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 № 3**

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

Выполнил студент группы 25/2                                       Б. В. Ким

Направление подготовки  02.03.03  Математическое обеспечение и администрирование информационных систем

Курс    2

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

Краснодар

2023 г.

**Задание**: Найти зависимости средней временной сложности T(n) алгоритма поиска изображений на квадратной картинке размера n x n для n от 100 до 1000 с шагом 100.

**Решение**.

Для анализа сложности алгоритма написана программа, которая считает и выдает средние значения выполненных алгоритмом операций для 1000 случайных массивов с картинками с символом знака меньше. Длина массивов меняется от 100 до 1000.

Текст программы:

#include <fstream>

#include <iostream>

using namespace std;

int\*\* P;

int MAX\_SIZE = 1000;

int n = 100;

int S() {

int k = 0;

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

int size = 0, x = 0, y = 0;

k += 3;

for (int j = 0; j < n; j++) {

if (P[i][j] == 1) {

x = i;

y = j;

k += 3;

while (P[i][y + 1] != 0) {

y++;

k += 3;

}

k += 2;

while (P[i - 1][y] == 0)

{

i++;

size++;

k += 4;

}

k += 2;

x = i;

k += 1;

}

k += 1;

}

/\*if (size != 0) {

cout << "Изображение знака меньше найдено!\n";

cout << "Координаты левого верхнего угла: (" << x - size / 2 << ", " << y - (size-1)<< ")\n";

cout << "Координаты нижнего правого угла: (" << x + size / 2 << ", " << y + (size-1) << ")\n";

cout << endl;

}\*/

}

return k;

}

void F(int x,int y,int t)

{

int x1 = x, y1 = y;

bool flag\_1 = 0;

int l = 0;

while (!flag\_1 or l<t+t-1)

{

while (l < t + t - 1 and !flag\_1)

{

while (l<t and P[x1+1][y1]!=1 and P[x1-1][y1]!=1 and P[x1][y1+1]!=1 and P[x1][y1-1]!=1 and !flag\_1)

{

l++;

if (P[x1][y1] != 0)

flag\_1 = 1;

if (P[x1 + 1][y1] == 1 or P[x1 - 1][y1] == 1 or P[x1][y1 + 1] == 1 or P[x1][y1 - 1] == 1)

flag\_1 = 1;

x1++;

y1 -= 1;

}

while ( l < t + t - 1 and P[x1 + 1][y1] != 1 and P[x1 - 1][y1] != 1 and P[x1][y1 + 1] != 1 and P[x1][y1 - 1] != 1 and !flag\_1)

{

l++;

if (P[x1][y1] != 0)

flag\_1 = 1;

x1++;

y1++;

}

break;

}

break;

}

l = 0;

x1 = x;

y1 = y;

if (!flag\_1)

{

while (l <t-1 ) {

P[x1][y1] = 1;

l++;

x1++;

y1 -= 1;

}

while (l < t + t - 1)

{

P[x1][y1] = 1;

l++;

x1++;

y1++;

}

}

else

{

cout << "Не получится";

}

}

int main()

{

srand(time(0));

ofstream f("myfile.txt");

setlocale(LC\_ALL, "rus");

for (n = 100; n <= MAX\_SIZE; n += 100)

{

P = new int\* [n];

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

{

P[i] = new int [n];

for (int j = 0; j < n; j++)

{

P[i][j] = 0;

}

}

int x = rand() % 5;

int y = rand() % 5;

int t = rand() % 5;

F(x,y,t);

int k = S();

cout << "n=" << n << ": " << k << endl;

f << "n=" << n << ": " << k << endl;

}

}

Программа записывает в файл результаты в следующем виде:

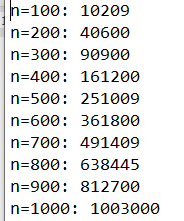


График роста среднего количества операций для сложности (объема) исходных данных *n* в диапазоне от 100 до 1000 выглядит следующим образом:

Анализ зависимости:

Решаем задачу анализа методом неопределенных коэффициентов. Полученный в результате расчетов график очень близок к графику зависимости

, где *n* > 0

Найдем неизвестные коэффициенты *a*, *b*, *c*.

Поскольку параболу можно построить по трем точкам, возьмем некоторые три точки на графике (*n* = 100, *y* = 10209); (*n* = 200, *y* = 40600); (*n* = 300; *y* = 90900) и решим систему из 3 уравнений

a = 0.99545, b = 5.275, c = – 273.

Возьмем приблизительные значения. Получим формулу вида:

, где *n* > 0.

**Вывод**: Средняя временная сложность T(n) алгоритма поиска изображений на квадратной картинке из *n2* элементов имеет вид

, где *n* > 0.