити справедливо, що не має чіткого однозначного означення, але всі інтуїтивно згодні, що «собі майже все, іншому майже нічого» несправедливо; отже, такого і не варто пропонувати, і, з точки зору 2-го гравця, таке варто карати відмовою.

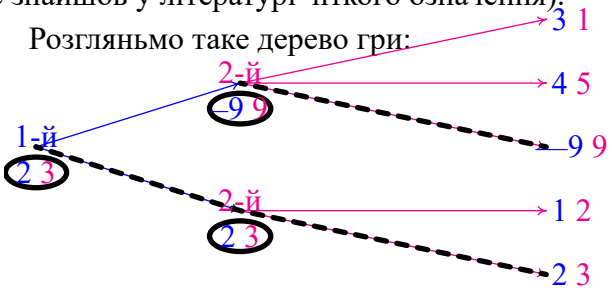
Ну, і відчуття «ото воно мені треба — заради такої дрібниці ризикувати тим, що люди про мене погане казатимуть» теж впливає, іноді не зовсім свідомо. Кінець кінцем, хоч на цей експеримент і витрачали помітні кошти, пропонована сума рідко коли виявлялася аж настільки значимою для гравця, щоб він погоджувався заради неї псувати свою репутацію. Такі (справді виявлені цими експериментами) кореляції, як «коли гравець відносно бідний, а сума відносно велика — відмови від запропонованого малого відсотку рідшають» та «коли гравці знайомі між собою або отримали виховання в малому суспільстві, де всі всіх знають — середній відсоток пропозиції більший, чим коли грають незнайомці, що виросли в мегаполіс(і/ах)» теж підтверджують цю думку.

Також, слід розуміти, що схожі проблеми мають і інші науки — наприклад, фізика на шкільному рівні може казати «опором повітря знехтуємо», й у результаті видавати результати, зовсім не відповідні спостереженням. Але лише завдяки тому, що фізика розвинулася, не зважаючи на цю критику не відповідних результатів, стало можливим розвинути вужчі технічні галузі, де той самий опір повітря враховують (причому, в різних галузях по-різному). От і коли теорія ігор дає не завжди добрі результати — слід контролювати, чи ідеалізований підхід більш-менш описує дійсність, чи ні; де не описує — шукати ме́жі застосовності занадто ідеалізованих засобів, альтернативні (покращені) засоби, тощо.

1.8.4 Комітменти: як заборона варіанта може покращити результат

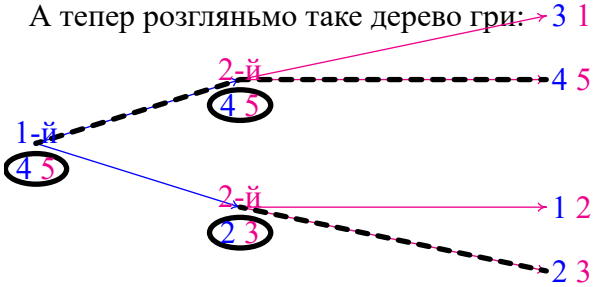
Слово «комітмент» у різних галузях людської діяльності означає досить різні речі, від майже повного синоніма сло́ва «зобов'язання» до цілком конкретних протоколів технічної фіксації деяких зобов'язань. У застосуванні са́ме до дерев рішень кількакрокових послідовних ігор воно має смисл, пояснений у найближчих обзацах на прикладі (без чіткого означення, бо так уже склалося, що я не знайшов у літературі чіткого означення).

Розгляньмо таке дерево гри:



Алгоритм зворотної індукції все'дно не використовує позначки на стрілках, тому їх пропустили. Зате зобразили дерево відразу з відповідями (оцінками проміжних та початкової позицій).

А тепер розгляньмо таке дерево гри:



Очевидно, що головна його відмінність від попереднього — відсутність (заборона) варіанту з виграшами «–9 9», решта відмінностей є наслідками.

Якщо обидва гравці знають і погоджуються, що той варіант щез/заборонений, то пораховані за алгоритмом зворотної індукції виграші *обох(!)* гравців покращилися. Тому, 2-му гравцю не слід сумувати за колишнім варіантом, який гіпотетично міг принести йому виграш «аж 9», а тепер його не стало. Він-то теоретично міг, але ніколи не наставав, бо дуже вже невигідний 1-му гравцю; це через нього 1-й гравець не вибирав (на першому рисунку) верхній варіант свого ходу. А коли цей варіант щез (на другому рисунку), 1-й гравець почав відчувати вигідним вибір верхнього варіанту свого ходу, що й покращило виграші обох гравців.

Ось такі ситуації (коли заборона як(ої/ихо)сь із гілок де́рева рішень сприяє підвищенню виграшів) і називають *комітментами* са́ме для дерев рішень.

Звісно, це лише приклад, він не є означенням. А вигідність проявилася, враховуючи відразу багато обставин: і категоричну невигідність 1-му гравцю завершення гри з виграшами «–9 9», і наявність іншого, досить взаємовигідного, ходу 2-го після того ж ходу 1-го. Тобто, тут нема серйозної теорії, це радше просто окреме дрібне спростереження.

Зате це спостереження (на відміну від результатів «Ультиматума») інтуїтивно здається цілком доречним та відповідним практиці.

Також це є ще й прикладом, коли, на відміну від цього «прикладу 2», зуміли якось виразити «Ось просто нехай 2-й не нахабніє…». Втім, від можливості якось це виразити до можливості знаходити такі ситуації (коли вони спочатку не задані) вельми довгий шлях, і ми не будемо розглядати, що на такому шляху теорія ігор вже вміє робити, а що ні.

Покращення завдяки комітментам працюють, якщо колишній хід до позиції, дуже невигідної (одному/деяким) з гравців і при цьому дуже вигідної інш(ому/им), або зовсім зникає, або щодо нього реально діє угода, за якою такі ходи жорстко й невідворотно караються. Але комітменти не працюють, якщо гравець, якому заборонений хід був вигідним, лише обіцяє не ходити туди, а технічна можливість туди піти лишається, й нема покарань за порушення таких обіцянок.

1.9 Ігри з неповною інформацією та недетерміновані (огляд)

Розгляньмо (дуже поверхово) ситуації, коли інформація *не* є повною та/або гра не є детермінованою.

(Деякий огляд того, які бувають невизначеності, вже був даний у розд. 1.1.4. Ще один (істотно інший) такий огляд можна бачити у [33].)

Якщо нема взагалі ніякої інформації, детальнішої за «ну, не знаю…», нічогісінько зробити неможливо, й результат буде таким самим «ну, не знаю…». Щоб отримати хоч скільки-небудь детальніший результат, абсолютно необхідна детальніший вигляд неповної інформації. Хоча б перелік можливих варіантів (наприклад: «ми недобачаємо, чи на картці написано 105, чи 106, чи 705, чи 706, але точно щось одне з цих чотирьох»). А якщо відомі ще й імовірності кожного з варіантів — можуть відкритися ширші можливості (а можуть і не відкритися).

(Ми не будемо розглядати ні сам частковий випадок неповної інформації «врахування в картярських іграх вже зроблених ходів», ні якісь його аналоги. Там, з одного боку, є з чим працювати (зокрема, але не тільки, всякі властивості у стилі «у цього гравця точно нема тузів, бо два уже у відбої, один у мене, один точно в такого-то іншого гравця»), й це може мати певне практичне застосування, але з іншого боку все це про особливості чи то конкретної гри, чи то картярської колоди, але не теорії ігор у цілому.)

Адекватних способів поєднувати неповну інформацію та засоби розділів 1.2–1.6 нема, тому все, що є і вже можна згадати тут, прив'язується до засобів розд. 1.7–1.8, тобто числових виграшів.

(Водночас, у розд. 1.1.6 вказано, чому ігри без числового виграшу можна вважати випадком ігор з числовим виграшем; у цих рамках, згадані тут засоби можуть бути в деяких випадках корисні для ігор на виграш/програш.)

1.9.1 Мінімум, матсподівання чи …?

Якщо виконуються обидві умови

1. відомі можливі варіанти («точно щось із …, або …, …, або …»), але ймовірностей цих варіантів не знаємо;
2. аналізуємо гру з точки зору конкретного гравця й тому можемо казати «наш хід/виграш/…»

— зазвичай варто вважати, що і суперник(и), і наше незнання «грають проти нас», тобто все станеться якнайгіршим для нас чином, і «всі неприємності, які можуть статися, справді стануться». Цим вдасться знайти *гарантовану* суму виграшу (щоб реальна виявилася точно не меншою, чим знайдена гарантована). Як цю ідею реалізовувіати — доброї й універсальної поради нема, багато залежить від особливостей конкретної гри. В деяких випадках слід просто замінити набір всіх можливих варіантів числа́ на мінімум з цих варіантів; у деяких інших випадках слід змінити свої уявлення про поведінку

іншого гравця (почати вважати, що він має мету зниження нашого виграшу); у деяких ще інших випадках варто припустити, ніби з'являється ще один гравець, який має мету знизити наш виграш.

Якщо можливі варіанти відомі не просто як перелік, а з імовірностями — як правило, варто враховувати ці ймовірності. Наприклад, якщо дуже вже багато уваги приділяти тому, що *в принципі ж можливі* «приморозки під кінець травня» чи «тривале повне затоплення поля у липні», і поєднати це з принципом «усі неприємності, що можуть статися, стануться» — можна дійти вельми сумнівного «висновку», ніби «взагалі не варто займатися ніяким сільським господарством на відкритому ґрунті». Якщо ж урахувати статистику, як-то «ризик приморозків під кінець травня менший 0,1%» та «ризик тривалих липневих затоплень ніде не є високим, але у різних місцях різний, і для такого-то конкретного по́ля становить 10%, для такого-то іншого 1%, а для такого-то ще іншого менше 0,1%» — результат істотно змінюється.

Коли ймовірності відомі й ураховуються при аналізі гри — найчастіше вважають, що кожен гравець прикладає зусиль до максимізації *матсподівання* (див., наприклад, [34]) свого виграшу.

(Нагадаю коротко: матсподівання (у дискретному варіанті, який тут і потрібен), виражає середнє з урахуванням імовірностей; у застосуванні до виграшів гри являє собою суму добутків

$$\sum_{i=1}^{k} p_i \cdot w_i,$$

де k — загальна кількість варіантів, з яких випадково береться той один, який фактично настане; p_i — ймовірність i-го варіанту; w_i — виграш цього i-го варіанту.

Наприклад, якщо відома така сукупність виграшів та їхніх імовірностей, як «є 1% імовірності виграти 1000, 9% виграти 5 і 90% виграти (–10) (програти 10)», то матсподівання слід порахувати як $0{,}01 \cdot 1000 + 0{,}09 \cdot 5 + 0{,}9 \cdot (-10) = 10 + 0{,}45 - 9 = 1{,}45$.

Якщо раптом хтось із читачів настільки погано пам'ятає властивості матсподівання, що це коротке нагадування не прояснило картину — для розуміння як решти поточного розд. 1.9 корисно вивчити/повторити відповідну тему теорії ймовірності за іншими джерелами.)

Однак, слід розуміти, що максимізація са́ме матсподівання — хоча й поширений спосіб аналізу ситуацій, коли є невизначеність і відомі ймовірності варіантів, але ні в якому разі не «єдино правильний». Наприклад, якщо порівняти щойно згадану сукупність результатів «є 1% імовірності виграти 1000, 9% виграти 5 і 90% виграти (–10) — матсподівання виграшу 1,45» з іншою сукупністю результатів «є 60% імовірності виграти 1 і 40% виграти 2 — матсподівання виграшу 1,4» (бо $0{,}6 \cdot 1 + 0{,}4 \cdot 2 = 0{,}6 + 0{,}8 = 1{,}4$), і припустити, що гравець може вибрати якусь із цих двох сукупностей (і лише з них), то дуже *не* схоже на правду, щоб хтось вибирав першу з цих сукупностей суто тому, що $1{,}45 > 1{,}4$. Незрівнянно більш схоже на правду, що першу сукупність обиратимуть лише ті, хто і взагалі нормально ставиться до ризику (та/або кайфує від шансу виграти 1000), і може собі дозволити програти 10; натомість, ті, хто не бажає програвати 10 (байдуже, чи тому, що взагалі не люблять ризик, чи тому, що 10 для них завеликий програш), обиратимуть дру́гу сукупність, не зважаючи на менше матсподівання.

(За великого бажання можна запровадити ще якісь способи порівняння сукупностей варіантів, які враховували б і найгірший варіант кожної з сукупностей, і середній (причому, поняттю «середній» можна надавати різний смисл).

Один з можливих альтернативних підходів: визначити максимальну межу ризику, як-то «розумію, що в умовах невизначеності можу як виграти (додатну величину), так і програти (виграти від'ємну величину); однак, відмовляюся від тих варіантів, де мій програш може вийти більшим деякого A (виграш меншим $(-A)$), й намагаюся максимізувати матсподівання лише серед решти варіантів». Щось подібне (але інакше сформульоване) цілком можна почути від деяких реальних підприємців. Але неясно, наскільки доброю є така ідея: і неясно, звідки брати таке A; і неясно, що робити, якщо неможливо забезпечити ліміт програшу в межах A; і такі обмеження цілком можуть призводити до зменшення матсподівання виграшу (через поспішні «фіксації програшу» там, де дуже висока ймовірність відіграти, але є дуже низька ймовірність вийти за ліміт програшу); і це досить складно — особливо, якщо згадати, що є ще інші гравці: а вони теж будуть запроваджувати такі ліміти програшу? а чужі ліміти будуть загальновідомі, чи секретні? якщо загальновідомі, то чи можуть гравці чогось добитися, обманюючи інших щодо свого ліміту програшу? і так далі…

Так і виходить, що найчастіше розглядають максимізацію матсподівання, нерідко максимізацію мінімума, часом якісь альтернативні підходи/способи/цілі, але з ними все складно.)

1.9.2 Кілька ймовірнісних задач, де максимізують матсподівання

Як щойно згадано, максимізувати са́ме матсподівання — не єдиний підхід, але поширений. Ось і розглянемо кілька задач, де са́ме це і роблять.

Деякі з них є виключеннями з правила «теорія ігор розглядає лише ситуації, де зацікавлених свідомих сторін дві чи більше». Хоч іноді й кажуть, ніби в них «цілком собі грають, просто проти random-а, а не проти свідомого гравця», однак «гра проти random-а» і гра проти свідомого суперника — все-таки різні речі.

Задача 1.9:A. «Плюси й мінуси»

Є a карток, на яких написано "+", та b карток, на яких написано "–"; ці написи є лише з одного боку, а з іншого боку ці картки (всі $a+b$ штук) однакові. Всі ці карткú якось випадково перемішані, й усі лежать догори тією стороною, з якої вони однакові.

Єдиний гравець може бра́ти ці карткú по одній, перевертати й дивитися, чи взяв "+", чи "–". За кожну картку, на якій написано "+", виграш гравця збільшується на 1, а за кожну, на якій написано "–", зменшується на 1, і може ставати від'ємним. Зокрема, якщо забрати всі карткú, то виграш буде $a-b$.

Але гравець не зобов'язаний забирати всі карткú; він має право в будь-який момент (після взяття будь-якої ка́ртки, вже побачивши її позначку, або на са́мому початку, ще нічого не взявши) заявити «закінчую» і припинити гру. Тоді його виграш дорівнює сумі, яку він уже набрав на той момент (включно з останньою взятою карткою, якщо така була).

Яке максимальне матсподівання виграшу може забезпечити собі гравець при правильній грі?

Вхідні дані. У єдиному рядку через пропуск (пробіл) вказано два числа a, b (обидва цілі, з проміжку від 0 до 100) — кількості карток з позначками "+" та "–" відповідно.

Результати. Виведіть єдине дійсне число — матсподівання виграшу для найкращої можливої стратегії єдиного гравця, якщо карткú перемішані випадково, а гравець знає чúсла a, b. Формат виведення може бути будь-яким зі стандартних (зокрема, байдуже, чи вивести `0.5`, чи `0.500000000`, чи `5e-1`), важливо лише забезпечити точність, вказану в «Оцінюванні».

Приклади:

Вхідні дані	Результати
2 0	2
0 2	0
7 50	0
1 1	0.5
7 9	0.299038462

Примітки. У першому прикладі, всі карткú мають позначки "+", їх вигідно забрати всі (обидві). У дру́гому прикладі, всі карткú мають позначки "–", їх вигідно взагалі не бра́ти. У третьому прикладі, мінусів настільки більше, чим плюсів, що теж вигідно взагалі не бра́ти. У четвертому прикладі, вигідно взяти першу картку, далі так:

- якщо на ній "+", то спинитися; виграш буде 1;
- якщо на ній "–", то взяти й наступну (останню), на ній точно буде "+"; виграш буде $(-1)+1=0$.

Ймовірності кожного з цих випадків $\frac{1}{2}$, тому матсподівання $\frac{1}{2}\cdot 1+\frac{1}{2}\cdot 0=\frac{1}{2}+0=\frac{1}{2}$. Також зверніть увагу, що міркування «раз плюсів і мінусів однаково, то й сума буде 0», хоч і може здатися природнім, насправді нічого не дає в цій задачі. Са́ме тому, що в деяких ситуаціях видно, що вигідніше зупинитися й не забирати решту карток. П'ятий приклад надто громіздкий, щоб пояснити число-відповідь детально; але зверніть увагу, що вміння вчасно зупинятися може дати додатне матсподівання виграшу, навіть коли мінусів (трохи) більше, чим плюсів.

Оцінювання. Потестове (кожен тест перевіряють і оцінюють незалежно від решти). Тест зараховують, коли абсолютна або відносна похибка (хоча б одна з двох) не перевищує 10^{-9}.

Розбір задачі. Повторю, що це не зовсім теорія ігор, бо нема свідомого суперника, який прикладає зусиль для свого виграшу. Однак, і сюжет задачі настільки схожий на ігри з числовим виграшем, що багато хто назве це «грою», і це варто розглянути, щоб уміти поєднувати такі засоби із засобами справжніх ігор, коли будуть задачі, де грають водночас і «проти ranom-a», і проти свідомого суперника. А раз так — дозволимо собі вживати словá «виграш», «гравець» та «стратегія гри».

Поєднаймо динамічне програмування з теорією ймовірностей. Поставимо серію підзадач «Яким буде матсподівання виграшу $W(i,j)$ за умови оптимальної стратегії гри, якщо є i карток з позначкою “+” та j карток з позначкою “–”?» (при $0\leqslant i\leqslant a$, $0\leqslant j\leqslant b$).

Тривіальними можна взяти, наприклад, усі підзадачі при $i=0$ та/або $j=0$:

- при $i=0$ нема плюсів, тож нема смислу бра́ти карткú, краще відразу відмовитися від ходів і тим забезпечити $W(0,j)=0$;
- при $j=0$ нема мінусів, варто бра́ти *всі* карткú, й тим забезпечити $W(i,0)=i$.

За цими пункктами виходить однаково $W(0,0)=0$, й це добре. Можна було б зробити інакше, взявши лиш одну тривіальну підзадачу $W(0,0)=0$, але це зробило б заплутанішим основне рівняння для нетривіальних підзадач, тому нехай краще всі вищезгадані будуть тривіальними, а основне рівняння ДП (19) буде лише для $i>0$ та $j>0$.

$$W(i,j)=\max\left\{0,\quad \begin{array}{l}\frac{i}{i+j}\cdot\big(W(i-1,j)+1\big)+\\ +\frac{j}{i+j}\cdot\big(W(i,j-1)-1\big)\end{array}\right\}. \tag{19}$$

Чому це правильно? При $i>0$ та $j>0$ гравець на кожному кроці (зокрема й на са́мому початку) може хоч сказати, що зупиняє гру (подальший виграш становитиме 0), хоч продовжити (подальший виграш слід обчислювати залежно від того, які можуть бути подальші розвитки подій); якщо матсподівання подальшого виграшу додатне, слід продовжити, а якщо від’ємне, слід зупинитися — це пояснює, звідки “$\max\{0,\ldots\}$”.

Якщо продовжувати, то слід взяти одну картку; це випадковий вибір з відомими ймовірностями $\frac{i}{i+j}$ узяти “+” та $\frac{j}{i+j}$ узяти “–”. Якщо (з імовірністю $\frac{i}{i+j}$) це виявилася картка з позначкою “+”, то далі вдасться набрати $W(i-1,j)$ (бо кількість карток з плюсами зменшилася на 1), але крім того ще +1 прямо за́раз. Аналогічно, якщо (з імовірністю $\frac{j}{i+j}$) це виявилася картка з позначкою “–”, то далі вдасться набрати $W(i,j-1)$ (бо кількість карток з мінусами зменшилася на 1), але крім того ще –1 прямо за́раз.

А якщо результат кожного з варіантів помножити на його ймовірність, і всі ці добутки додати — це і є матсподівання.

(Звісно, при цьому важливо врахувати справді всі варіанти, щоб жоден випадок не міг ні потрапити відразу до кількох варіантів, ні не потрапити до жодного, а ймовірності варіантів були правильними, й сума цих ймовірностей дорівнювала 100%, тобто 1. За́раз це так, бо не може бути третього варіанту, крім «витягти “+”» чи «витягти “–”», картка не може бути і “+”, і “–” водночас, імовірності $\frac{i}{i+j}$ та $\frac{j}{i+j}$ випливають безпосередньо з означення «кількість сприятливих поділити на кількість усіх», а для крайнього випадку $i=j=0$ формулу (19) не використовуватимуть. Можна також переконатися, що $\frac{i}{i+j}+\frac{j}{i+j}=\frac{i+j}{i+j}=\frac{1}{1}=1$.)

Тобто, для завершення розв’язування цієї задачі досить запрограмувати формулу (19), з урахуванням вказаних перед тим тривіальних підзадач.

Також про всяк випадок проговорю очевидний факт, що знайдене в цій задачі максимальне матсподівання — лише матсподівання. *Нема* й не може бути абсолютно ніякої гарантії, ніби результат кожної окремо взятої такої гри буде близьким до матсподівання. Діючи згідно з (19), тобто продовжуючи гру лише коли для поточних кількостей карток $\frac{i}{i+j}\cdot(W(i-1,j)+1)+\frac{j}{i+j}\cdot(W(i,j-1)-1)>0$, цілком можна отримати значно гіршу суму окремо взятої гри. Покращення в середньому — вони такі…

Задача 1.9:B. «Гра на максимум суми (1/2/3) з випадковими втратами карток»

N карток викладені в ряд зліва направо. На кожній картці написане ціле число. Два гравці по черзі забирають карткú, причому забирати можна лише з правого боку, або 1 картку, або 2 картки, або 3 картки (але, звісно, не більше карток, чим їх є). Закінчується гра, коли забрано всі карткú (поки хоч одна картка є, гравець зобов'язаний робити один із можливих ходів). Крім того, після кожного ходу кожного з гравців з імовірністю 10% стається (отже, з імовірністю 90% не стається) випадкове віднесення вітром картки, яка щойно стала крайньою правою. (Це *найголовніша* відмінність цієї задачі від задачі 1.7:A.) Мета гри — отримати якнайбільшу суму (чисел, записаних на забраних картках).

Які матсподівання сум, що їх наберуть гравці, якщо обидва гратимуть якнайкраще, намагаючись максимізувати кожен своє матсподівання?

Вхідні дані. У першому рядку вказано кількість карток N ($1 \leqslant N \leqslant 1234$). У другому рядку через пробіли задані N цілих чисел (що не перевищують за модулем 10^3; інакше кажучи, з проміжку $[-1000; +1000]$) — значення, записані на картках.

Результати. Виведіть в одному рядку через пропуск два дійсні числá — матсподівання сум, що наберуть гравці (спочатку 1-й, потім 2-й). Формат виведення може бути будь-яким зі стандартних (зокрема, байдуже, чи вивести `41.91`, чи `41.910000000`, чи `4.191e+1`), важливо лише забезпечити точність, вказану в «Оцінюванні».

Приклади:

Вхідні дані	Результати
7 11 11 11 11 11 11 11	41.91 33
8 3 2 999 4 6 7 -5 1	810.17 116.2
9 23 -17 2 -13 5 3 11 -7 19	26.7633 2.215

Примітки. У першому тесті, числа додатні й однакові, тому гравцям вигідно забирати якнайбільше карток: спочатку 1-й забирає три штуки; потім вітер чи то відносить одну картку, чи то ні, й карток лишається чи то три штуки, чи то чотири; в будь-якому разі, 2-й забирає три штуки; залежно від вітру, повторний хід 1-го гравця може бути, а може й не бути: він буде з ймовірністю $0{,}9 \cdot 0{,}9 = 0{,}81$, якщо вітер не віднесе картки ні після ходу 1-го гравця, ні після ходу 2-го; в цьому разі, лишиться одна картка, яку вигідно забрати. Тобто, 1-й з імовірністю 81% набере $33 + 11 = 44$, а з імовірністю $100\% - 81\% = 19\%$ набере 33; матсподівання дорівнює $0{,}81 \cdot 44 + 0{,}19 \cdot 33 = 35{,}64 + 6{,}27 = 41{,}91$. Для цих вхідних даних, вітер не може завадити 2-му гравцеві взяти три картки по 11 кожна, й результат 2-го завжди $11+11+11 = 33$. (Від вітру залежить, *які* це будуть три картки; але на кожній з них однакове 11, тому на результат не впливає.)

У дрýгому прикладі, як і в задачі 1.7:A, значення 999 усе ще настільки перевищує всю решту, що все ще варто дбати в першу чергу про те, щоб узяти це найбільше число. Через те, що тепер є випадкові впливи вітру, перший хід «узяти дві картки 1 та (–5)» вже не гарантує цього, але все ще дає найбільшу ймовірність $0{,}9 \cdot 0{,}9 = 0{,}81$, якщо вітер не віднесе картки ні після ходу 1-го гравця, ні після ходу 2-го.

Як від ще детальніших пояснень відповіді дрýгого тесту, так і від будь-яких коментарів щодо третього тесту утримаюся.

Оцінювання. Потестове (кожен тест перевіряють і оцінюють незалежно від решти). Тест зараховують, коли абсолютна або відносна похибка (хоча б одна з двох) не перевищує 10^{-9}.

Розбір задачі. Хоч ця задача й схожа на задачу 1.7:A за дуже багатьма ознаками, перший спосіб розв'язування 1.7:A (який включає рівняння ДП (12)) абсолютно непридатний для модифікацій під теперішню задачу: тоді якнайістотнішим чином використали, що вся сума розділяється між двома гравцями, а тепер це неправда.

А от «дру́гий альтернативний розв'язок» 1.7:A (який включає переходи (17)) перетворити можна; його поєднання з формулами теорії ймовірності (схожими на вжиті в попередній задачі 1.9:A), хоч і дещо громіздке, ідейно більш-менш очевидне.

Нехай позиції будуть ті са́мі «лишилось i карток, де $0\leqslant i\leqslant N$ (включно)», для кожної такої позиції так са́мо знаходимо пару (a,b), де $p(i).a$ виражає матсподівання суми, яку набере при правильній грі обох гравців той, кому дісталася позиція «лишилось i карток», а $p(i).b$ виражає матсподівання суми, яку набере при цьому його суперник.

Які варто виділити тривіальні підзадачі? Точно треба $p(0)=(0;0)$, тобто коли карток нема, жоден з гравців ніц не набере. Ще варто включити до тривіальних підзадач $p(1)=(c(0);0)$[7], тобто коли картка одна, її забирає той гравець, якому дістається ця позиція, інший не отримує нічого, й на все це ніяк не впливає випадок (вітер). Також важливо, що для $i\geqslant 2$ вже треба враховувати можливість «картку забрати можна, але не вигідно (наприклад, на ній від'ємне число)», й тому це краще включити в основне рівняння динпрога, а для $i=1$ такого вибору нема, бо єдину картку треба забрати як «хоча б одну».

Для $i\geqslant 2$, що відбувається, якщо гравець вирішує забрати одну картку? З імовірністю 90%, супернику дістанеться позиція з $i-1$ карток, а з імовірністю 10% позиція з $i-2$ карток (бо одну зідме вітер). Оскільки в цій задачі гра неупереджена, то з причин, пояснених у все тому ж пункті «дру́гий альтернативний ро́зв'язок …» (щоправда, щодо (16), бо (17) там дане майже без пояснень) при такому переході виграш 1-го гравця визначається через виграш 2-го і навпаки. Таким чином, вибір гравця «забрати одну картку» дає матсподівання виграшу

$$\begin{matrix}\text{зараз}\\ \downarrow\end{matrix}\qquad \begin{matrix}\text{матсподівання подальших кроків}\\ \downarrow\end{matrix}$$
$$\begin{matrix}.a\to\\ .b\to\end{matrix}\quad\begin{pmatrix}c(i{-}1)+(0{,}9\cdot p(i{-}1).b+0{,}1\cdot p(i{-}2).b),\\ (0{,}9\cdot p(i{-}1).a+0{,}1\cdot p(i{-}2).a)\end{pmatrix}. \tag{20}$$

Це ще не $p(i)$, а лиш оцінка одного з можливих ходів «забрати одну картку». Ще треба розглянути аналогічні вирази для ходів «забрати дві картки» та «забрати три картки», і з усіх можливих варіантів вибрати максимум. Причому, са́ме з *можливих*; спільна біда (12), (15) та (17), що треба перевіряти, які з ходів можливі, а які ні (бо замало карток) тепер тільки посилилася (бо додалися ще стохастичні (ймовірнісні) варіанти «вітер (віднесе/не віднесе) картку»).

Тим не менш, пропоную читачам зробити цей етап самостійно. Відзначу лише, що запровадження двох тривіальних підзадач $p(0)=(0;0)$ та $p(1)=(c(0);0)$ (замість само́ї лише $p(0)=(0;0)$ й обчислення $p(1)$ як нетривіальної) дає можливість хоч трохи спростити ці марудні перевірки.

Приклад аналіза дерева рішень із імовірнісними розгалуженнями. Розгляньмо (без глибокої теорії та без задачі на написання програми) приклад, де поєднуються ймовірнісна невизначеність і кількакрокові ігри на дереві рішень з розд. 1.8. Нехай є окремо вершини, де, як у розд. 1.8, рішення залежить лише від відповідного гравця, й окремо вершини, де переходи відбуваються суто випадково. Наприклад, унизу стор. 111 наведено спочатку приклад умови такої гри, потім її розв'язок. Як і раніше, сині стрі́лки й вершини відповідають рішенням 1-го гравця, магентові (темночервоні) — 2-го, але ще є зеле́ні, де замість свідомого вибору гравцем робиться випадковий вибір з відомими ймовірностями.

Процес розв'язування такої гри (якщо треба «максимізувати матсподівання, не переймаючись іншими статистичними характеристиками») досить очевидний для уважних читачів: так са́мо, як у розд. 1.8, починаємо від найдальших від кореня «листків», і рухаємося до початкової позиції; для «синіх» (хід 1-го) та «магентових» (хід 2-го) проміжних вузлів, так са́мо, як у розд. 1.8, вибираємо серед вузлів-«синів» оцінку, найкращу з точки зору того гравця, який приймає рішення в цьому вузлі, й оголошуємо її оцінкою цього вузла; для «зелених» (випадковий вибір) проміжних вузлів рахуємо матсподівання так са́мо, як у попередніх задачах поточного розд. 1.9.2, на основі відомих

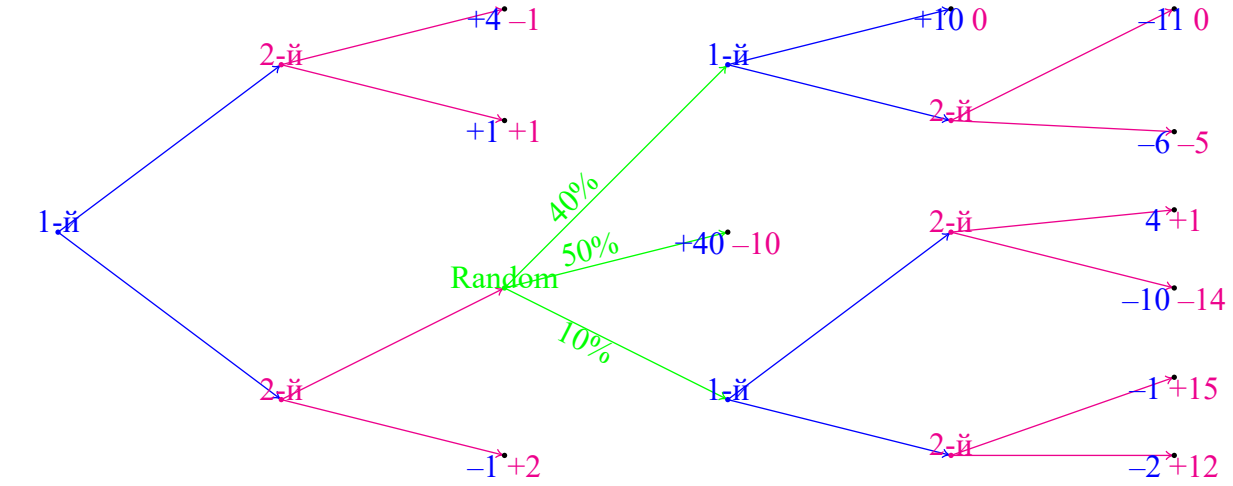
[7] вважаючи, як і в (17), що в масиві p індекси від 0 до N, а в масиві c індекси від 0 до $N-1$, тож оцінка $p(i)$ (при $i>0$) стосується позиції «лишилися картки з 0-ї по $(i-1)$-у включно»

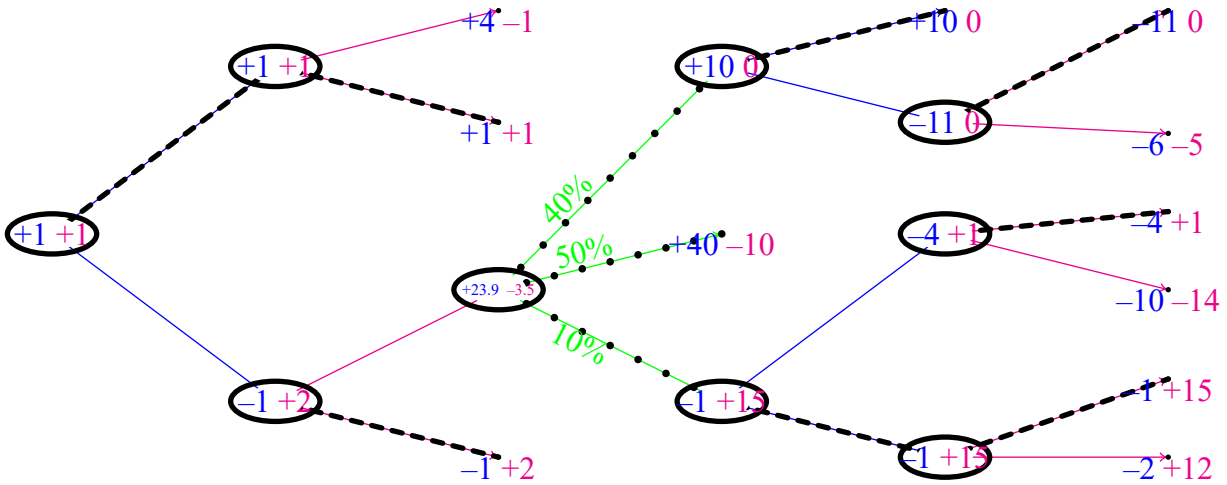
імовірностей та відомих оцінок вузлів-«синів», і при цьому байдуже, чи ті вузли-«сини» є «синіми», чи «магентовими», чи «зеленими», чи все це якось змішано.

1.9.3 Вплив несиметричної інформації. Стаття «ринок “лимонів” та ринок “персиків”»

У цьому розд. 1.9.3 я переповідаю своїми словами розповідь про статтю Akerlof, George A. (1970). “The Market for ’Lemons’: Quality Uncertainty and the Market Mechanism”. Quarterly Journal of Economics. 84 (3). The MIT Press: 488–500. Однак, оскільки легальний доступ до цієї статті платний, я сам читав&дивився лише її перекази та розповіді про неї (головним чином, [33, 35, 36, 37]). Через це, враховуючи рішення включати у список джерел лише фактично доступні, виходить той сумний факт, що список джерел не містить згаданого кілька рядків тому першоджерела. (Якщо раптом хтось знайде, як отримати самý цю статтю-першоджерело легально й безкоштовно — прошу повідомити.)

Розглянемо таку модель перепродажу вживаних автомобілів. Нехай кожен вживаний автомобіль є або «лимоном» (автомобілем з прихованими дефектами; са́ме прихованими, а не «взагалі не їздить»), або «персиком» (автомобілем, стан якого, як на його вік, добрий). Нехай уявлення продавця й покупця про ціни на ці види автомобілів такі:





Конкретні значення $(+23{,}9; -3{,}5)$ у єдиному «зеленому» вузлі цього прикладу рахуються так:

$$0{,}4\cdot 10 + 0{,}5\cdot 40 - 0{,}1\cdot\ 1 = 4 + 20 - 0{,}1 = \ 23{,}9$$
$$0{,}4\cdot\ 0 - 0{,}5\cdot 10 + 0{,}1\cdot 15 = 0 - \ \ 5 + 1{,}5 = -3{,}5$$

	«лимон»	«персик»
продавець згоден продати за…	від K$1	від K$5
покупець згоден купити за…	до K$2	до K$7

(де “K$” — «кілодолари», тобто тисячі доларів).

Можуть бути три ситуації щодо рівня інформованості сторін про те, чи конкретний автомобіль є «лимоном», чи «персиком»:

- обидві сторони знають;
- жодна сторона не знає;
- продавець знає, а покупець ні.

(Варіант «покупець знає, а продавець ні» зазвичай відкидають, як нереалістичний: продавець вже мав справу са́ме з цим автомобілем, покупець ні. Хоча, в разі бажання, можете розглянути цей варіант самостійно по аналогії з «продавець знає, а покупець ні».)

Далі пропонується розглянути кожну з цих трьох ситуацій окремо.

(В цій моделі є деякі спрощення й неточності (а пізніше їх стане ще більше). Дехто через це навіть відмовляється сприймати її серйозно. Джере́ла [33, 38] стверджують, що вищезгадану статтю Дж. Акерлофа тричі відмовлялися приймати до друку в наукових журналах (причому причини відмови були дуже різними; якщо перекласти їх з наукової мови на розмовну, то від «це якісь занадто очевидні міркування студентського рівня, не варті публікації в солідному науковому журналі» до «все це неправда, бо якось же продають ті вживані автомобілі»), потім її нарешті опублікували, потім усвідомили, що підняті в ній проблеми важливі, а ще через ≈30 років групі з трьох вчених (цей самий Джорж Акерлоф, а також Майкл Спенс і Джозеф Стігліц) дали Нобелівську премію з формулюванням (цитата згідно [39]) «За їх аналіз ринків з асиметричною інформацією». Тобто, не за само́ статтю “The Market for ’Lemons’: Quality Uncertainty and the Market Mechanism”, але за дослідження, які почалися са́ме з цієї статті.

Тож, *що* неприроднє (чи здається таким) у цій моделі?

Неясно, що це взагалі за суми грошей і звідки вони взялися? Це правда, але так уже склалося, що в такого роду статтях з теорії ігор частенько розглядають суто приклади, тож питання більше в тому, чим обґрунтована (й чи обґрунтована взагалі) адекватність цих чисел. Розгляньмо, чому тут вони більш-менш обґрунтовані. Добровільні нерегульовані купівлі-продажі відбуваються (а не скасовуються з криком «ну, я міг би (продати/купити), але ж не за таку ціну!») тоді, коли об’єкт торгівлі покупцеві потрібніший, чим продавцеві, й знаходиться така сума, за яку продавець згоден продати, а покупець згоден купити. Водночас, покупцю було б приємно купити дешевше, а продавцю — продати дорожче. Тому багатьом (але не всім) фактам купівлі-продажу притаманно, що покупець в принципі міг би погодитися й на трохи вищу ціну, а продавець — на трохи нижчу (наскільки — залежить від того, наскільки добре сто́рони вміють торгуватися, й детальніше не розглядається). Оголошені суми грошей (як «від K$1» й «до K$2» для «лимона», так і «від K$5» і «до K$7» для «персика») підібрані так, щоб було безсумнівно, що простір взаємоприйнятних цін ϵ, але ***окремо*** свій простір «від K$1 до K$2» для «лимонів», свій «від K$5 до K$7» для «персиків». А розібратися, що буде, коли ніби й є можливість для купівлі-продажі за взаємовигідною ціною, але є також і залежність ціни від якості — це значно цікавше хоч за ситуації, де очевидний результат «ну, я міг би (продати/купити), але ж не за таку ціну!», хоч за ситуації, де очевидний результат «легко знайти ціну, узгоджена ціна визначається легко, й однакова для автомобіля будь-якої якості».

Не вказані модель та вік автомобіля? Просто мається на увазі, що ці параметри вже зафіксовані; звісно, для іншої моделі та/або іншого року випуску ціни будуть іншими — ну то нехай собі будуть, міркування правильні для кожного з випадків, де теж є окремий простір для взаємовигідних цін на «лимони» й окремий для «персиків», але нема одного спільного.

Якість вживаних автомобілів точніше вимірювати числом, яке може набувати багато значень (чи навіть кількома числами), а не зводити до бінарного вибору лише з варіантів «лимон»/«персик»? Що ж, це важливіша (чим попередні) претензія. Справді, не всі висновки, зроблені щодо бінарного вибору, будуть правильними також для вибору з багатьма проміжними градаціями. Про це варто пам’ятати. Але, по-перше, аналіз проблем часто починають з простішого випадку, лиш потім переходячи до складніших. По-друге, автомобілі тут є лише прикладом; для якихось інших виробів якість може бути значно ближчою до бінарної «чудово» / «поганенько, але якщо дешево, то змиритися можна». По-третє (цього не можна знати наперед, ще не зробивши аналіз; але коли Дж. Акерлоф вже написав статтю, редактори журналів та рецензенти мали дочитати її до кінця й узнати це), са́ме в цьому випадку остато́чний висновок описує певні проблеми, наявні навіть у простому випадку «лимонів» та «персиків»; природньо, що від того, що до цих категорій буде додана купа проміжних, проблеми навряд чи зникнуть.

Проміжки «від K$1 до K$2» та «від K$5 до K$7» якісь занадто широкі, значно природніше було б, наприклад, «від $1100 до $1200» та «від $5800 до $6000»? Широкі проміжки взяті свідомо, щоб показати: проблеми остато́чного висновку спричинені *не* тим, що продавці та/або покупці занадто непоступливі.)

То що з оголошеними трьома ситуаціями?

Ситуація «обидві сторони знають». Це найпростіша ситуація. Знаючи, що авто є «лимоном», сторони зійдуться на ціні десь між K$1 і K$2 (наприклад, на K$1,5). Знаючи, що авто є «персиком», сторони зійдуться на ціні десь між K$5 і K$7 (наприклад, на K$6). (Як уже сказано, конкретна сума залежить від умінь сторін торгуватися.)

(Також варто відзначити, що якщо обидві сторони притомні, то в рамках цієї ситуації цієї моделі вони нізащо не зійдуться ні на якій ціні в проміжку між $2001 та $4999: для «лимона», покупець відмовиться стільки платити, а для «персика» продавець відмовиться за стільки продавати.)

Ситуація «жодна сторона не знає». Цю ситуацію ускладнює вже згадане раніше питання прийнятності ризику. Хоч далі й показано, що «у середньому, прийнятно продати&купити таке авто за, наприклад, K$4», але ж «конкретний покупець, якому дістався “лимон”, переплатить K$2 (аж удвічі!) відносно максимуму своїх початкових уявлень», а «конкретний продавець, який втратить “персика”, недоотримає K$1 відносно своїх». І, з одного боку, це правда. З іншого — «якщо вони бояться таких наслідків, то хай витратять гроші й зусилля, щоб узнати, а не продають/купують “кота в мішку”; якщо ж витрачати гроші й прикладати зусилля не хочуть, то хай миряться з суто статистичним результатом, без гарантій для окремого випадку».

Якщо, проігнорувавши всі «але ж» і «а раптом», зосередитися суто на матсподіваннях і припустити, що при випадковому виборі автомобіля ймовірність «персика» складає p (відповідно, ймовірність «лимона» $(1-p)$), усе гаразд. Продавець готовий продавати за ціну (в K$) від $p\cdot 5+(1-p)\cdot 1 = 1+4p$. Покупець згоден платити до $p\cdot 7+(1-p)\cdot 2 = 2+5p$. Враховуючи $p>0$, кожен доданок суми $1+4p$ менший за відповідний доданок суми $2+5p$, й теж є можливість домовитися про взаємоприйнятну ціну десь у цьому проміжку (наприклад, при $p=0{,}5$ це буде між $1+4\cdot 0{,}5=3$ та $2+5\cdot 0{,}5=4{,}5$, куди входить, наприклад, 4).

(*Примітка 1.* Це так, ***якщо*** обидві сторони мають однакове уявлення про p; якщо ж, наприклад, продавець вважає, що $p=0{,}9$, а покупець, що $p=0{,}1$, то ніц не вийде, бо продавець згоден продавати лише від $1+4\cdot 0{,}9=4{,}6$, покупець згоден платити лише до $2+5\cdot 0{,}1=2{,}5$, й угоди не буде. (Втім, точна рівність уявлень про p не обов’язкова, взаємна згода може бути й при *трохи* різних уявленнях про p.)

Примітка 2. Як вже сказано, важливі не конкретні чи́сла 1, 2, 5, 7, а факт, що і для «лимона», і для «персика» є проміжок цін, де продавець вже згоден продавати, а покупець ще згоден купляти. Пропоную охочим читачам, замінивши 1 на a, 2 на b, 5 на c, 7 на d, і спершися лише на «$a\leqslant b$», «$c\leqslant d$» та «сто́рони мають однакове уявлення про p» довести, що проміжок матсподівань, в якому продавець вже згоден продавати, а покупець ще згоден купляти, буде непорожнім, яке б не було (з осмисленого для ймовірностей проміжку $0\leqslant p\leqslant 1$) однакове для обох сторін p. Тут нема нічого складно́го, досить скористатися шкільними алгебраїчними засобами перетворення нерівностей.)

Ситуація «продавець знає, а покупець ні» — головна й найцікавіша. Для цієї ситуації додатково доуточнимо модель гри. Нехай це буде послідовна гра, в якій спочатку покупець пропонує ціну, потім продавець погоджується чи відмовляється:

- якщо погоджується, то факт купівлі-продажу відбувається за цією ціною (позначимо її як s (K$)), причому виграш покупця — те, наскільки s менша за максимальну суму, яку він згоден платити, а продавця — те, наскільки s більша за мінімальну суму, за яку він згоден продати;
- якщо відмовляється — намір купівлі-продажу скасовується, виграші обох сторін = 0.

Ну і найголовніше:

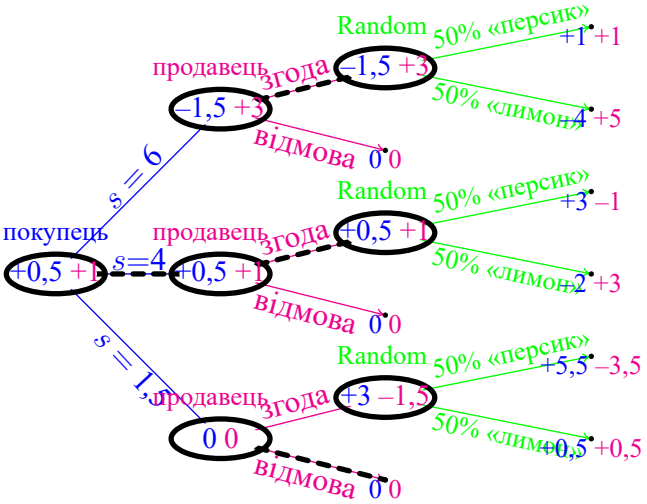
- продавець (який знає, але не повідомляє, за «лимон» чи за «персик» торгуються), приймає рішення, враховуючи це знання;
- покупець (який того не знає), приймає рішення, вважаючи задачу випадковою, міркуючи в термінах максимізації матсподівання, й визначивши для себе деяку ймовірність p, що предмет торгу є «персиком»; нехай $p=0{,}5$.

(Це уточнення теж можна критикувати. Починаючи з того, що чому раптом ціну називає покупець. І це теж одна з причин, чому цю статтю не всі сприймають серйозно. Однак… (1) Са́ме для *вживаних* речей такий формат торгу має право на існування; (2) Можна уявити, ніби той хід робить не покупець, а торг між покупцем і продавцем. Просто, торг часто багатокроковий і має ризик зациклювань, а ми хочемо спростити модель, втиснувши його в один хід. Тоді, думка, що торг остато́чно завершується згодою чи відмовою са́ме продавця, більш-менш природня.

Припущення $p=0{,}5$ теж дуже дискусійне, але це досить стандартне припущення для ситуацій, коли взагалі нічого не відомо. Крім того, пізніше будуть наведені деякі міркування якраз щодо наслідків змін p.

Але якщо говорити не про значення p, а про факт наявності якогось p — це *ніяке не додаткове* припущення, а головна особливість ситуації, яку вирішив розглянути Дж. Акерлоф: продавець враховує, чи він продає «лимон», чи «персик», а покупець ставиться до ситуації як до стохастичної (ймовірнісної).)

Можливих значень s вельми багато, тож, розглядаючи всі, отримали б занадто розгалужене дерево рішень. Тому, обмежимося варіантами K\$1,5, K\$4 та K\$6, бо са́ме вони були названі як приклади взаємоприйнятних для ситуацій «обоє знають, що “лимон”», «обоє думають, що “персик” чи “лимон” з шансами 50%:50%» та «обоє знають, що “персик”» відповідно. Тоді, очікування покупця можна описати таким деревом рішень:



(воно наведене відразу розв’язаним, позначення цілком аналогічні сторінкам 96 та 111–110). Тобто, покупець (якщо, звісно, він погодився ризикнути й змирився, що результат є статистичним, а не гарантованим) вважатиме найдоречнішим пропонувати середню ціну, вважаючи, що так і продавцеві в середньому вигідно, і є шанс купити «персика».

(Це знову момент, який потенційно можна критикувати: щоб покупець міг дійти висновку, що в умовах невизначеності, ймовірнісної моделі та $p=0{,}5$ продавець відмовиться продавати за K\$1,5, але погодиться за K\$4, покупцю ніби як потрібно знати, що продавець згоден продати «лимона» за K\$1 і «персика» за K\$5. Що ж, на це можна відповісти, що замість точних сум покупець може використати приблизні, взяті з якихось загальнодоступних джерел (скільки хочуть інші продавці, скільки пропонують інші покупці, за скільки реально відбувалися інші факти купівлі-продажу, тощо). До речі, са́ме тому тут не варто розглядати багато значень s і намагатися підібрати мінімальну з цін, при якій купівля-продаж вже відбудеться (а нижче неї продавець відмовиться): для визначення такої межі́ справді потрібно знати, за скільки продавець готовий продавати.)

А що думатиме продавець? У нього нема є́диного де́рева гри, яке містить і «лимона», й «персика». Для нього є окреме уявлення «“лимона” варто продавати за будь-яку ціну, більшу K\$1», й окреме «“персика” варто продавати за будь-яку ціну, більшу K\$5». Тобто, якщо покупець запропонує K\$4 (чи на цій сумі скінчиться торг), то продавець продасть «лимона» й відмовиться продавати «персика».

Тобто, нібито «компромісний» і «виправданий у середньому» варіант середньої ціни не працює. Покупець отримає за нього лише «лимона», якого міг би отримати й за значно нижчу ціну.

Чи допоможе, якщо покупець пропонуватиме не K\$4, а K\$6? Навряд: шанс усе-таки отримати «персика» ніби з’являється, але ж і ціна зростає, і ризик сильно (ще сильніше!) переплатити за «лимона» лиш трохи зменшується, не щезаючи й навіть не роблячись дуже малим.

Чи допоможе, якщо покупець будуватиме дерево рішень з імовірнісними ходами, спираючись не на ймовірність персика $p=0{,}5$, а на, наприклад, $p=0{,}2$, і відповідним чином змінить середню пропозицію ціну (або додасть до варіантів пропозицій ціни цю четверту)? Можливо, й допоможе менше переплатити за «лимона», але геть зовсім не допоможе купити «персика» (раз продавець відмовляється продавати його за K\$4, то тим паче відмовиться продавати його за ще меншу ціну).

Чи допоможе, якщо покупець прямо спитає продавця про якість автомобіля? Це переплетено з питаннями «чи засуджує суспільство брехню на переговорах?» та «чи цінує продавець свою репутацію?». Якщо не цінує (або суспільство вважає, що в таких ситуаціях вводити в оману можна), то відповідь очевидна: «не допоможе, бо продавцеві вигідно казати на “лимон”, ніби він “персик”, і

ставити за нього завищену ціну». Більш того: продавець може й не брехати явно, але замість відповіді жартувати чи казати «ой, я теж не знаю» (коли знає, що авто є «лимоном»).

(Розмови про репутацію можуть завести далеко. Є надто різні свідчення як про «надзвичайну важливість репутації в сучасних бізнес-елітах глобального Заходу», так і про «немає такого злочину, на який капітал не ризикнув би заради 300% прибутку», а також «nothing personal, it's just business». Чи згадував про вплив репутації на такі ситуації сам Дж. Акерлоф, я, на жаль, не знаю. (Нагадаю, що мені не вдалося знайти безкоштовно (а платно й не шукав) оригінальний текст само́го Дж. Акерлофа, лише його вільні перекази.) Тому я відмовляюся обговорювати питання репутації детальніше, лише констатую, що описано випадок, коли ніщо не перешкоджає продавцеві обманювати покупця.

А ще… Теорія ігор зазвичай виходить із припущення, що гра обмежена суто її правилами, й нема ніякої репутації, яка на це впливає та/або на яку впливають ходи у грі. Власне, у грі «Ультиматум» (розд. 1.8.3) наслідки, які більш-менш пояснюються якраз побоюванням погіршити репутацію, виставлялися як парадокс і невідповідність між теорією ігор і реаліями життя.)

Які з цього всього можна зробити висновки? Є глибокі причини для не просто можливості, а й природності ситуацій, коли продавець обманює покупця, який не може перевірити якість товару. Й це проблема. Але ще гірше, що це не єдина проблема. Нехай є продавець, який має «персика» й хоче його продати, але лише за гідну «персика» ціну. А покупець вже мав негативний досвід (можливо, з іншими продавцями), й тепер не вірить, що продають са́ме «персика». В такого продавця нема справді добрих засобів переконати такого покупця… принаймні, лишаючись у рамках обмежень «купівлю-продаж укладають суто продавець і покупець», «покупець не може перевірити якість товару» та «відразу після купівлі-продажу (або відмови укладати купівлю-продаж) взаємодія продавця й покупця завершується».

Тому, навіть якщо суспільний лад бажає ринку, вільної конкуренції й мінімізації втручань держави — са́ме за́раз маємо той випадок, в якому підхід «ринок сам усе повирішує» працює украй погано, й суспільно (не лише державно, а й суспільно!) бажаними є деякі додаткові засоби, які допомагали б тим покупцям, які не є спеціалістами, отримати хоч якусь інформацію про будь-який автомобіль; це зменшить можливості продавців обманювати чи приховувати інформацію, й таким чином пом'якшить описану проблему.

Само собою, можуть бути й інші засоби врегулювання цієї ситуації. Але у переказах цієї статті найчастіше згадується са́ме цей.

А ще, дозволю собі процитувати, як головні висновки з цієї ж статті Дж. Акерлофа сформулював (у [35]) проф. О. П. Ігнатенко: «Асиметрична інформація руйнує ринок "персиків". Їх неможливо купити, неможливо продати (для раціональних гравців)[8], залишається тільки ринок "лимонів". Це явище отримало назву лимонізація ринку. … Після публікації роботи стало зрозуміло, що процеси лимонізації зустрічаються у багатьох областях. … Одним із способів боротьби з лимонізацією ринку вживаних автомобілів став carfax — база даних з історією всіх вживаних автомобілей Америки.».

Список джерел

Автор посібника не зміг знайти іншого посібника, підручника чи монографії, де викладався б увесь потрібний (на думку автора) матеріал. (Власне, це й було причиною писати цей посібник.) Враховуючи відсутність чітко вираженого обмеженого кола основних джерел, було прийняте свідоме рішення віддавати перевагу легкодоступним через Інтернет джерелам (іноді, за рахунок вибору не найкращих у плані фундаментальності та/або широти охоплення, але неможливо поєднати відразу всі позитивні риси).

[1] Online Judge https://dmoj.cdu.edu.ua, що дозволяє автоматичну перевірку більшості задач, наведених у цьому посібнику. Перевірено: 24.04.2025.

[2] Теорія ігор (вікіпедія) https://uk.wikipedia.org/wiki/Теорія_ігор Перевірено: 26.02.2024.

[8] тобто, продавців, які не хочуть продавати їх собі у збиток — примітка автора посібника

[3] Матрична гра (вікіпедія) https://uk.wikipedia.org/wiki/Матрична_гра Перевірено: 26.02.2024.

[4] Послідовна гра (вікіпедія) https://uk.wikipedia.org/wiki/Послідовна_гра Перевірено: 27.02.2024.

[5] Комбінаторна теорія ігор (вікіпедія) https://uk.wikipedia.org/wiki/Комбінаторна_теорія_ігор Перевірено: 26.02.2024.

[6] Гра з повною інформацією (вікіпедія) https://uk.wikipedia.org/wiki/Гра_з_повною_інформацією Перевірено: 09.03.2024.

[7] Граф_(математика) (вікіпедія) https://uk.wikipedia.org/wiki/Граф_(математика) Перевірено: 26.02.2024.

[8] Порубльов І. М. Дискретна математика. Навчальний посібник для студентів 1-го курсу бакалаврату галузі знань «Інформаційні технології» та споріднених. // Черкаси: видавець ФОП Гордієнко Є. І., 2018 — 220 с. Доступна у репозитарії ЧНУ за посиланням https://eprints.cdu.edu.ua/4177/1/paryblev2018%20%281%29.pdf (перевірено: 22.03.2024).

[9] Цикл (теорія графів) (вікіпедія) https://uk.wikipedia.org/wiki/Цикл_(теорія_графів) Перевірено: 29.03.2024.

[10] Спрямований ациклічний граф (вікіпедія) https://uk.wikipedia.org/wiki/Спрямований_ациклічний_граф Перевірено: 23.03.2024.

[11] Індукція назад (вікіпедія) https://uk.wikipedia.org/wiki/Індукція_назад Перевірено: 23.03.2024.

[12] Неупереджена гра (вікіпедія) https://uk.wikipedia.org/wiki/Неупереджена_гра Перевірено: 20.04.2024.

[13] Упереджена гра (вікіпедія) https://uk.wikipedia.org/wiki/Упереджена_гра Перевірено: 20.04.2024.

[14] Theory of Impartial Games https://web.mit.edu/sp.268/www/nim.pdf Перевірено: 10.03.2024.

[15] Game Theory. Class notes for Math 167, Fall 2000 by Thomas S. Ferguson https://www.cs.cmu.edu/afs/cs/academic/class/15859-f01/www/notes/comb.pdf Перевірено: 26.03.2024.

[16] Ілля Порубльов. Динамічне програмування // У збірнику: Міжнародний конкурс з інформатики та комп'ютерного мислення «Бебрас–2018». Матеріали школи програмування (Львів-2018) / С. С. Жуковський, І. М. Порубльов, І. В. Скляр, Р. С. Шпакович — Кам'янець-Подільський, «Аксіома», 2019 — с. 50–97. Доступна у репозитарії ЧНУ за посиланням https://eprints.cdu.edu.ua/6047/1/bebras-2018_29-07%20%281%29.pdf, а також за посиланням https://drive.google.com/file/d/1CeqO3lxERehEqknJZCjeh-vpgVDSmW5W/view (перевірено: 19.03.2024).

[17] C. L. Bouton, Nim, a game with a complete mathematical history, Ann. Math., Princeton (2), 3(1902), pp. 35–39. Копія тексту (перенабрана в іншому форматуванні) доступна за посиланням https://en.wikisource.org/wiki/Nim,_A_Game_with_a_Complete_Mathematical_Theory (перевірено: 31.03.2024).

[18] Виключна диз'юнкція (вікіпедія) https://uk.wikipedia.org/wiki/Виключна_диз'юнкція Перевірено: 22.03.2024.

[19] Mex (mathematics) (wikipedia) https://en.wikipedia.org/wiki/Mex_(mathematics) Перевірено: 22.03.2024.

[20] Множина (вікіпедія) https://uk.wikipedia.org/wiki/Множина Перевірено: 22.03.2024.

[21] Функція Шпрага — Гранді (вікіпедія) https://uk.wikipedia.org/wiki/Функція_Шпрага_—_Гранді Перевірено: 23.03.2024.

[22] Декартів добуток (вікіпедія) https://uk.wikipedia.org/wiki/Декартів_добуток Пе-

ревірено: 23.03.2024.

[23] Gundy's game (wikipedia) https://en.wikipedia.org/wiki/Grundy's_game Перевірено: 24.03.2024.

[24] Порубльов І. М. Щодо можливості узагальнень засобів аналізу неупереджених комбінаторних ігор на упереджені. // Інформаційні моделюючі технології, системи та комплекси (ІМТСК-2023). IV міжнародна науково-практична конференція. Збірка тез — с. 31–34. Доступна у репозитарії ЧНУ за посиланням https://eprints.cdu.edu.ua/5955/1/Book_IMTCK_2023-31-34.pdf, а також за посиланням https://fotius.cdu.edu.ua/wp-content/uploads/2023/06/Book_IMTCK_2023.pdf#page=31 (перевірено: 26.03.2024).

[25] Games on arbitrary graphs (стаття на сайті) https://cp-algorithms.com/game_theory/games_on_graphs.html (перевірено: 13.04.2025).

[26] Minimum Cost Polygon Triangulation (стаття на сайті) https://www.geeksforgeeks.org/minimum-cost-polygon-triangulation (перевірено: 14.04.2024).

[27] Порубльов І. М., завдання для виконання на папері з дисципліни «Алгоритми та структури даних», тема «Динамічне програмування» (на правах робочого матеріалу, що не проходив офіційну рекомендацію до друку). https://drive.google.com/open?id=1O1CpOdm9v4hL-tdtArczcoBBHf-s3chz (перевірено: 14.04.2024).

[28] Дерево (теорія графів) (вікіпедія) https://uk.wikipedia.org/wiki/Дерево_(теорія_графів) (перевірено: 04.05.2024).

[29] Tree (graph teory) → Rooted tree (wikipedia) https://en.wikipedia.org/wiki/Tree_(graph_theory)#Rooted_tree (перевірено: 04.05.2024).

[30] Ігнатенко О., Теорія ігор: моделі переговорів, ігри «ультиматум» і «диктатор» (стаття на сайті) https://site.ua/olexii.ignatenko/teoriya-igor-modeli-peregovoriv-igri-ultimatum-i-diktator-i09nkey (перевірено: 04.05.2024).

[31] Ігнатенко О., Деякі нотатки про рівні знання та зворотну індукцію (стаття на сайті) https://site.ua/olexii.ignatenko/deyaki-notatki-pro-rivni-znannya-ta-zvorotnu-indukciyu-i7xjvle (перевірено: 04.05.2024).

[32] Ultimatum game (wikipedia) https://en.wikipedia.org/wiki/Ultimatum_game (перевірено: 03.05.2024).

[33] Ігнатенко О., Теорія ігор. Ігри з неповною і недосконалою інформованістю (відео) https://youtu.be/i9l27nge3Ms (перевірено: 07.05.2024).

[34] Математичне сподівання (вікіпедія) https://uk.wikipedia.org/wiki/Математичне_сподівання (перевірено: 08.08.2024).

[35] Ігнатенко О., Ринок «лимонів», теорія ігор і проблеми науки (стаття на сайті) https://site.ua/olexii.ignatenko/rinok-limoniv-teoriya-igor-i-problemi-nauki-i0911rk (перевірено: 07.05.2024).

[36] The Market for Lemons (wikipedia) https://en.wikipedia.org/wiki/The_Market_for_Lemons (перевірено: 10.08.2024).

[37] Ринок «лимонів»: невизначеність якості і ринковий механізм (вікіпедія) https://uk.wikipedia.org/wiki/Ринок_«лимонів»:_невизначеність_якості_і_ринковий_механізм (перевірено: 10.08.2024).

[38] Джордж Акерлоф (вікіпедія) https://uk.wikipedia.org/wiki/Джордж_Акерлоф (перевірено: 12.08.2024).

[39] Список лауреатів Премії імені Нобеля з економіки (вікіпедія), розділ «2001—2010» https://uk.wikipedia.org/wiki/Список_лауреатів_Премії_імені_Нобеля_з_економіки#2001—2010 (перевірено: 12.08.2024).