МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5**

**по дисциплине  
 «АЛГОРИТМЫ И АНАЛИЗ СЛОЖНОСТИ»**

Выполнил студент группы 24/1                                       А.А.Иванов

Направление подготовки  02.03.03  Математическое обеспечение и администрирование информационных систем

Курс    2

Отчет принял доктор физико-математических наук, профессор                                                                                       А.И. Миков

Краснодар

2022 г.

**Задача:** найти зависимости среднего числа коллизий С(ɑ) при хэшировании потока объектов от величины ɑ загруженности хэш-таблицы при изменении а от 0,05 до 0,95 с шагом 0,05. Анализ провести для линейного (L) и случайного рехэширования (R).

**Решение:** напишем функции **Hash** и **RandomString**, чтобы получать случайные строки и хэшировать их. Сделаем структуру для хэш-таблицы. Будем помещать туда элементы и считать число коллизий, которые могут возникнуть при добавлении нового элемента к уже заполненной таблице.

**Код программы:**

//Линейное хэширование

#include<iostream>

#include<string>

#include<random>

#include<ctime>

#include<cstdlib>

using namespace std;

struct HashTable //Структура хэш-таблицы

{

int number[2000];

string stroka[2000];

int hashcode[2000];

};

int Hash(string &s) //Хэш-функция

{

int N = 2000, Hash = 0, k = 0;

while (s[k] != '\0')

{

Hash += s[k];

k++;

}

Hash %= N;

return Hash;

}

string RandomString(string &s) //Функция-генератор случайных строк

{

char Symb[93] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G',

'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '~', '`', '!', '@',

'"', '#', '№', '$', ';', '%', '^', ':', '&', '?', '\*', '(', ')', '-', '\_', '+', '=', '|', '/', '{', '[', ']', '}', ',', '<', '.', '>'};

int k = rand() % 16 + 5;

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

{

int symb = rand() % 93;

s += Symb[symb];

}

return s;

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(NULL));

HashTable Table;

for (double a = 0.05; a <= 0.95; a += 0.05) //2000 = 1.00 --> 0.05 = (2000 / 100) \* 5 = 100

{

int kolkolliz = 0;

for (int k = 0; k < 1000; k++)

{

int i = a \* 2000;

for (int j = 1; j <= i; j++) //Заполняем таблицу на величину "а"

{

string s;

Table.number[j] = j;

Table.stroka[j] = RandomString(s);

Table.hashcode[j] = Hash(s);

}

string obj;

string obj1 = RandomString(obj);

int obj2 = Hash(obj1);

for (int j = 1; j <= i; j++) //Проверяем новый obj на коллизии

if (obj2 == Table.hashcode[j])

kolkolliz++;

if (kolkolliz > 0)

obj2 += 10000;

}

cout << "Среднее колличество коллизий для заполненности таблицы " << a << " : " << kolkolliz / 1000 << endl << endl;

}

return 0;

}

//Случайное хэширование

#include<iostream>

#include<string>

#include<random>

#include<ctime>

#include<cstdlib>

using namespace std;

struct HashTable //Структура хэш-таблицы

{

int number[2000];

string stroka[2000];

int hashcode[2000];

};

int Hash(string& s) //Хэш-функция

{

int N = 2000, Hash = 0, k = 0;

while (s[k] != '\0')

{

Hash += s[k];

k++;

}

Hash %= N;

int r = rand() % 2000 + 1;

Hash += r % N;

return Hash;

}

string RandomString(string& s) //Функция-генератор случайных строк

{

char Symb[93] = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G',

'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '~', '`', '!', '@',

'"', '#', '№', '$', ';', '%', '^', ':', '&', '?', '\*', '(', ')', '-', '\_', '+', '=', '|', '/', '{', '[', ']', '}', ',', '<', '.', '>' };

int k = rand() % 16 + 5;

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

{

int symb = rand() % 93;

s += Symb[symb];

}

return s;

}

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(NULL));

HashTable Table;

for (double a = 0.05; a <= 0.95; a += 0.05) //2000 = 1.00 --> 0.05 = (2000 / 100) \* 5 = 100

{

int kolkolliz = 0;

for (int k = 0; k < 1000; k++)

{

int i = a \* 2000;

for (int j = 1; j <= i; j++) //Заполняем таблицу на величину "а"

{

string s;

Table.number[j] = j;

Table.stroka[j] = RandomString(s);

Table.hashcode[j] = Hash(s);

}

string obj;

string obj1 = RandomString(obj);

int obj2 = Hash(obj1);

for (int j = 1; j <= i; j++) //Проверяем новый obj на коллизии

if (obj2 == Table.hashcode[j])

kolkolliz++;

if (kolkolliz > 0)

obj2 += 10000;

}

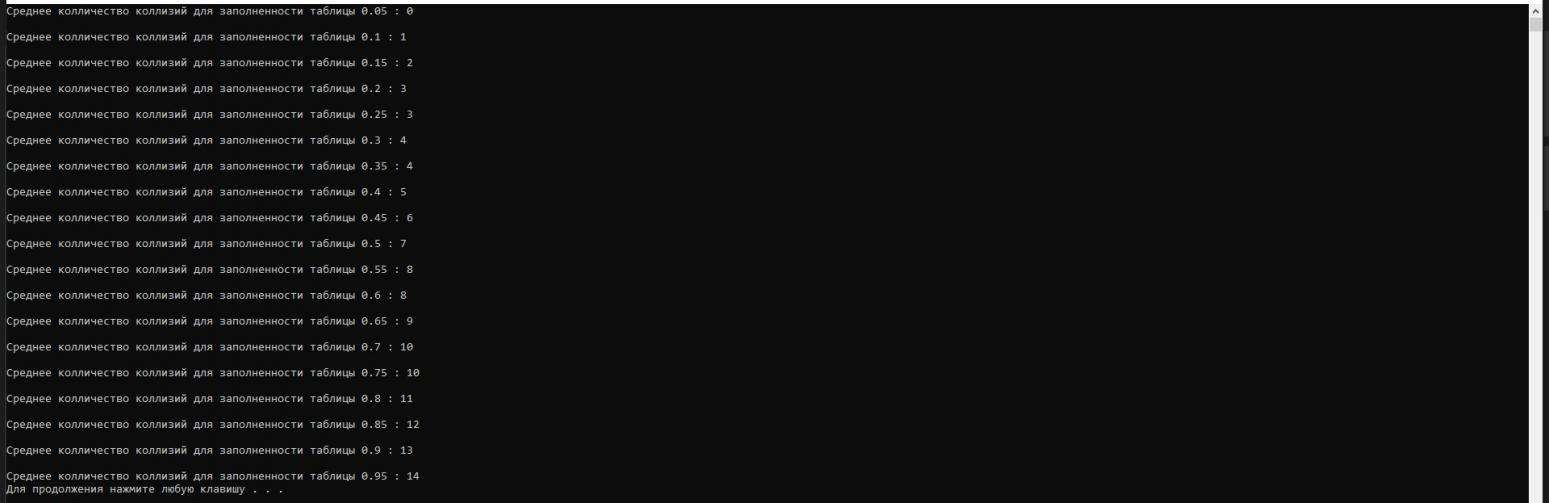
cout << "Среднее колличество коллизий для заполненности таблицы " << a << " : " << kolkolliz / 1000 << endl << endl;

}

return 0;

}

**Пример вывода в консоль:**

****

****

**Таблицы значений:**

|  |  |
| --- | --- |
| Линейное хэширование | |
| Загруженность таблицы | Количество коллизий (среднее значение для 1000 экспериментов) |
| 0,05 | 0 |
| 0,1 | 1 |
| 0,15 | 2 |
| 0,2 | 2 |
| 0,25 | 3 |
| 0,3 | 4 |
| 0,35 | 4 |
| 0,4 | 5 |
| 0,5 | 7 |
| 0,55 | 8 |
| 0,6 | 9 |
| 0,65 | 9 |
| 0,7 | 10 |
| 0,75 | 11 |
| 0,8 | 11 |
| 0,85 | 12 |
| 0,9 | 13 |
| 0,95 | 14 |

|  |  |
| --- | --- |
| Случайное хэширование | |
| Загруженность таблицы | Количество коллизий (среднее значение для 1000 экспериментов) |
| 0,05 | 0 |
| 0,1 | 0 |
| 0,15 | 1 |
| 0,2 | 1 |
| 0,25 | 2 |
| 0,3 | 2 |
| 0,35 | 2 |
| 0,4 | 3 |
| 0,45 | 3 |
| 0,5 | 3 |
| 0,55 | 4 |
| 0,6 | 4 |
| 0,65 | 5 |
| 0,7 | 5 |
| 0,75 | 6 |
| 0,8 | 6 |
| 0,85 | 7 |
| 0,9 | 6 |
| 0,95 | 7 |

**Графики:**

**Исходная формула:**

Y = k2 \* (1 - ɑ)-1 + k1 \* log(1 - ɑ) + k0, где ɑ < 1

Сделаем систему из 3-ёх уравнений, и реши методом **Гаусса** для СL(ɑ) и СR(ɑ):

K2 \* (1 - ɑ0)-1 + k1 \* log(1 - ɑ0) + k0 = y0 = C(ɑ0)

K2 \* (1 - ɑ1)-1 + k1 \* log(1 - ɑ1) + k0 = y1 = C(ɑ1)

K2 \* (1 - ɑ2)-1 + k1 \* log(1 - ɑ2) + k0 = y2 = C(ɑ2)

Подставим коэффиценты в уравнения и получим следующие значения (сначала для **линейной** системы, потом для **случайной**): вместо ɑ возьмём 0,3 , 0,6 , 0,9 , а вместо y - 4, 9, 13 и 2, 4, 6 соответственно. Получим следующие коэффиценты:

1. K2 = ; k1 = ; k0 =
2. K2 = ; k1 = ; k0 =

**Тогда наши формулы имеют следующий вид:**

Y = ) \* (1 - ɑ)-1 + ) \* log(1 - ɑ) + , где ɑ < 1

и

Y = ) \* (1 - ɑ)-1 + ) \* log(1 - ɑ) + , где ɑ < 1