Министерство образования Республики Беларусь

Учреждение образования

“Брестский Государственный технический университет”

Кафедра ИИТ

Лабораторная работа №

По дисциплине «Математические основы И С»

Тема: ***Нахождение кратчайшего пути в графе***

Выполнил:

Студент 2 курса

Группы ИИ-23

Макаревич Н. Р.

Проверил:

Козинский А. А.

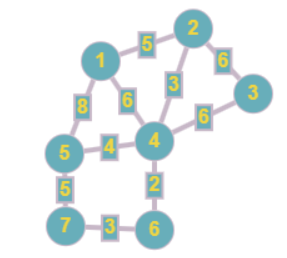
Брест 2023

Вариант 3

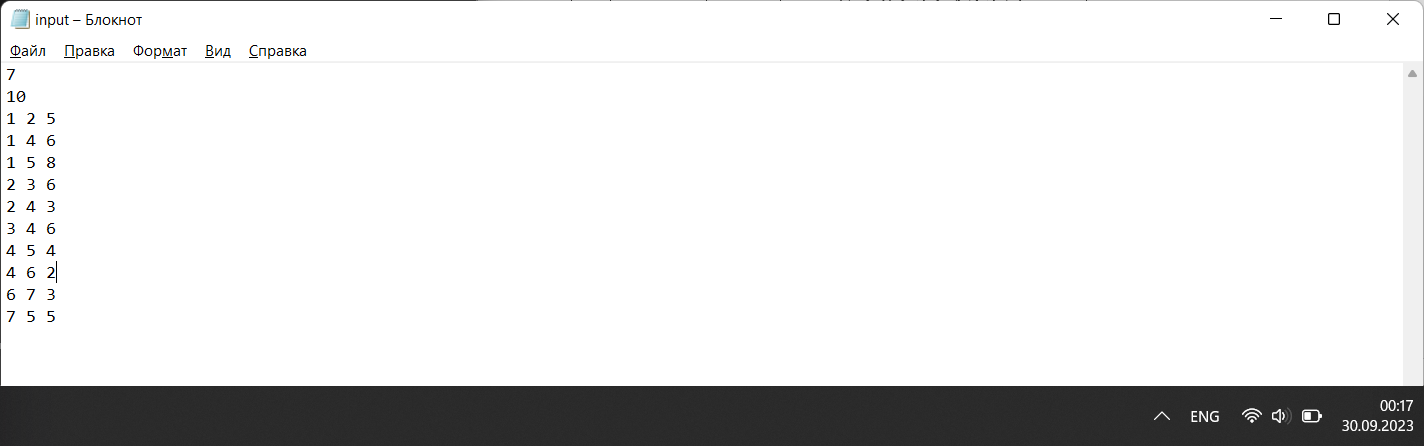
***Задание.***

1. Алгоритмом Дейкстры вычислить кратчайшие пути от вершины  ко всем вершинам графа. Варианты графов указаны в таблице 1. Графы заданы списком ребер, в квадратных скобках указаны веса соответствующих ребер.
2. Алгоритмом Флойда-Уоршолла вычислить кратчайшие пути от вершины ко всем вершинам графа.

|  |  |  |  |
| --- | --- | --- | --- |
| ***№*** | ***Кол.***  ***вершин*** | ***Кол.***  ***ребер*** | ***Задание***  ***графа*** |
| 3 | 7 | 10 | {1,2},{1,4},{1,5},{2,3},{2,4},  {3,4},{4,5},{4,6},{6,7},{7,5};  [5,6,8,6,3,6,4,2,3,5]; |



Файл с входными данными input.txt:



Код программы: #include <iostream>

#include <fstream>

#include <vector>

using namespace std;

class Graph {

public:

string name;

int\*\* matrix;

int numOfVertex, numOfEdge;

Graph(string name) {

this->name = name;

ifstream fin;

fin.open(name);

fin >> numOfVertex >> numOfEdge;

}

void createAdjacencyMatrix() {

ifstream fin;

fin.open(name);

matrix = new int\* [numOfVertex];

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

matrix[i] = new int[numOfVertex];

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

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

matrix[i][j] = 999999;

}

}

int trash;

fin >> trash >> trash;

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

int from, to, weight;

fin >> from >> to >> weight;

matrix[from - 1][to - 1] = weight;

matrix[to - 1][from - 1] = weight;

}

ofstream fout;

fout.open("Adjacency.txt");

fout << "\t";

for (int i = 1; i <= numOfVertex; i++)

fout << i << "\t";

fout << endl;

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

fout << i + 1 << "\t";

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

fout << matrix[i][j] << "\t";

}

fout << endl;

}

fout.close();

fin.close();

}

void Dejkstra() {

ofstream fout("Dejkstra.txt");

vector <int> distance(numOfVertex, 10000);

vector <bool> isVisited(numOfVertex, false);

vector <int> prev(numOfVertex, -1);

distance[0] = 0;

isVisited[0] = true;

prev[0] = 0;

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

if (matrix[0][i] <= distance[i]) {

distance[i] = matrix[0][i];

prev[i] = 0;

}

for (int i = 0; i < numOfVertex - 1; i++) {

int minInd = findSmalest(distance, isVisited);

int minDist = distance[minInd];

isVisited[minInd] = true;

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

if (!isVisited[i] && distance[i] > matrix[minInd][i] + minDist) {

distance[i] = matrix[minInd][i] + minDist;

prev[i] = minInd;

}

}

}

fout << "Vertex\tDist\tPrev" << endl;

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

fout << i + 1 << "\t" << distance[i] << "\t" << prev[i] + 1 << endl;

}

fout.close();

}

void Floyd() {

ofstream fout;

fout.open("Floyd.txt");

int\*\* temp = matrix;

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

temp[i][i] = 0;

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

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

cout << temp[i][j] <<"\t";

}

cout << endl;

}

for(int k = 0; k < numOfVertex; k++)

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

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

if (temp[i][k] + temp[k][j] < temp[i][j])

temp[i][j] = temp[i][k] + temp[k][j];

fout << "\t";

for (int i = 1; i <= numOfVertex; i++)

fout << i << "\t";

fout << endl;

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

fout << i + 1 << "\t";

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

fout << temp[i][j] << "\t";

}

fout << endl;

}

fout.close();

}

private:

int findSmalest(vector <int> dist, vector <bool> isVisited) {

int min = 999999;

for(int i = 0; i < dist.size();i++){

if (dist[i] < min && !isVisited[i]) {

min = i;

}

}

return min;

}

};

int main()

{

string name;

cout << "Enter name of the file: " << endl;

cin >> name;

Graph graph(name);

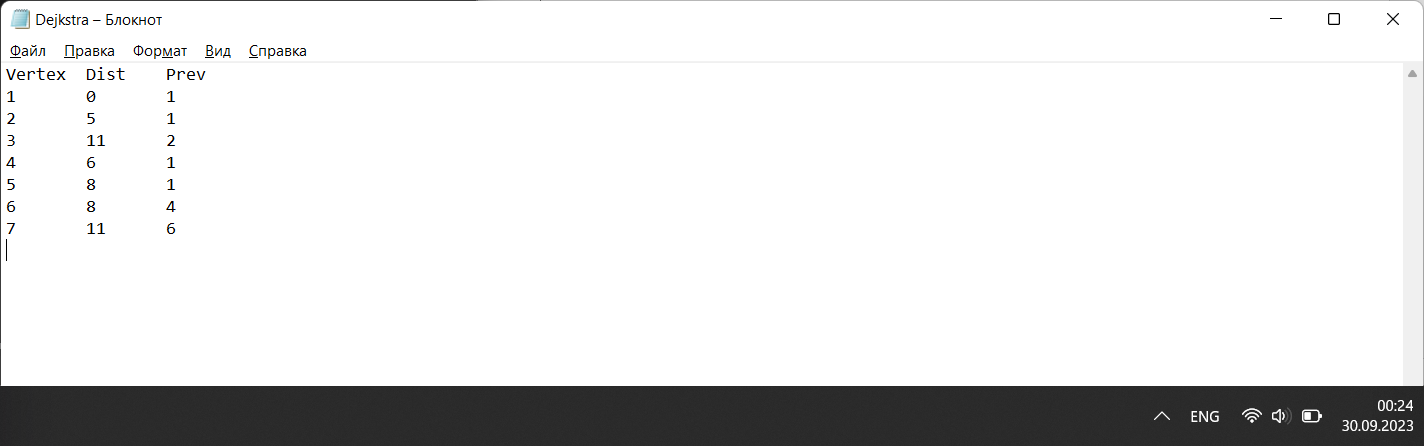
graph.createAdjacencyMatrix();

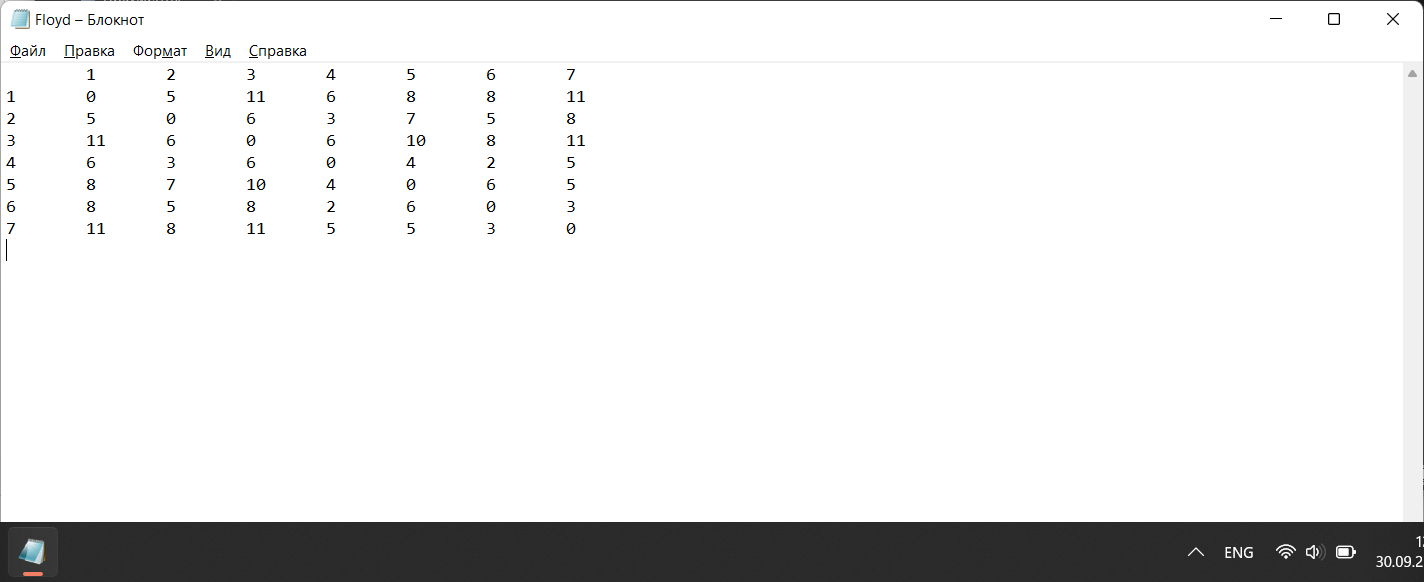
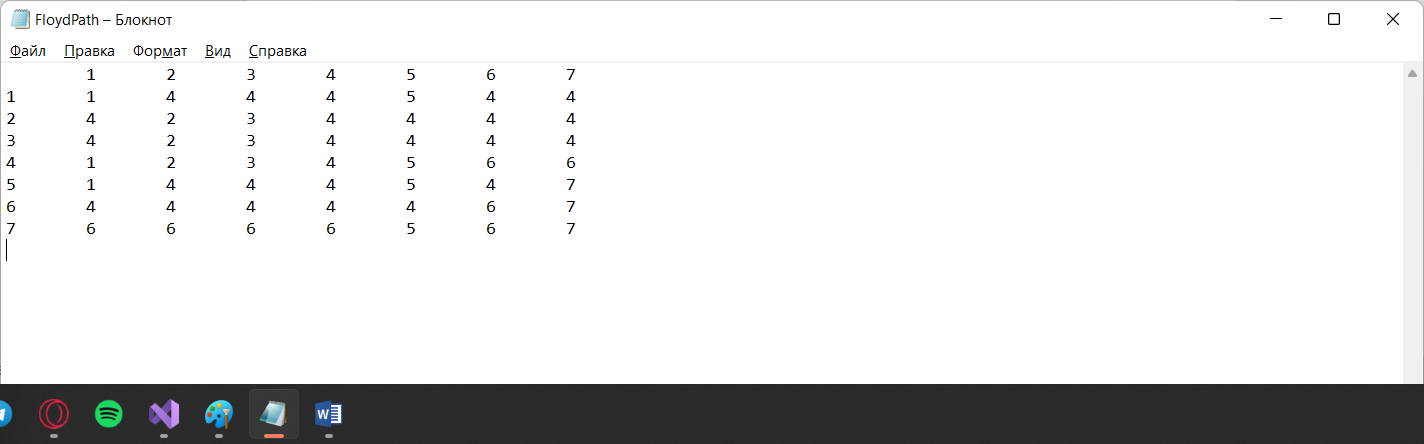
graph.Dejkstra();

graph.Floyd();

}

Вывод программы:



Вывод:

Алгоритмами Дейкстры и Флойда-Уоршела вычислил кратчайшие пути от вершины x1 ко всем остальным вершинам.