Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Завдання №1

“ Ідеальне хешування для раціональних чисел ”

Варіант №T5

Виконала студентка 2-го курсу

Групи ІПС-22

Клевчук Марія Вячеславівна

Київ - 2024

Завдання

Реалізувати алгоритм ідеального хешування за типом даних - раціональні числа.

Теорія

Ідеальна хеш-функція - це функція, яка перетворює елементи статичної множини даних (ключів) у число з діапазону натуральних чисел без колізій. Це означає, що кожному ключу відповідає унікальне значення хеш-функції. Хеш функція є детермінованою.

Ідеальне хешування використовує два рівні хешування для того, щоб запобігти колізіям. Тобто, у кожній комірці хеш-таблиці створюється вторинна хеш-таблиця. Її розмір визначається як квадрат кількості ключів, захешованих у комірку.

Колізія (в хешуванні) - ситуація, коли декілька елементів початкової множини отримують однакові хеш-значення після застосування хеш-функції.

Ключ (в хешуванні) - це елемент з множини початкових даних, до якого застосовується хеш-функція.

Хеш-таблиця – це структура даних, яка зберігає пари "ключ-значення" і забезпечує швидкий доступ до даних за допомогою хеш-функції. Хеш-функція визначає індекс (або комірку) в таблиці, куди зберігається відповідне значення

Алгоритм

1) Отримуємо набір вхідних даних (вектори з раціональних чисел, подаються у вигляді чисельника та знаменника).
2) Перевіряємо, чи є коректним набір вхідних даних.
3) Перевіряємо дроби на скоротність, та за необхідністю скорочуємо дріб.
4) Перевіряємо вхідні дані на повтори.
5) Створюємо хеш-таблицю, розмір якої дорівнює кількості елементів вихідних даних.
6) Для кожного елемента вхідних даних визначаємо хеш значення за допомогою формули $h(x) = ((a * num(x) + b * den(x)) \bmod p) \bmod n$ (num(x) - сума чисельників чисел з вектора x, den - сума знаменників чисел з вектора x, n - кількість унікальних елементів з набору вхідних даних, p - деяке просте число, a і b - випадково задані числа).
7) У випадку виникнення колізії у відповідній комірці створюємо хеш-таблицю розміром квадрату кількості ключів, що потрапили в одну комірку.

8) Заповнюємо вторинну хеш-таблицю, обравши нові випадкові значення a та b. Значення обираються до тих пір, поки у вторинній хеш-таблиці буде відсутня колізія.
9) Виводимо в консоль хеш-таблицю (за наявності і вторинну хеш-таблицю) з індексами і ключами, які туди входять.

Складність алгоритму

Базові операції (пошук) виконуються за константний час O(1).
Створення хеш-таблиці в найгіршому випадку відбувається за час O(n).

Мова реалізації алгоритму

C++

Модулі програми

Функції: int gcd(int a, int b). Обчислює НСД для чисельника і знаменника, використовується для скорочення дробів.

Структури: Rational. Представляє раціональні числа у вигляді numerator/ denominator. Скорочує дроби за необхідності, переносить знак із знаменника в чисельнику за потреби. Реалізовує порівняння дробів == і <.

Класи: SecondaryHashTable. Реалізовує вторинну хеш-таблицю у випадку виникнення колізій. Методи: vector<vector<Rational>> table – внутрішня структура для зберігання наборів чисел; int hash(const vector<Rational>& values) – хеш-функція; void rehash(const vector<vector<Rational>>& values) – перевизначає хеш-функцію, поки не отримається унікальне розташування всіх значень; bool search(const vector<Rational>& values) – шукає набір у хеш-таблиці; void display() – виводить таблицю;.

PerfectHashTable. Створює основну хеш-таблицю. Методи: vector<vector<vector<Rational>>> primaryTable – основний масив з хешованими наборами; vector<SecondaryHashTable*> secondaryTables – допоміжні хеш-таблиці для обробки колізій; int hash(const vector<Rational>& values) – хеш-функція; void rehash() – перебудовує таблицю, щоб забезпечити відсутність колізій; void insert(const vector<Rational>& vec) – додає новий набір раціональних чисел; bool search(const vector<Rational>& vec) – шукає набір у таблиці; void display() – виводить хеш-таблицю.

Інтерфейс користувача

Через інтерфейс користувача вводиться кількість векторів, кількість чисел у окремому векторі та самі числа у форматі чисельника і знаменник.

У випадку, якщо користувач вводить знаменник 0, програма видає помилку та завершує роботу.

За коректно введених даних програма виводить таблицю у вигляді Index(0….n-1), де показано вектори, які відповідають індексу. У випадку наявності вторинної хеш-таблиці вона виводиться у такому ж вигляді.

Тестові приклади

Приклад 1 (введення некоректних даних).

```
Enter the number of sets of rational numbers: 3
Enter number of elements in set 1: 1
Enter numerator and denominator for element 1: 1 0
Denominator cannot be zero. Exiting program.

Process finished with exit code 0
```

Приклад 2 (показує, що програма не розглядає дроби з однаковим значенням як окремі елементи. А також показує, що програма розглядає дроби з протилежними знаками як окремі елементи).

```
Enter the number of sets of rational numbers: 4
Enter number of elements in set 1: 2
Enter numerator and denominator for element 1: 1 2
Enter numerator and denominator for element 2: 1 2
Enter number of elements in set 2: 2
Enter numerator and denominator for element 1: 1 2
Enter numerator and denominator for element 2: 2 4
Enter number of elements in set 3: 2
Enter numerator and denominator for element 1: -1 2
Enter numerator and denominator for element 2: -1 2
Enter number of elements in set 4: 1
Enter numerator and denominator for element 1: 1 2
Index 0: (1/2 1/2 )
Index 1: (-1/2 -1/2 )
Index 2: (1/2 )
Index 3: _
```

Приклад 3 (задані вручну a=7, b=11, p=101. Для вторинної хеш-таблиці: a=11, b=7, p=101. Також виконується пошук елемента в хеш-таблиці. Наведений приклад де елемент знайдено і не знайдено)

Наша хеш-функція: $h(x) = ((a * num(x) + b * det(x)) \bmod p) \bmod m$.

Обчислимо для 5/6: $((7*5 + 11*6) \bmod 101) \bmod 4 = ((35+66) \bmod 101) \bmod 4 = (101) \bmod 101) \bmod 4 = 0 \bmod 4 = 0$.

Обчислимо для 1/2, -1/7: $((7*(1-1) + 11*(2+7)) \bmod 101) \bmod 4 = ((0+99) \bmod 101) \bmod 4 = (99) \bmod 101) \bmod 4 = 99 \bmod 4 = 3$.

```
Enter the number of sets of rational numbers: 5
Enter number of elements in set 1: 1
Enter numerator and denominator for element 1: 5 6
Enter number of elements in set 2: 1
Enter numerator and denominator for element 1: 5 6
Enter number of elements in set 3: 2
Enter numerator and denominator for element 1: 1 2
Enter numerator and denominator for element 2: -1 7
Enter number of elements in set 4: 2
Enter numerator and denominator for element 1: 1 9
Enter numerator and denominator for element 2: 7 8
Enter number of elements in set 5: 3
Enter numerator and denominator for element 1: -1 2
Enter numerator and denominator for element 2: 2 3
Enter numerator and denominator for element 3: 4 5
Index 0:
Secondary Hash Table:
  0: (-1/2 2/3 4/5 )
  1: (5/6 )
  2: -
  3: -

Index 1: (1/9 7/8 )
Index 2: _
Index 3: (1/2 -1/7 )
Enter number of elements in the set to search: 1
Enter numerator and denominator for element 1: 5 6
The set exists in the hash table.
Enter number of elements in the set to search: 2
Enter numerator and denominator for element 1: -1 2
Enter numerator and denominator for element 2: 2 3
The set was not found in the hash table.
```

Висновки

У ході лабораторної роботи було реалізовано ідеальне хешування, яке дозволяє здійснювати пошук елементів за постійний час O(1) без колізій. Цей тип хешування ефективний при роботі зі статичними множинами даних, адже він забезпечує пошук елемента за константи час O(1) та забезпечує відсутність колізій.

Використані літературні джерела

- Алгоритми та складність, Лекція 1
- https://en.wikipedia.org/wiki/Perfect_hash_function
- https://www.youtube.com/watch?v=LJqmVfTSOIU
- https://courses.cs.vt.edu/~cs3114/Fall10/Notes/T17.PerfectHashFunctions.pdf
- https://www.cs.otago.ac.nz/cosc242/pdf/L11.pdf