– широка категоризація;
– рекомендації;
– детальна інформація;
– наявність пропозицій.

Недоліки:

– відсутність автоматичного оновлення місць;
– відсутність категорії як відстань.

Ukraine.is.com. Цей сервіс є локальним сервісом для України, він має достатню кількість інформації про місця відвідування, рекомендації та геолокацію, але відсутня категоризація за оцінками, відстанню, середньою ціною меню і інших зручних можливостей.

Переваги:

– велика кількість інформації і відкликів про місця;
– досить часте оновлення.

Недоліки:

– є локальним сервісом, тобто немає інформації про міста не в Україні;
– відсутній мобільний додаток;
– відсутня категоризація;
– використовуються оцінки лише з сайту.

Швидкодія досягається за рахунок викликів до бази даних,які обмежені лише реєстрацією користувача, відсутності клієнт-серверної обробки за рахунок використання Full-stack фреймворку Django, та використання швидких Http запитів до сервісів-ресурсів.

В результаті отриманий програмний дозволяє знайти місця, які можна відвідати та/або провести приємно вечір. Сервіс допоможе вам визначитись з цікавим місцем відпочинку за допомогою сервісу побудови шляху, а також може запропонувати інші місця, які могли б вам сподобатись на основі ваших вподобань і вподобань інших користувачів.

Список літератури:

1. Google Maps API pricing got you down? See these awesome alternatives. web-сайт: geoawesomeness.com URL: https://geoawesomeness.com/google-maps-api-alternatives-best-cheap-affordable (дата звернення 3.03.2019 р.)
2. Google Maps Api documentation. web-сайт: developers.google.com URL: https://developers.google.com/maps/documentation/ (дата звернення 10.03.2019 р)
3. Django documentation. web-сайт: docs.djangoproject.com URL: https://docs.djangoproject.com/en/2.2/ (дата звернення 21.03.2019 р)

Програмна реалізація мінімізації булевих функцій за допомогою методів Квайна та Квайна-Мак-Класкі

Нікітюк В.С., Бєсєдіна С.В., Черкаський національний університет імені Богдана Хмельницького, Черкаси, Україна, Vladnik2k@gmail.com

Software implementation of minimization of boolean functions by Quine and Quine-McCluskey methods

Nikitiuk V., Besedina S., The Bohdan Khmelnytsky National University of Cherkasy, Cherkasy, Ukraine, Vladnik2k@gmail.com

Abstract

The article describes different methods of minimization of boolean functions, how to implement Quine and Quine-McCluskey methods in a program code, explains the difference of them and shows what method is better and easier for coding and understanding.

З розвитком комп'ютерних технологій об'єм інформації все більше зростає, програмні та апаратні можливості комп'ютерних систем теж не відстають від прогресу, тому виникає необхідність вибору мінімальної та оптимальної інформації з усього потоку, вибору даних та створення запитів в базах даних, проектування та синтез комбінаційних схем. Для створення найбільш ефективного варіанту комбінаційних схем необхідно математично-логічним шляхом обрати оптимально мінімальний варіант – саме для цього використовується мінімізація булевих функцій.

Проблеми та недоліки відомих методів мінімізації досконалих кон'юктивних та диз'юнктнивних нормальних форм (ДКНФ і ДДНФ) булевих функцій пов'язані зі зростанням обсягу обчислень при збільшенні кількості змінних логічних функцій. Складність задачі мінімізації булевих функцій n змінних у класі ДКНФ та ДДНФ з ростом n зростає за експоненціальним законом.

Існує велика кількість методів мінімізації булевих функцій [1; 2]. Розглянемо найбільш поширені з них:

Метод Вейча передбачає побудову тупикової форми логічної функції за допомогою спеціальної таблиці, яка називається діаграмою (картою) Вейча і являє собою спеціально перебудовану таблицю істинності функції. Ця діаграма виступає графічним способом мінімізації функції і забезпечує відносну простоту роботи із виразами. Але при збільшенні кількості змінних з 5-6 визначення мінімізованої функції є досить складним й трудомістким процесом.

Карта Карно – це метод спрощення виразів булевої алгебри, зроблене Морісом Карно для поліпшення діаграм Вейча. Карта Карно зменшує потребу в великих обчисленнях, використовуючи перевагу людської можливості розпізнання шаблонів, дозволяє швидке розпізнавання і виключення потенційних станів гонитви. Але проблеми залишаються із збільшенням змінних: карта Карно зазвичай становиться важкою для розпізнання при збільшені кількості змінних. Загальне правило таке: карта Карно добре працює до чотирьох-п'яти змінних, і не має використовуватись із кількістю більше ніж шість змінних.

Метод Квайна полягає в послідовному виконанні всіх можливих склеювань і потім всіх поглинань, що призводить до мінімізованої ДНФ. Він досить легкий в реалізації та непогано працює з багатьма змінними, але через використання імен змінних в функції досить важко в апаратній частині реалізувати розпізнавання функції, що також заважає оптимізованій роботі з даним методом.

Метод Квайна-Мак-Класкі являє собою формалізований на етапі знаходження простих імплікант метод Квайна. У цьому методі використовується геометричне подання логічних функцій. Недоліком методу Квайна є необхідність повного попарного порівняння всіх мінтермів на етапі знаходження первинних імплікант. Із їх збільшенням зростає кількість попарних порівнянь, але числове подання функції алгебри логіки дозволяє спростити етап знаходження первинних імплікант. Саме тому цей метод реалізовувати програмно легше та краще за будь-який інший.

Проаналізувавши різні методи мінімізації булевих функцій, цікавим лишається питання програмної реалізації методів Квайна та Квайна-Мак-Класкі.

Тому надзвичайно важливо спершу визначитись з інструментарієм роботи. Було вибрано мову програмування C++, адже саме вона вважається однією з найкращих мов програмування для розв'язання задач алгоритмічного характеру.

Потім необхідно визначити за допомогою якого типу даних краще та надійніше зберігати інформацію. Для розгляду брались списки та масиви. У списках немає доступу до кожного елементу за короткий проміжок часу, тому було вирішено використовувати

динамічні масиви, а саме вектори. Кількість даних даного типу може збільшуватися і зменшуватися за необхідністю. Векторам зазвичай потрібно більше пам'яті, ніж статичним масивам, але в даній задачі це не є важливим критерієм, адже навіть на великих кількостях змінних розмір затраченої пам'яті не буде суттєвим. Асимптотично головні дії з вектором в C++ можна описати таким чином [3]:

1. Доступ до елементу масиву – O(1).
2. Вставка та видалення елемента в кінці – O(1), але це значення може змінюватись.
3. Вставка та видалення елементів – O(n), де n – кількість елементів масиву.

Можна помітити, що операції видалення та вставки елементів досить негативно впливають на асимптотику всього алгоритму, тому для найбільш оптимізованого рішення треба намагатись уникати цих операцій.

Для програмної реалізації при порівнянні методів Квайна та Квайна-Мак-Класкі в основу буде покладено ДДНФ.

Розглянемо алгоритм роботи програмної реалізації методу Квайна-Мак-Класкі:

1. Спочатку на основі заданої функції записується ДДНФ. При програмній реалізації необхідно спочатку просто пройти циклом по всій функції при зустрічі «1» – набір, який відповідає їй записуємо в масив ДДНФ, а якщо зустрічається «0», то він пропускається. Для більш зрозумілого сприйняття всі обрані набори записуються за допомогою нулів та одиниць.
2. Далі всі номери розбити на групи, що не перетинаються. Ознакою утворення і-ї групи є наявність і одиниць у кожному двійковому номері конституенти одиниці.
3. Потім виконується склеювання тільки між номерами сусідніх груп. Номери, що склеюються, позначаються якою-небудь позначкою (закреслюванням або *). На місце змінної, яка зникає під час склеювання, ставиться знак «–». Для того, щоб це краще реалізувати, в результаті склеювання потрібно використати новий масив, який необхідно заповнювати методом push_back() для vector. У наслідок цього асимптотично алгоритм не втратить свою цінність, адже вставка елемента в кінець становить O(1).
4. На наступному етапі виконати всі можливі склеювання одержаних елементарних добутків. Склеюються тільки ті добутки, що мають знаки «–» в однакових позиціях. Для реалізації цього пункту не потрібно створювати новий масив, досить використати отриманий в п.3 масив, але потрібно зберігати індекси початку та кінця масиву. У такому разі досить легко буде визначати, до якої межі робити склеювання. Для додавання нових склеювань доцільно знову ж використати метод push_back().
5. Далі виконати всі можливі склеювання одержаних елементарних добутків. Склеюються тільки ті добутки, що містять однакові змінні.
6. Для одержання мінімальної ДНФ необхідно забрати зі скороченої ДНФ усі зайві прості імпліканти. Це робиться за допомогою побудови спеціальної імплікантної матриці. Рядки такої матриці відзначаються простими імплікантами булевої функції, тобто членами скороченої ДНФ, а стовпці — конституентами одиниці, тобто членами ДДНФ булевої функції.
7. Проста імпліканта поглинає деяку конституенту одиниці, якщо є її власною частиною. Відповідну клітинку імплікантної матриці на перетині рядка (з розглянутою простою імплікантою) і стовпця (з конституентою одиниці) позначити певною позначкою (хрестиком, плюсиком і т.і.).
8. Далі відшукати стовпці імплікантної матриці, що мають тільки одну позначку. Відповідні цим позначкам прості імпліканти називаються базисними і складають так зване ядро булевої функції. Ядро обов'язково входить у мінімальну ДНФ. Всі використані базисні імпліканти записуємо у масив остаточної відповіді. Їх видалення займає значну частину часу, тому краще створити булевий масив статусів використаних наборів.
9. Розглянути різні варіанти вибору сукупності простих імплікант, що накриють позначками інші стовпці імплікантної матриці, і обрати варіанти з мінімальним сумарним числом літер у такій сукупності імплікант. Вибрану імпліканту потрібно позначити

видаленою (в масиві статусів) та записати у остаточну відповідь. Після цього здійснюється перехід до пункту 8 та цей цикл повторюється доти, доки не залишиться зникнуть всі початкові набори функції.

На даному етапі у нас залишається масив, який містить відповідь. Саме її потрібно вивести користувачу: шукана МДНФ. Приклад результату програми, для знаходження МДНФ для заданої функції методом Квайна-Мак-Класкі зображений на рис. 1.

```
Enter your function: 1010101010101010


DDNF: 0000 || 0010 || 0100 || 0110 || 1000 || 1010 || 1100 || 1110


Let's do gluing:
1--0 | (5 + 7) + (6 + 8)
---0 | ((1 + 2) + (3 + 4)) + ((5 + 6) + (7 + 8))


0000  0010  0100  0110  1000  1010  1100  1110
1--0 + +++
-- - 0 + +++++++
MDNF: ---0
```

Рисунок 1. Приклад роботи програми знаходження МДНФ методом Квайна-Мак-Класкі.

Реалізація програми алгоритму методу Квайна ідентична, але в цьому методі на відміну від Квайна-Мак-Класкі потрібно додатково реалізовувати розпізнавання змінних, що викликає деякі труднощі в написанні програми та створює проблеми із збереженням даних. В цьому і полягає головна відмінність між даними методами. Приклад результату програми для знаходження МДНФ для заданої функції Квайна зображений на рис. 2.

```
Enter your function: 1010101010101010

DDNF: -x1-x2-x3-x4||-x1-x2x3-x4||-x1x2-x3-x4||-x1x2x3-x4||x1-x2-x3-x4||x1-x2x3-x4||x1x2-x3-x4||x1x2x3-x4

Let's do gluing:
x1-x4 | (5 + 7) + (6 + 8)
-x4 | ((1 + 2) + (3 + 4)) + ((5 + 6) + (7 + 8))

            -x1-x2-x3-x4  -x1-x2x3-x4  -x1x2-x3-x4  -x1x2x3-x4  x1-x2-x3-x4  x1-x2x3-x4  x1x2-x3-x4  x1x2x3-x4
x1-x4                                                            +           +           +           +
-x4                   +           +           +           +           +           +           +           +
MDNF: -x4
```

Рисунок 2. Приклад роботи програми знаходження МДНФ методом Квайна.

Розглянуті в роботі методи мінімізації булевих функцій було реалізовно програмно. Запропонована програмна реалізація методу Квайна-Мак-Класкі, яка показала коректність її використання при мінімізації булевих функцій. Простота алгоритму дозволяє провести його технічну реалізацію на засобах обчислювальної техніки. Використання даного методу дозволяє зменшити вимоги до програмно-апаратних засобів автоматизованих систем проектування дискретних пристроїв.

Список літератури:

1. Kondratenko N. R. Computer Workshop on Mathematical Logic: A Textbook. Vinnitsa: VNTU, 2010. 117 p.
2. Applied Theory of Digital Machines / K.G. Samofalov, A.M. Romankevich, V.N. Valuezky, J.S. Kanevsky, M.M. Pinevych. K.: Higher school. The main ed., 1987. 375 p.
3. C++. Контейнеры. std::vector. web-сайт: ru.cppreference.com URL: https://ru.cppreference.com/w/cpp/container/vector. (дата звернення 14.05.2019).