Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Алгоритми та складність

Завдання №1

“Ідеальне хешування”

Виконав студент 2-го курсу

Групи К-29

Пупов Нікіта Андрійович

2020

**Завдання**:

Ідеальне хешування.

**Теорія:**

Ідеальне хешування використовується в задачах зі статичною множиною ключів(після того, як всі ключі збережені в таблиці, їх множина ніколи не змінюється) для забезпечення ефективної асимптотики навіть в гірших випадках. В такому хешуванні для доступу до даних знадобиться тільки розрахунок двох хеш-функцій. Такий алгоритм використовується в різних словниках, базах даних, а також в алгоритмах зі статичною(відомою заздалегідь) інформацією.

Ми будемо використовувати дворівневе хешування з універсальним хешуванням на кожному рівні.

**Алгоритм**

Спочатку побудуємо перший рівень хеш-таблиці розміру N. Для цього знайдемо таку хеш-фунцію h(x), що . Де N-кількість елементів. n(i)-кількість елементів, що знаходяться в i-му бакеті(тобто кількість елементів, у яких однаковий хеш за h(x)). Коли ми знайшли таку функцію, розміщуємо елементи в хеш-таблиці з такою хеш-функцією.

Далі для кожного i-го бакета будуємо окрему хеш-таблицю розміру . І для кожної знаходимо таку хеш-функцію, щоб все її елементи були розміщені окремо.

Якщо ми захочемо знайти елемент в цій хеш-таблиці, то ми спочатку знайдемо в якому бакеті першого рівня він знаходиться(маючи h(x)), а потім зробимо аналогічно для бакета другого рівня.

**Мова програмування** С++

**Структура програми**

Програма складається з головного класу алгоритму IdealHashTable, який працює з об’єктами типу Folder. Хешування відбувається по полю name класу Folder. Для запуску програми на тестових даних використовується Source.cpp.

**Модулі програми:**

* **IdealHashTable(const vector<Folder> &folders);**

Конструктор таблиці. Приймає вектор папок. На основі цієї інформації створює дворівневу структуру хеш-таблиць.

Складність O(n)

* **Folder get(int name)**

Знаходить папку по її імені.

Складність O(1)

**Тестові приклади**

Для простоти будемо працювати з числами.

Наприклад, у нас є набір чисел {10, 21, 6, 16, 13}.

Ми підібрали таку h(x) = (3x)\*5. І на її основі побудували таку хеш-таблицю

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** |
| {10} |  |  | {21, 6, 16} | 13 |

Тепер для кожного бакету будуємо свою хеш-таблицю розміру .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** |
| 10 |  |  |  | 13 |
|  |  |  |  |
|  |  |  |  |
|  |  | 21 |  |
|  |  |  |  |
|  |  |  |  |
|  |  | 6 |  |
|  |  | 16 |  |
|  |  |  |  |

Для 3-го бакету маємо хеш-функцію f(x)=x%9.

Таким чином, мы знаємо всі хеш-функії, і можемо швидко знайти потрібний нам елемент.

**Матеріали**

<https://www.youtube.com/watch?v=x27h6WmC6YU>