Лабораторная работа №5

“Поиск кратчайших путей на графах”

1 ЦЕЛЬ РАБОТЫ

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

2 ПОСТАНОВКА ЗАДАЧИ

1. По таблице нарисовать исходный граф.
2. Составить структурную схему программы, определяющей кратчайший путь на графе от заданной начальной вершины до заданной конечной вершины с помощью метода динамического программирования.
3. Составить структурную схему программы, реализующей алгоритм топологической сортировки с произвольной нумерацией вершин графа.
4. Создать программу, реализующую метод динамического программирования и алгоритм топологической сортировки вершин. Исходный граф задается в виде матрицы смежности, вводимой построчно с помощью консоли.
5. Создать программу, которая использует приведенный в данной работе алгоритм Дейкстры для заданного графа.
6. Сравнить время выполнения двух алгоритмов.



Рисунок 1 – Задание по варианту

3 ХОД РАБОТЫ

1. Создадим исходный граф по таблице:

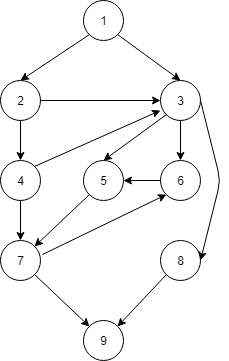


Рисунок 2 – Исходный граф к варианту

1. Текст программы, реализующая алгоритм Дейкстры для заданного вариантом графа.

#include <iostream>

#include <windows.h>

#include <ctime>

using namespace std;

const int n = 9; // количество вершин

const int MAX\_INT = 1000000;

void solve(int g[n][n], int d[n], int s)

{

int i, j, // i-я итерация, j - для поиска минимальной

v, // минимальная вершина

to, // ребро из вершины v в to

len; // длины len

bool used[n] = {false}; // массив для пометок вершин

d[s] = 0;

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

{

v = -1;

for (j = 0; j < n; j++) // поиск вершины с минимальным d[j]

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

v = j;

used[v] = true; // пометка вершины

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;

}

}

}

}

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

cout << d[i] << " ";

}

int main()

{

double start\_time, end\_time;

int s = 0; // стартовая вершина

int g[n][n] = {

0, 2, 1, 0, 0, 0, 0, 0, 0,

0, 0, 7, 2, 0, 0, 0, 0, 0,

0, 0, 0, 0, 2, 3, 0, 6, 0,

0, 0, 3, 0, 0, 0, 8, 0, 0,

0, 0, 0, 0, 0, 0, 4, 0, 0,

0, 0, 0, 0, 3, 0, 0, 0, 0,

0, 0, 0, 0, 0, 2, 0, 0, 10,

0, 0, 0, 0, 0, 0, 0, 0, 1,

0, 0, 0, 0, 0, 0, 0, 0, 0,

}; // матрица смежности

int d[n]; // массив ответов

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

d[i] = MAX\_INT;

start\_time = clock();

cout << "Кратчайшие расстояния:" << endl;

solve(g, d, s);

end\_time = clock();

double time = end\_time - start\_time;

cout<<"Время работы программы (MS): "<<time/(double)CLOCKS\_PER\_SEC<< endl;

return 0;

}

1. Текст программы, реализующая метод динамического программирования и алгоритм топологической сортировки вершин.

#include <iostream>

#include <algorithm>

#include <ctime>

using namespace std;

const int INF = 1000000000; // бесконечность

const int n = 9; // количество вершин

bool used[n] = { 0 }; // массив для пометок

int top[n] = { 0 }; // топологический список

int g[n][n] = {

0, 2, 1, 0, 0, 0, 0, 0, 0,

0, 0, 7, 2, 0, 0, 0, 0, 0,

0, 0, 0, 0, 2, 3, 0, 6, 0,

0, 0, 3, 0, 0, 0, 8, 0, 0,

0, 0, 0, 0, 0, 0, 4, 0, 0,

0, 0, 0, 0, 3, 0, 0, 0, 0,

0, 0, 0, 0, 0, 2, 0, 0, 10,

0, 0, 0, 0, 0, 0, 0, 0, 1,

0, 0, 0, 0, 0, 0, 0, 0, 0,

}; // матрица смежности

int l; // l-я вершина для добавления

int s = 0; // стартовая вершина

int f; // конечная вершина

int d[n] = { 0 }; // массив ответов

int dfs(int v)

{

if (used[v])

return 0;

used[v] = true;

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

if (g[v][to])

dfs(to);

top[l++] = v; // добавление вершины v в отсортированный список

}

int topSort()

{

l = 0; // номер добавляемой вершины в отсортированный список

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

dfs(i); // запустить пробежку из всех вершин

reverse(top, top + l); // развернуть массив

return 0;

}

int 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]]);

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

cout << d[i] << " ";

return 0;

}

int main()

{

double start\_time, end\_time;

start\_time = clock();

cout << "Кратчайшие расстояния:" << endl;

topSort();

solve();

end\_time = clock();

double time = end\_time - start\_time;

cout<<"Время работы программы (MS): "<<time/(double)CLOCKS\_PER\_SEC<< endl;

}

4 РЕЗУЛЬТАТЫ

C:\Users\Юрий\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1.png

Рисунок 3 – Результат работы алгоритма Дейкстры



Рисунок 4 – Результат работы алгоритма топологической сортировки

ВЫВОД

В ходе лабораторной работы были изучены алгоритмы поиска кратчайших путей на графах. Также были написаны две программы: первая – реализует метод Дейкстры, вторая – метод топологической сортировки. Сравнив результаты выполнения программы, можно заключить, что алгоритм топологической сортировки работает незначительно быстрее алгоритма Дейкстры.