Київський національний університет імені Тараса Шевченка

Алгоритми та складність

Лабораторна робота 1  
Виконав студент 2-го курсу Групи ПІ-22

Дідківський Володимир Вячеславович

# Завдання

Реалізувати ідеальне хешування.

# Теорія

Ідеальна хеш-функція – хеш функція, яка перетворює завчасно відому статичну множину ключів в діапазоні цілих чисел [0,n-1] без коллізій, тобто один ключ відповідає лише одному унікальному значенню.

# Алгоритм

1. Хешуємо елементи по групах(buckets). Таким чином ми уникнимо колізій.
2. Сортуємо групи від тих, з найбільшою кількістю елементів до тих, з найменшою кількістю.
3. Знаходимо модифікатор(salt value) для кожної групи, що коли кожен елемент з групи хешований, вони займають лише вільні місця.
4. Збережемо масив модифікаторів на кращі часи, він нам знадобиться під час пошуку інформації.
5. Якщо група має тільки 1 предмет, то замість того, щоб шукати значенння модифікатора, ми можемо просто знайти вільний індекс і зберегти (індекс+1) в масив модифікаторів.

Коли ми обрахували ідеальне хешування, пошук будемо здійснювати наступним чином:

1. Хешуємо ключ і використовуємо цей хеш, щоб знайти який модифікатор використовувати.
2. Якщо значення модифікатора >= 0, то хешуємо ключ знову, використовуючи значення модифікатора.Результат цього хешування буде індексом в таблиці інформації(data table).
3. Якщо ж негативне, то беремо абсолютне значення і віднімаємо 1 – отримаємо індекс в таблиці інформації.
4. Можливо і таке, що ключ, який ми шукаємо, може бути не в таблиці, порівняємо цей ключ з ключем за цим індексом в таблиці. Якщо вони співпадають, повернемо інформацію по цьому індексу. В інакшому випадку ключа не було в таблиці.

# Складність

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

# Мова програмування

C++

# Модулі програми

student.h

class Student{};

std::string getName();

void getStudent();

void setName(std::string name);

group.h

class Group {};

Group() : title("NULL");

Group(std::string title);

Group(std::string title, Student\* first\_student);

std::string getGroupTitle();   
 std::vector<Student\*> getGroupStudents();   
 void setGroupTitle(std::string title);

void setGroupStudents(std::vector<Student\*> students);

void addStudent(Student\* student);

void printStudents();

perfectHash.h

struct UniversalHash;

class HashRow;

class Core;

void Core::run();  
void Core::printGroups();

std::vector<std::string> Core::getGroupsTitles(int count);

void Core::fillHashTable(HashRow\* hashTable);   
int Core::createHash(int id, int a, int b, int m);

void Core::printHashes(std::vector<std::vector<Group\*>> hashesTable);

std::ostream& operator<<(std::ostream& os, HashRow hashRow);

# Інтерфейс користувача

Користувач вводить дані в консолі. Дані виводяться в текстовий файл

# Тестовий приклад

Візьмемо таку множину груп. {K18,K19, K28, K29,IPS31}. Переведемо його до виду ASCII, отримаємо {754956,754957,755056,755057,7380835149}.

Звідки m = 6, p = 754923. Візьмемо а = 41, b = 460. Розрахуємо індекс комірки за формулою k = h(i) = ((a \* i + b) mod p) mod m.

ℎ(754956) = ((41 ∗ 754956 + 460)𝑚𝑜𝑑 754923)𝑚𝑜𝑑 6 = 1

ℎ(754957) = ((41 ∗ 754957 + 460)𝑚𝑜𝑑 754923)𝑚𝑜𝑑 6 = 0

ℎ(755056) = ((41 ∗ 755056 + 460)𝑚𝑜𝑑 754923)𝑚𝑜𝑑 6 = 3

ℎ(755057) = ((41 ∗ 755057 + 460)𝑚𝑜𝑑 754923)𝑚𝑜𝑑 6 = 2

ℎ(7380835149) = ((41 ∗ 7380835149 + 460)𝑚𝑜𝑑 754923)𝑚𝑜𝑑 6 = 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| k | p | m | a | b | i | |
| 0 | 754923 | 6 | 41 | 460 | 754957 | |
| 1 | 754956 | 7380835149 |
| 2 | 755057 | |
| 3 | 755056 | |

Для числа 1 отримали 2 початкові ключі (754956,7380835149), тому створюємо хеш-рядок розміром 𝑚2 = 9, підбираємо нові а і b(a = 43321, b=2312). Знові рахуємо хеш-числа

ℎ(754956) = ((1423 ∗ 754956 + 2312)𝑚𝑜𝑑 754923)𝑚𝑜𝑑 9 = 5

ℎ(7380835149) = ((1423 ∗ 7380835149 + 2312)𝑚𝑜𝑑 754923)𝑚𝑜𝑑 9 = 8

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| k | p | m | a | b | i | | | |
| 0 | 0 | 1 | 0 | 0 | 754957 | | NULL | |
| 1 | 754923 | 9 | 1423 | 2312 | 754956 | NULL | 7380835149 | NULL |
| 2 | 0 | 1 | 0 | 0 | 755057 | | NULL | |
| 3 | 0 | 1 | 0 | 0 | 755057 | | NULL | |
| 4 | 0 | 0 | 0 | 0 | 755056 | | NULL | |
| 5 | 0 | 0 | 0 | 0 | NULL | | | |
| 6 | 0 | 0 | 0 | 0 | NULL | | | |
| 7 | 0 | 0 | 0 | 0 | NULL | | | |
| 8 | 0 | 0 | 0 | 0 | NULL | | | |

# Висновки

Реалізували ідеальне хешування для учбових груп. Ідеальне хешування є досить ефективним, до плюсів можна віднести те що при використанні ідеального хешування ми уникаєм колізій, але до мінусів можна віднести те, що множина ключів статична.

# Література:

* <https://en.wikipedia.org/wiki/Perfect_hash_function>