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 № 2**

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

Выполнил студент группы 25/2                                       А.А. Козин

Направление подготовки  02.03.03  Математическое обеспечение и администрирование информационных систем

Курс    2

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

Краснодар

2021 г.

**Задание**: Найти функции минимально, средней и максимальной сложности для алгоритма Дейкстры.

**Решение**.

Для анализа сложности алгоритма написана программа, которая считает и выдает значения выполненных операций алгоритмом. Для среднего значения алгоритм выполняется для 1000 случайных графов. При этом количество вершин графов меняется от 5 до 30.

**Код программы:**

#include<iostream>

#include<ctime>

#include<set>

#include<vector>

using namespace std;

int maximum = INT16\_MAX;

struct Vertex {

int visited;

int weight;

vector <pair <int, int>> neighbors;

};

typedef vector<Vertex> Graph;

int Deikstra(Graph graph, int vertex) {

int counter = 0;

set <pair<int, int>> s;

s.insert(make\_pair(0, vertex));

while (!s.empty()) {

pair<int, int> p = \*(s.begin());

counter++;

s.erase(s.begin());

counter++;

if (graph[p.second].visited == 1) {

counter++;

continue;

}

graph[p.second].visited = 1;

counter++;

graph[p.second].weight = p.first;

counter++;

for (auto i = graph[p.second].neighbors.begin(); i != graph[p.second].neighbors.end(); ++i)

if (graph[i->first].visited == 0) {

s.insert(make\_pair(graph[p.second].weight + (i->second), i->first));

counter += 2;

}

}

return counter;

}

Graph rand(int size)

{

Graph graph;

graph.resize(size);

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

for (int j = i + 1; j < size; j++) {

int q = rand() % 5;

if (q)

{

graph[i].neighbors.push\_back(make\_pair(j, q));

graph[j].neighbors.push\_back(make\_pair(i, q));

}

}

}

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

{

graph[i].visited = 0;

graph[i].weight = maximum;

}

return graph;

}

Graph max(int size)

{

Graph graph;

graph.resize(size);

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

for (int j = i + 1; j < size; j++) {

graph[i].neighbors.push\_back(make\_pair(j, 5));

graph[j].neighbors.push\_back(make\_pair(i, 5));

}

}

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

{

graph[i].visited = 0;

graph[i].weight = maximum;

}

return graph;

}

Graph min(int size) {

Graph graph;

graph.resize(size);

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

graph[i].visited = 0;

graph[i].weight = maximum;

}

return graph;

}

void main()

{

srand(time(NULL));

setlocale(LC\_ALL, "ru");

cout << "size| " << "|Среднее значение| " << "|Max| " << "|Min| " << endl;

cout << "-----------------------------------------------------" << endl;

for (int size = 5; size <= 30; size++) {

int average = 0;

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

average += Deikstra(rand(size), rand() % size);

cout << size << "| |" << float(average) / 1000 <<"| |"<< Deikstra(max(size), 0) << "| |" << Deikstra(min(size), 0) << "|" << endl;

}

system("pause>nul");

}

Программа выдает результат в консоль в следующем виде:

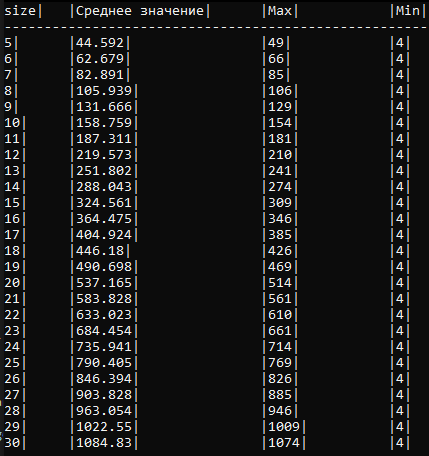


График роста количества операций для **минимальной** сложности (объема) исходных данных *n* в диапазоне от 5 до 30 выглядит следующим образом:

График роста количества операций для **максимальной** сложности (объема) исходных данных *n* в диапазоне от 5 до 30 выглядит следующим образом:

График роста количества операций для **средней** сложности (объема) исходных данных *n* в диапазоне от 5 до 30 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 , , , , *.*

Для **минимальной** сложности:

Возьмем некоторые пять точек на графике (*n* = 5, *y* = 4); (*n* = 10, *y* = 4); (*n* = 15; *y* = 4); (n = 20; y = 4); (n = 30; y = 4) и решим систему из 5 уравнений

, , , ,

, где *n* > 0.

Для **средней** сложности:

Возьмем следующие пять точек на графике (*n* = 5, *y* = ); (*n* = 10, *y* = ); (*n* = 15; *y* = ); (n = 20; y = ); (n = 30; y = ) и решим систему из 5 уравнений

, , , ,

, где *n* > 0.

Для **максимальной** сложности:

Возьмем следующие пять точек на графике (*n* = 5, *y* = 49 ); (*n* = 10, *y* = 154); (*n* = 15; *y* = 309); (n = 20; y = 514); (n = 30; y = 1074) и решим систему из 5 уравнений

, , , ,

, где *n* > 0.

**Вывод**:

*Минимальная* сложность алгоритма Дейкстры для графов с *n* вершинами имеет вид

, где *n* > 0.

*Средняя*:

, где *n* > 0.

*Максимальная*:

, где *n* > 0