**5 Лабораторная работа №5**

**«ПОИСК КРАТЧАЙШИХ ПУТЕЙ НА ГРАФАХ»**

**5.1** **Цель работы**

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

**5.2 Вариант задания ­– 14**

Создать программу, реализующую метод динамического программирования и алгоритм топологической сортировки вершин. Исходный граф задается в виде матрицы смежности, вводимой построчно с помощью консоли. Указание: для определения вершин, входящих во множество Г-1 (xi) используйте j-й столбец матрицы смежности.

Создать программу, которая использует приведенный в данной работе алгоритм Дейкстры для заданного графа.

Сравнить время выполнения двух алгоритмов.

Таблица 5.1 – Граф по варианту представленный в виде таблицы

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 1 |  | 6 | 1 |  |  |  |  |  |  |
| 2 |  |  | 8 |  |  |  |  |  |  |
| 3 |  |  |  |  |  | 6 |  |  |  |
| 4 |  |  |  |  | 3 |  | 1 | 1 |  |
| 5 |  |  |  |  |  |  | 5 |  |  |
| 6 |  |  |  | 2 | 4 |  |  |  |  |
| 7 |  |  |  |  |  | 5 |  |  | 4 |
| 8 |  |  |  |  |  |  |  |  | 3 |

**5.3** **Ход работы**

С помощью таблицы 5.1 был создан граф, представленный на рисунке 5.1.

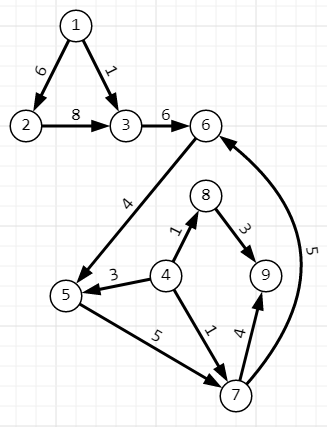


Рисунок 5.1 – Граф заданный по варианту

Была написана программа на языке С++ выполняющая требуемые задания:

#include <iostream>

#include <iomanip>

#include <algorithm>

#include <chrono>

#include <climits>

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 = 10;

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

cin >> s;

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

cin >> f;

cout << "Input matrix 10x10:" << endl;

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

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

cin >> g[i][j];

}

}

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;

}

Результаты, полученные в ходе выполнения программы, отображены таблице 5.2.

Таблица 5.2 – Результаты работы программы

|  |  |  |  |
| --- | --- | --- | --- |
| Алгоритм | Кол-во прогонов | Кол-во вершин | Время |
| Дейкстры | 1000000000 | 9 | 540 |
| Динамического программирования | 613 |

**Выводы**

В ходе лабораторной работы были изучены алгоритмы поиска кратчайшего пути на графе на примере алгоритма Дейкстры и метода динамического программирования. Сделан вывод, что алгоритм Дейкстры работает быстрее чем алгоритм динамического программирования.