Київський національний університет імені Тараса Шевченка
Факультет комп`ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем
Алгоритми та складність

Лабораторна робота №1
“Ідеальне хешування”

Виконав студент 2-го курсу
Групи ІПС-21
Ляшенко Матвій Олексійович

2025

Завдання

Реалізувати ідеальне хешування.

Теорія

Ідеальна хеш-функція (perfect hash function) — це така хеш-функція, яка без колізій відображає унікальні елементи набору на унікальні хеш-значення. Іншими словами, кожному елементу відповідає своє, неповторне хеш-значення.

Предметна область

Тип даних: Раціональні числа
Особливості: Мають очевидний лінійний порядок.

Алгоритм

1. Спочатку всі елементи розподіляються по **кошиках (buckets)** за допомогою першого рівня хеш-функції. Це дозволяє зменшити ймовірність колізій між різними ключами.
2. Кошики сортуються у порядку спадання за кількістю елементів — від найбільш заповнених до найменш заповнених.
3. Для кожного кошика підбираються параметри хеш-функції (так званий модифікатор або salt value) таким чином, щоб при повторному хешуванні всі елементи з кошика займали унікальні, вільні індекси в підтаблиці.
4. Обчислені модифікатори зберігаються в окремому масиві — вони будуть потрібні для здійснення пошуку.
5. Якщо кошик містить лише один елемент, немає потреби обчислювати модифікатор — достатньо знайти будь-яку вільну позицію в підтаблиці та зберегти її індекс (наприклад, як **-index - 1**).

Пошук у таблиці

Після побудови ідеальної хеш-таблиці пошук виконується наступним чином:

1. Ключ хешується першою функцією для визначення кошика, до якого він належить.
2. Далі:
    - Якщо модифікатор для кошика додатній — ключ повторно хешується з цим модифікатором, і результат вказує на індекс у підтаблиці.
    - Якщо модифікатор від’ємний — з нього береться абсолютне значення, і після зменшення на 1 отримаємо індекс у таблиці.
3. Останній крок — **верифікація**: перевіряється, чи співпадає знайдене значення з шуканим ключем. Якщо так — повертаємо дані, якщо ні — ключ відсутній у таблиці.

Складність

Алгоритм працює за константний час O(1) в найгіршому випадку.

Мова програмування

C++

Модулі програми

1. **Rational** - представляє раціональне число у вигляді дробу `a/b`. Методи:
   - `Rational(int numerator, int denominator)` – конструктор із автоматичним скороченням;
   - `double value()` – повертає десяткове значення дробу;
   - `std::string toString()` – текстове представлення у форматі `"a/b"`;
   - оператори `==`, `!=` – для порівняння дробів.
2. **UniversalHash** - реалізує універсальну хеш-функцію з параметрами `a`, `b`, `p`, `m`. Використовується для обчислення хешів як на першому, так і на другому рівні таблиці.
3. **HashRow** - відповідає за окремий рядок (підтаблицю) другого рівня хешування. Зберігає:
   - параметри хеш-функції `a`, `b`;
   - розмір підтаблиці;
   - масив `Rational*` з елементами.
   - метод `print()`, який виводить інформацію про підтаблицю в консоль.
4. **PerfectHashTable** - реалізує основну логіку побудови двоступеневої хеш-таблиці:
   - `build(std::vector<Rational>)` — будує таблицю з множини дробів;
   - `print()` — виводить усі кошики та підтаблиці у зручному вигляді.
5. **main() -** Головна функція ініціалізує множину раціональних чисел, будує хеш-таблицю за допомогою `PerfectHashTable::build()` та викликає `print()` для демонстрації структури.

Інтерфейс користувача

Користувач не вводить дані вручну — програма працює зі заздалегідь визначеною множиною раціональних чисел, що зберігається у векторі.

Тестовий приклад

Візьмемо наступну множину раціональних чисел:

{ 1/2, 2/3, 3/4, 4/5, 5/6, 6/7, 7/8, 8/9, 9/10, 10/11 }

Для побудови ідеального хешування оберемо наступні параметри:

- m = 10 - кількість ключів (розмір першої таблиці)

- p = 10000019 - велике просте число
- a = 31, b = 17 - параметри першої хеш-функції

. Розрахуємо індекс комірки за формулою $h(x)=((a \cdot int(x \cdot 10^6)+b) mod p) mod m$.

Оскільки дроби мають десяткове представлення, помножимо їх на 10^6 для отримання цілого представлення ключа:

Дріб	Значення	int(значення · 1e6)	h(x) = ((a · x + b) mod p) mod m
1/2	0.5	500000	((31 · 500000 + 17) mod p) mod 10 = 2
2/3	0.666...	666666	= 6
3/4	0.75	750000	= 7
4/5	0.8	800000	= 9
5/6	0.833...	833333	= 1
6/7	0.857...	857142	= 0
7/8	0.875	875000	= 5
8/9	0.888...	888888	= 8
9/10	0.9	900000	= 4
10/11	0.909...	909090	= 3

Усі кошики отримали **по 1 ключу**, тож ідеальне хешування завершено **на першому рівні**. Для кожного з них можна просто зберегти індекс у підтаблиці без потреби в додатковому підборі **a і b**.

У випадку знаходження дублікату доведеться створити таблицю розміром **s^2=4** і підібрати **a та b**, які розмістять обидва значення **без колізій.**

Висновки

У результаті роботи було реалізовано ідеальне хешування для множини раціональних чисел. Такий підхід є ефективним, оскільки дозволяє повністю уникнути колізій при зберіганні даних. Основною перевагою ідеального хешування є гарантований доступ до елементів за константний час. Водночас основним обмеженням залишається вимога попереднього знання всієї множини ключів, тобто її статичність.

Література:

- https://neerc.ifmo.ru/wiki/index.php?title=%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0#.D0.A5.D0.B5.D1.88.D0.B8.D1.80.D0.BE.D0.B2.D0.B0.D0.BD.D0.B8.D0.B5
- https://neerc.ifmo.ru/wiki/index.php?title=%D0%98%D0%B4%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%85%D0%B5%D1%88%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%

D0%B8%D0%B5

- https://habr.com/ru/post/254431/
- https://en.wikipedia.org/wiki/Perfect_hash_function