Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

Дисциплина «Алгоритмы и структуры данных»

Отчет к лабораторной работе № 5

«Оценка качества хеш-функций»

Выполнил:

Василения Иван Валерьевич

Мельниченко Артём Олегович

Принял:

Рыженков Д.В.

Орёл, 2024г

Листинг файла main.cpp:

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <cmath> // floor

#include <fstream>   // Для работы с файлами

using namespace std;

string getKey() {

    char symbols[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";

    const int size = 62;

    string key = "";

    for (int i = 0; i < 6; i++)

    {

        key += symbols[rand() % size];

    }

    return key;

}

void clearHashMap(int map[], int size) {

    for (int i = 0; i < size; i++)

    {

        map[i] = 0;

    }

}

int addMethod(string key) {

    int sum = 0;

    for (char c: key)

    {

        sum += (int)c;

    }

    return sum;

}

void getR(int r[], int size) {

    for (int i = 0; i < size; i++)

    {

        r[i] = rand() % 256;

    }

}

int xorMethod(string key, int r[]) {

    int sum = 0;

    for (int i = 0; i < key.length(); i++)

    {

        sum += (int)key[i] ^ r[i];

    }

    return sum;

}

int divideHashMethod(int M, int k) {

    return k % M;

}

int multiplyMethod(int M, int k) {

    float kA = 0.6180339887 \* k;

    float kka = kA - floor(kA);

    return M \* (kA - floor(kA));

}

int main() {

    setlocale(LC\_ALL, "rus");

    srand(static\_cast<unsigned int>(std::time(0))); // Инициализация генератора случайных чисел

    const int mapSize = 1000;

    const int keysAmount = 100;

    string keys[keysAmount];

    int hashMap[mapSize] = {}; //все нули

    ofstream outputFile("data.csv");

    if (!outputFile) {

        std::cerr << "Не удалось открыть файл!" << std::endl;

        return 1;

    }

    for (int i = 0; i < keysAmount; i++)

    {

        keys[i] = getKey();

    }

    // add + divide

    cout << "Аддитивный метод + метод деления" << endl;

    int collisCount = 0;

    for (int i = 0; i < keysAmount; i++)

    {

        string key = keys[i];

        int k = addMethod(key);

        int address = divideHashMethod(mapSize, k);

        if (hashMap[address] > 0) collisCount++;

        hashMap[address]++;

    }

    cout << "Количество коллизий " << collisCount << " из общего количества ключей: "<< keysAmount << endl;

    cout << "Процент коллизий: " << (float)collisCount / (float)keysAmount \* 100 << "%" << endl << endl;

    for (int i = 0; i < mapSize; ++i) {

        outputFile << hashMap[i];

        if (i < mapSize - 1) { // Если это не последний элемент, добавляем запятую

            outputFile << ";";

        }

    }

    outputFile << endl;

    clearHashMap(hashMap, mapSize);

    // xor + divide

    cout << "XOR метод + метод деления" << endl;

    collisCount = 0;

    int r[6];

    getR(r, 6);

    for (int i = 0; i < keysAmount; i++)

    {

        string key = keys[i];

        int k = xorMethod(key, r);

        int address = divideHashMethod(mapSize, k);

        if (hashMap[address] > 0) collisCount++;

        hashMap[address]++;

    }

    cout << "Количество коллизий " << collisCount << " из общего количества ключей: " << keysAmount << endl;

    cout << "Процент коллизий: " << (float)collisCount / (float)keysAmount \* 100 << "%" << endl << endl;

    for (int i = 0; i < mapSize; ++i) {

        outputFile << hashMap[i];

        if (i < mapSize - 1) { // Если это не последний элемент, добавляем запятую

            outputFile << ";";

        }

    }

    outputFile << endl;

    clearHashMap(hashMap, mapSize);

    // add + multiply

    cout << "Аддитивный метод + метод умножения" << endl;

    collisCount = 0;

    for (int i = 0; i < keysAmount; i++)

    {

        string key = keys[i];

        int k = addMethod(key);

        int address = multiplyMethod(mapSize, k);

        if (hashMap[address] > 0) collisCount++;

        hashMap[address]++;

    }

    cout << "Количество коллизий " << collisCount << " из общего количества ключей: " << keysAmount << endl;

    cout << "Процент коллизий: " << (float)collisCount / (float)keysAmount \* 100 << "%" << endl << endl;

    for (int i = 0; i < mapSize; ++i) {

        outputFile << hashMap[i];

        if (i < mapSize - 1) { // Если это не последний элемент, добавляем запятую

            outputFile << ";";

        }

    }

    outputFile << endl;

    clearHashMap(hashMap, mapSize);

    // xor + multiply

    cout << "XOR метод + метод умножения" << endl;

    getR(r, 6);

    collisCount = 0;

    for (int i = 0; i < keysAmount; i++)

    {

        string key = keys[i];

        int k = xorMethod(key, r);

        int address = multiplyMethod(mapSize, k);

        if (hashMap[address] > 0) collisCount++;

        hashMap[address]++;

    }

    cout << "Количество коллизий " << collisCount << " из общего количества ключей: " << keysAmount << endl;

    cout << "Процент коллизий: " << (float)collisCount / (float)keysAmount \* 100 << "%" << endl << endl;

    for (int i = 0; i < mapSize; ++i) {

        outputFile << hashMap[i];

        if (i < mapSize - 1) { // Если это не последний элемент, добавляем запятую

            outputFile << ";";

        }

    }

    outputFile << endl;

    clearHashMap(hashMap, mapSize);

    outputFile.close();

    return 0;

}