|  |
| --- |
| **МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ** |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ** |
| **УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ** |
| **«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»** |
|  |
| Институт информационных технологий и управления в технических системах |
| (полное название института) |
|  |
| кафедра «Информационные системы» |
| (полное название кафедры) |

**ОТЧЁТ**

по лабораторной работе №5

на тему

«Поиск кратчайших путей на графах»

Выполнил

студент ИИТУТС

группы ИС/б-18-2-о

Радыгина Екатерина

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | | | | | | | |
|  | (должность, учёная степень преподавателя) | | | | | | | | |
|  |  | | | | | | | | |
|  | (ФИО преподавателя) | | | | | | | | |
|  | « |  | » |  |  |  | 20 | 20 | г. |
|  |  | | | | | | | | |
|  | (оценка) | | | | | | | | |
|  |  | | | | | | | | |
|  |  | | | | | | | | |

Севастополь 2020

**5.1 Цель работы**

Целью работы является изучение алгоритмов поиска кратчайших путей на графах на примере метода динамического программирования.

**5.2 Вариант задания**

Составлен граф (рисунок 5.1). Создать программу, реализующую метод динамического программирования и алгоритм топологической сортировки вершин. Исходный граф задается в виде матрицы смежности, вводимой построчно с помощью консоли. Создать программу, которая использует приведенный в данной работе алгоритм Дейкстры для заданного графа.

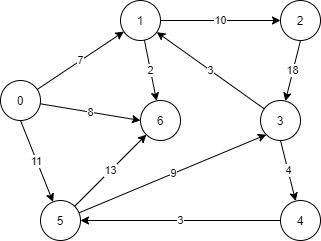


Рисунок 5.1 – Составленный граф

**5.3 Ход работы**

5.3.1 Составим структурную схему программы, определяющей кратчайший путь на графе от заданной начальной вершины s до заданной конечной вершины t с помощью метода динамического программирования (рисунок 5.2).

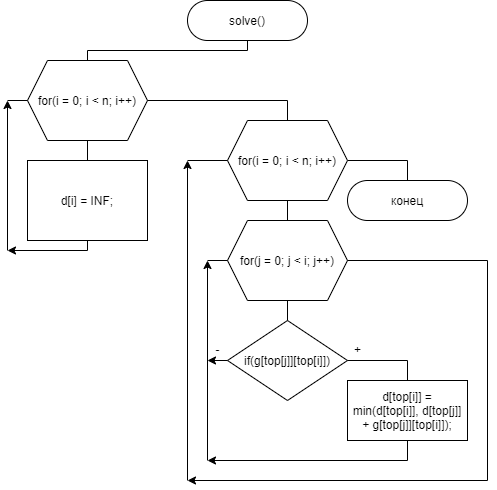


Рисунок 5.2 – Схема алгоритма solve

5.3.2 Составим структурную схему программы, реализующей алгоритм топологической сортировки с произвольной нумерацией вершин графа (рисунок 5.3).

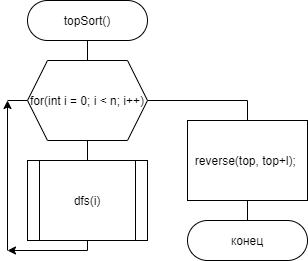


Рисунок 5.3 – Схема алгоритма topSort

5.3.3 Создадим программу, реализующую метод динамического программирования и алгоритм топологической сортировки вершин. Исходный граф задается в виде матрицы смежности, вводимой построчно с помощью консоли. Используем приведенный в данной работе алгоритм Дейкстры для заданного графа.

Текст программы.

#include <iostream>

#include <iomanip>

#include <algorithm>

#include <climits>

using namespace std;

const int INF = 1000000000;

bool used[100] = {0};

int top[100] = {0};

int g[100][100] = {0};

int d[100] = {0};

int n, l, s, f;

void dfs(int v) {

if(used[v])

return;

used[v] = true;

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

if(g[v][to])

dfs(to);

top[l++] = v;

}

void topSort(){

l = 0;

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

dfs(i);

reverse(top, top+l);

}

void solve(){

int i, j;

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

d[i] = INF;

d[s] = 0;

for(i = 1; i < n; i++)

for(j = 0; j < i; j++)

if(g[top[j]][top[i]])

d[top[i]] = min(d[top[i]], d[top[j]] + g[top[j]][top[i]]);

}

void djeikstra(){

int i,j,

v,

to,

len;

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

d[i] = INF;

d[s] = 0;

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

v = -1;

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

if(!used[j] && (v == -1 || d[j] < d[v]))

v = j;

used[v] = true;

for(to = 0; to < n; to++){

if(g[v][to]){

len = g[v][to];

if(d[v] + len < d[to]){

d[to] = d[v] + len;

}

}

}

}

}

int main(){

n = 7;

cout << "Input start vertex:" << endl;

cin >> s;

cout << "Input finish vertex:" << endl;

cin >> f;

cout << "Input matrix 7x7:" << endl;

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

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

cin >> g[i][j];

}

}

for(int i = 0; i < 1000000; i++){

for(int j = 0; j < 100; j++)

used[j] = false;

djeikstra();

}

cout << "By Djeikstra: " << d[f] << endl;

for(int i = 0; i < 1000000; i++){

for(int j = 0; j < 100; j++)

used[j] = false;

topSort();

solve();

}

cout << "By DP: " << d[f] << endl;

return 0;

}

Тест программы (рисунок 5.4).

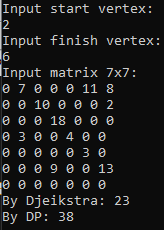


Рисунок 5.4 – Тест программы

**Вывод**

Мы изучили алгоритмы поиска кратчайших путей на графах на примере метода динамического программирования.