Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Завдання №1

“Ідеальне хешування”

Виконав студент 2-го курсу

Групи К-29

Демиденко Андрій Геннадійович

2021

**Предметна область**

Предметна область: Відділ кадрів

Об’єкти: Підрозділ (Групи), Співробітники (Студенти)

Примітка: Існує множина підрозділів підприємства (навчальних груп). Кожен підрозділ включає в себе множину співробітників.

**Завдання**

Реалізувати ідеальне хешування. Ми будемо хешувати групи і отримаємо вихідну хеш-таблицю.

**Теорія**

Хеш-функція, або геш-функція — [функція](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0), що перетворює вхідні дані будь-якого (як правило великого) розміру в дані фіксованого розміру. Хешування (гешування, [англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0" \o "Англійська мова) hashing) — перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. Такі перетворення також називаються хеш-функціями, або функціями згортання, а їхні результати називають хешем, хеш-кодом, хеш-сумою.

Хеш таблиці:

* Кількість реально збережених ключів може бути мала відносно простору можливих ключів U або кількість елементів в U завелика.
* При хешуванні елемент з ключем k зберігатиметься в комірці h(k): елемент з ключем k хешується в комірку h(k). Величина h(k) – хеш-значення ключа k.
* Хеш-функція h відображає простір ключів U на комірки хеш-таблиці T[0..m–1]:
* Мета хеш-функції – зменшити робочий діапазон масиву з |U| до m значень.
* Хеш-функція детермінована: для однакових k має давати те саме хеш-значення h(k).

**Алгоритм**

1. Хешування імен різних груп.
2. Перший рівень хешування: n ключів хешуються в m комірок за допомогою універсальної хеш-функції h (h(k)=((a \* k + b) mod p) mod m).
3. Другий рівень хешування: для кожної комірки своя вторинна хеш-таблиця зі своєю універсальною хеш-функцією, вибраною так, щоб уникнути колізій; її розмір – квадрат кількості ключів, захешованих в комірку.

**Складність**

* Ο(1) звертань до пам’яті в найгіршому випадку.
* Очікувана загальна пам’ять під таку структуру O(n).

**Мова програмування**

С++

**Модулі програми**

//Class for description worker

class Worker

{

private:

string name\_;

public:

//Constructors

Worker();

Worker(const string& name);

//Method that return name of worker

string GetName() const;

};

//Class for description different departments

class Department

{

private:

string name\_;

vector<Worker> workers\_;

public:

//Constructors

Department() : name\_("NULL") {};

Department(const string& name);

Department(const vector<Worker>& workers);

Department(const string& name, const vector<Worker>& workers);

//Method for setting list of workers

void SetWorkers(const vector<Worker>& workers);

//Method for adding new workers

void AddWorkers(const Worker& worker);

//Method for adding list of workers

void AddWorkers(const vector<Worker>& workers);

//Method for getting name of department

string GetName() const;

//Method for getting list of workers

vector<Worker> GetWorkers() const;

};

//Structure for description parameters of hashing

struct Params

{

int m;

int a;

int b;

int p;

};

//overloads of operator for output different classes

ostream& operator<<(ostream& os, const Params& p);

ostream& operator<<(ostream& os, const Worker& worker);

template<class K, class V>

ostream& operator<<(ostream& os, const pair<K, V> p);

template<class T>

ostream& operator<<(ostream& os, const vector<T>& v);

ostream& operator<< (ostream& os, const Department& m);

//Function for finding next simple number

int NextSimple(int number);

//Function for hashing strings

int Hash(const string& t);

//Function for hashing department

int HashDepartment(const Department& m);

//Function that returns hash-table

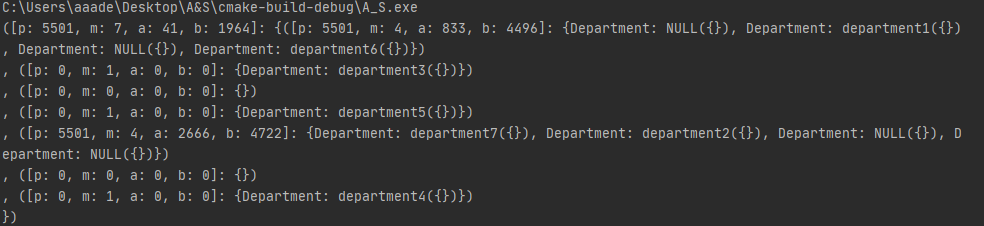
pair<Params, vector<pair<Params, vector<Department>>>> IdealHash(const vector<Department>& v);

**Інтерфейс користувача**

Вхідні дані вводяться генеруються програмою і виводяться в консоль.

**Приклад виводу програми**

Ось так захешувався набір підрозділів:

****

**Тестові приклади**

Нехай ми хешуємо наступні числа k = {10,15,23,46,57,83}

Звідки m = 6, p = 89, підбираємо a і b. Візьмемо a = 41, b = 44. Розрахуємо індекс комірки за формулою: k = h(i) = ((a \* i + b) mod p) mod m.

Маємо наступну таблицю з параметрами:

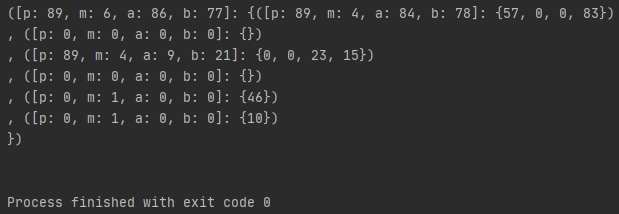
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| k | p | m | a | b | i | |
| 0 | 0 | 1 | 0 | 0 | 15 | |
| 1 | 2 | ? | ? | 46 | 57 |
| 2 | 1 | 0 | 0 | 23 | |
| 3 | 1 | 0 | 0 | 10 | |
| 4 | 0 | 0 | 0 | NULL | |
| 5 | 1 | 0 | 0 | 83 | |

Для числа 1 отримали два початкові ключі(46, 57), тому створюємо хеш-рядок розміром , підбираємо нові a i b(a = 15, b = 67). Знову рахуємо хеш-числа:

Маємо нову хеш-таблицю:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| k | p | m | a | b | i | | | |
| 0 | 0 | 1 | 0 | 0 | 15 | | | |
| 1 | 89 | 2 | 15 | 67 | 46 | 57 | NULL | NULL |
| 2 | 0 | 1 | 0 | 0 | 23 | | | |
| 3 | 0 | 1 | 0 | 0 | 10 | | | |
| 4 | 0 | 0 | 0 | 0 | NULL | | | |
| 5 | 0 | 1 | 0 | 0 | 83 | | | |

Вивід програми:



**Висновки**

Реалізували ідеальне хешування для класу підрозділів, а також для чисел. Ідеальне хешування є досить ефективним, але до мінусів можна віднести те, що множина ключів статична.

**Література**

* Лекція 1
* [Хеш-функція — Вікіпедія (wikipedia.org)](https://uk.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F)