Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Лабораторна робота №1

“Реалізація ідеального хешування”

Виконав студент 2-го курсу

Групи ІПС-22

Чорнокур Іван Вадимович

2024

Завдання:

Ідеальне хешування – тип даних дійсні.

Теорія:

Ідеальне хешування - це хеш-функція, яка без колізій відображає різні елементи з множини об’єктів на множину ключів за часом в найгіршому випадку. Це використовується в задачах із статичним множинами ключів (тобто після того, як всі ключі збережено в таблиці, їх множина ніколи не змінюється) для забезпечення хорошої асимптотики навіть в найгіршому випадку.

Основна ідея ідеального хешування полягає в використанні дворівневої схеми хешування з універсальним хешуванням на кожному рівні. На першому рівні використовується той же принцип, що і в разі хешування з ланцюжками: ключі хешуються в ячейки з використанням хеш-функції, випадково обраної з сімейства універсальних хеш-функцій. На другому рівні замість створення списку ключів використовується другорядна хеш-таблиця, яка зберігає всі ключі, хешовані функцією в ячейку, зі своєю функцією, обраною з множини. Ця технологія застосовується в різних словниках та базах даних, в алгоритмах зі статичною (відомою заздалегідь) інформацією.

Алгоритм:

1. Ініціалізація

• Створити клас PerfectHashing, який містить приватний член data типу std::unordered\_map<size\_t, double>.

2. Хеш-функція

• Визначити приватний метод hashFunction(double value), який використовує стандартну хеш-функцію для типу double (std::hash<double>()).

3. Вставка

• Визначити метод insert(double value), який:

• Викликає хеш-функцію для обчислення хеш-значення h для value.

• Перевіряє, чи існує ключ h у data.

• Якщо ключ h не існує, додає пару h і value до data.

• Якщо ключ h вже існує, викликає виняток з повідомленням про колізію.

4. Пошук

• Визначити метод search(double value), який:

• Викликає хеш-функцію для обчислення хеш-значення h для value.

• Перевіряє, чи існує ключ h у data та чи відповідає значення data[h] введеному value.

• Повертає true, якщо значення знайдено, і false в іншому випадку.

5. Видалення

• Визначити метод remove(double value), який:

• Викликає хеш-функцію для обчислення хеш-значення h для value.

• Перевіряє, чи існує ключ h у data та чи відповідає значення data[h] введеному value.

• Якщо значення знайдено, видаляє його з data.

• Якщо значення не знайдено, викликає виняток з повідомленням про відсутність значення.

6. Головна функція

• У головній функції створити об’єкт класу PerfectHashing.

• Виконати операції вставки, пошуку та видалення з обробкою винятків.

Псевдокод:

class PerfectHashing:

data = empty map of (size\_t, double)

method hashFunction(value):

return standard hash of value

method insert(value):

h = hashFunction(value)

if h not in data:

data[h] = value

else:

raise exception "Collision detected! Hash value already exists."

method search(value):

h = hashFunction(value)

if h in data and data[h] == value:

return true

else:

return false

method remove(value):

h = hashFunction(value)

if h in data and data[h] == value:

remove data[h]

else:

raise exception "Value not found!"

main function:

create instance ph of PerfectHashing

try:

ph.insert(3.14)

ph.insert(2.71)

ph.insert(1.618)

print "Searching for 3.14: " + (ph.search(3.14) ? "Found" : "Not Found")

print "Searching for 2.71: " + (ph.search(2.71) ? "Found" : "Not Found")

print "Searching for 1.618: " + (ph.search(1.618) ? "Found" : "Not Found")

print "Searching for 0.577: " + (ph.search(0.577) ? "Found" : "Not Found")

ph.remove(2.71)

print "Searching for 2.71 after removal: " + (ph.search(2.71) ? "Found" : "Not Found")

catch exception as e:

print e.message

Модулі програми:

Модуль 1: Клас PerfectHashing

Цей модуль містить основні методи для хешування, вставки, пошуку та видалення елементів.

Модуль 2: Головна функція

Цей модуль містить приклад використання класу PerfectHashing з операціями вставки, пошуку та видалення елементів.

Модуль 3: Заголовки бібліотек

Цей модуль включає необхідні бібліотеки для роботи програми.

Модуль 4: Опис класу PerfectHashing

Цей модуль описує структуру класу PerfectHashing, включаючи його приватні та публічні методи.

Модуль 5: Опис головної функції

Цей модуль описує основні операції, які виконуються у головній функції для демонстрації роботи класу PerfectHashing.

Висновок:

Ідеальне хешування - це ефективний метод організації даних, який найкраще підходить для ситуацій, коли множина ключів відома заздалегідь. Воно дозволяє досягти оптимального часу пошуку в найгіршому випадку, що робить його відмінним вибором для статичних наборів даних. Однією з основних переваг ідеального хешування є те, що воно гарантує відсутність колізій, тобто кожен ключ матиме унікальний індекс в хеш-таблиці. Однак, ідеальне хешування вимагає знання всіх ключів заздалегідь, що робить його менш підходящим для динамічних наборів даних. Цей метод часто використовується в базах даних, словниках та інших структурах даних, де потрібна швидка пошукова операція. Загалом, ідеальне хешування - це потужний інструмент для ефективного управління даними, але його використання найкраще підходить для специфічних сценаріїв.