Лабораторная работа №5

«Поиск кратчайших путей в графе»

Цель работы

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

Вариант 1

Ход работы

Для решения задачи поиска кратчайшего пути в графе была написана программа, реализующая поиск кратчайшего пути методом динамического программирования с топологической сортировкой а также алгоритмом Дейкстры.

Код программы:

#include <iostream>

#include <iomanip>

#include <algorithm>

#include <chrono>

#include <climits>

#include <fstream>

using namespace std;

const int INF = 1000000000;

bool used[100] = { 0 };

int top[100] = { 0 };

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

int n;

int l;

int s;

int f;

int d[100] = { 0 };

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(int argc, char\*\* argv) {

n = 9;

fstream file("D:\\уник\\ЛабыОТАлг\\Л5\\Л5\\Debug\\1.txt");

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

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

file >> g[i][j];

}

}

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

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

cout << g[i][j] << '\t';

}

cout << endl;

}

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

cin >> s;

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

cin >> f;

auto start\_time = chrono::high\_resolution\_clock::now();

auto end\_time = chrono::high\_resolution\_clock::now();

long long time;

start\_time = chrono::high\_resolution\_clock::now();

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

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

used[j] = false;

djeikstra();

}

end\_time = chrono::high\_resolution\_clock::now();

time = chrono::duration\_cast<chrono::milliseconds>(end\_time - start\_time).count();

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

cout << "Time: " << time << endl;

start\_time = chrono::high\_resolution\_clock::now();

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

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

used[j] = false;

topSort();

solve();

}

end\_time = chrono::high\_resolution\_clock::now();

time = chrono::duration\_cast<chrono::milliseconds>(end\_time - start\_time).count();

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

cout << "Time: " << time << endl;

return 0;

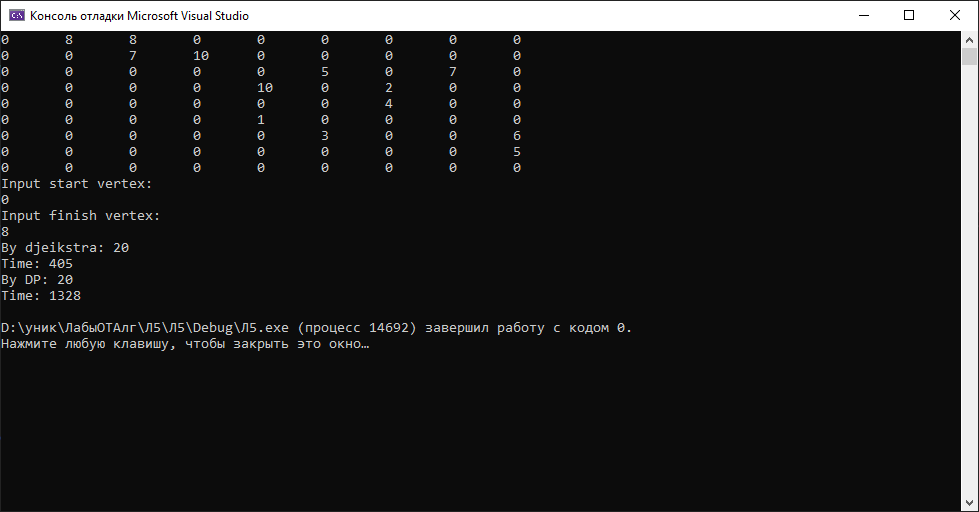
}

Рисунок 1 – Результат выполнения программы

Таким образом, алгоритм Дейкстры осуществляет поиск кратчайшего пути осуществила в 3 раза быстрее по суммарному времени 1000000 прогонов.

Вывод

В ходе выполнения лабораторной работы было выполнено ознакомление с алгоритмами динамического программирования с топологической сортировкой и алгоритмом Дейкстры для поиска кратчайшего пути в графе