**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Систем автоматизированного проектирования**

отчет

**по лабораторной работе №2**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: «**Алгоритм** **Дейкстры**»

|  |  |  |
| --- | --- | --- |
| Студент гр. 1302 |  | Арепьев В.О. |
| Преподаватель |  | Родионова Е.А. |

Санкт-Петербург

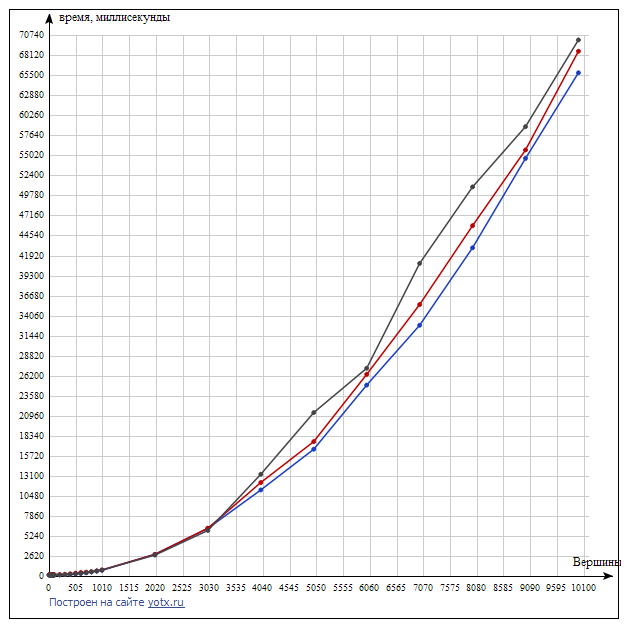
2023

### Исходная постановка задания

Реализовать алгоритм Дейстры с помощью выбранной базовой структуры данных, двоичной кучи и Фибоначчиевой кучи. Сравнить временные затраты на выполнение данных реализаций алгоритма для графов низкой/средней/высокой связности. Отчет должен содержать графики и выводы по проделанной работе.

### Длина деревьев

Синий график граф низкой связности, красный граф низкой связности, серый граф низкой связности.



### Листинг

#include <iostream>

#include <vector>

using namespace std;

const int INF = 1e9;

struct Node

{

int key, value;

Node(int k, int v) : key(k), value(v) {}

};

class BinaryHeap

{

vector<Node> heap;

public:

void push(Node node)

{

heap.push\_back(node);

int i = heap.size() - 1;

while (i > 0 && heap[(i - 1) / 2].key > node.key)

{

heap[i] = heap[(i - 1) / 2];

i = (i - 1) / 2;

}

heap[i] = node;

}

Node top()

{

return heap[0];

}

void pop() {

if (heap.empty())

{

return;

}

heap[0] = heap.back();

heap.pop\_back();

if (heap.empty())

{

return;

}

int i = 0;

while (i < heap.size())

{

int left = 2 \* i + 1;

int right = 2 \* i + 2;

if (left >= heap.size())

{

break;

}

int j = left;

if (right < heap.size() && heap[right].key < heap[left].key)

{

j = right;

}

if (heap[i].key <= heap[j].key)

{

break;

}

swap(heap[i], heap[j]);

i = j;

}

}

bool empty()

{

return heap.empty();

}

};

int main() {

setlocale(LC\_ALL, "Russian");

int n, s;

cout << "Введите количество вершин и исток (нумерация вершин начинается с 0)\n";

cin >> n >> s;

cout << "Матрица смежности\n";

vector<vector<int>> graph(n, vector<int>(n));

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

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

{

cin >> graph[i][j];

}

}

BinaryHeap heap;

vector<int> dist(n, INF);

heap.push(Node(0, s));

dist[s] = 0;

while (!heap.empty())

{

int u = heap.top().value;

heap.pop();

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

{

if (v != u && v >= 0 && v < n && graph[u][v] != 0)

{

int alt = dist[u] + graph[u][v];

if (alt < dist[v])

{

dist[v] = alt;

heap.push(Node(dist[v], v));

}

}

}

}

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

{

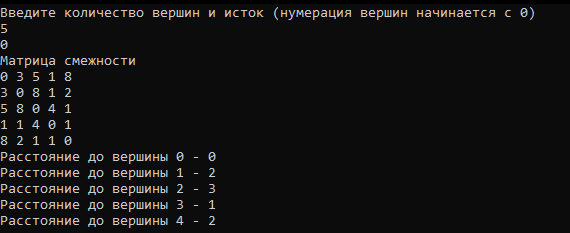
cout << "Расстояние до вершины " << i << " - " << dist[i] << endl;

}

return 0;

### }

### Пример работы программы



### Вывод

В ходе выполнения данной работы был реализован алгоритм Дейкстры при помощи бинарной кучи. Как видно из графиков временных затрат алгоритм работает эффективнее на графе меньшей связности.