using System;

using System.IO;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Diagnostics;

using Microsoft.Office.Interop.Excel;

namespace Экзамен\_2021\_ПП

{

public class Program

{

public static void Main(string[] args)

{

izm7: string listname = string.Empty;

Console.WriteLine("Какой лист хотите выбрать? ");

switch (Console.ReadLine())

{

case "1":

Console.WriteLine("Выбран 1 лист. Подождите минуту.");

listname = "Лист1";

break;

default:

Console.WriteLine("Такого листа нет.");

break;

}

Debug.WriteLine("Выбор между листами файла Excel");

}

static string listname = string.Empty;

static string filename = @"C:\Users\1\Downloads\komivoyazher.xlsx";

static int column = 0;

double[,] table = Excel.GetArray(filename, listname, out column);

double sum = 0;

//выбор листа пользователем в файле Excel

}

public class konec

{

public void k()

{

var g = new g();

//добавление вершин

g.AddVertex("1");

g.AddVertex("2");

g.AddVertex("3");

g.AddVertex("4");

g.AddVertex("5");

g.AddVertex("6");

g.AddVertex("7");

//добавление ребер

g.AddEdge("1", "2", 3);

g.AddEdge("1", "3", 7);

g.AddEdge("1", "4", 1);

g.AddEdge("2", "4", 8);

g.AddEdge("3", "4", 9);

g.AddEdge("4", "5", 4);

g.AddEdge("4", "6", 2);

g.AddEdge("5", "6", 6);

g.AddEdge("6", "7", 3);

Console.WriteLine(g);

var dijkstra = new D(g);

var path = dijkstra.FindShortestPath("1", "7");

Console.WriteLine(path);

Debug.WriteLine("ввод данных о вершинах");

Debug.Assert(false, "Произошла ошибка");

}

}

public class gvi1

{

public gv v { get; set; }

public bool IU { get; set; }

public int ews { get; set; }

public gv pv { get; set; }

public gvi1(gv v)

{

this.v = v;

IU = true;

ews = int.MinValue;

pv = null;

}

}

public class g

{

public List<gv> Vertices { get; }

public g()

{

Vertices = new List<gv>();

}

public void AddVertex(string vertexName)

{

Vertices.Add(new gv(vertexName));

}

public gv FindVertex(string vertexName)

{

foreach (var v in Vertices)

{

if (v.Name.Equals(vertexName))

{

return v;

}

}

return null;

}

public void AddEdge(string firstName, string secondName, int weight)

{

var v1 = FindVertex(firstName);

var v2 = FindVertex(secondName);

if (v2 != null && v1 != null)

{

v1.AddEdge(v2, weight);

v2.AddEdge(v1, weight);

}

Console.WriteLine("{0},{1}", v1, v2);

Debug.WriteLine("Вывод вершин");

// вывод вершин графа

Debug.Assert(false, "Произошла ошибка");

}

}

public class D

{

g g;

List<gvi1> infos;

public D(g graph)

{

this.g = graph;

}

void InitInfo()

{

infos = new List<gvi1>();

foreach (var v in g.Vertices)

{

infos.Add(new gvi1(v));

}

}

gvi1 GetVertexInfo(gv v)

{

foreach (var i in infos)

{

if (i.v.Equals(v))

{

return i;

}

}

return null;

}

public gvi1 FindUnvisitedVertexWithMinSum()

{

var maxValue = int.MinValue;

gvi1 maxVertexInfo = null;

foreach (var i in infos)

{

if (i.IU && i.ews > maxValue)

{

maxVertexInfo = i;

maxValue = i.ews;

}

}

return maxVertexInfo;

}

public string FindShortestPath(string startName, string finishName)

{

Console.WriteLine("Начало пути: {0}", startName);

Console.WriteLine("Конец пути: {0}", finishName);

return FindShortestPath(g.FindVertex(startName), g.FindVertex(finishName));

// вывод нача и конца путей графа

}

public string FindShortestPath(gv startVertex, gv finishVertex)

{

InitInfo();

var first = GetVertexInfo(startVertex);

first.ews = 0;

while (true)

{

var current = FindUnvisitedVertexWithMinSum();

if (current == null)

{

break;

}

SetSumToNextVertex(current);

}

Debug.WriteLine("Вывод начала и конца пути");

Debug.Assert(false, "Произошла ошибка");

Debug.Assert(true, "Всё верно");

return GetPath(startVertex, finishVertex);

}

void SetSumToNextVertex(gvi1 info)

{

info.IU = false;

foreach (var e in info.v.Edges)

{

var nextInfo = GetVertexInfo(e.ConnectedVertex);

var sum = info.ews + e.EdgeWeight;

if (sum > nextInfo.ews)

{

nextInfo.ews = sum;

nextInfo.pv = info.v;

Console.WriteLine("Сумма после добавления предыдущего временного промежутка: {0}", sum);

}//подсчёт суммы значений путей графа

}

Debug.WriteLine("Вывод и подсчёт суммы");

Debug.Assert(false, "Произошла ошибка");

//вывод суммы значений на путях графа

}

public string GetPath(gv startVertex, gv endVertex)

{

var path = endVertex.ToString();

while (endVertex == null)

{

endVertex = GetVertexInfo(endVertex).pv;

path = "Номер вершины:" + endVertex.ToString() + "; " + path;

}//вывод вершин

return path;

}

void fl(string p)

{

StreamWriter sw;

}

void wtf(string p1)

{

using (StreamWriter sw = new StreamWriter(p1, false))

{

StreamReader sr;

const int NmaxZap = 10;

sr = new StreamReader(@"\\main\RDP\31П\СергеевДИ\Desktop\практика.txt", UTF8Encoding.Default);

string[] d = new string[NmaxZap];

string t = sr.ReadLine();

int i = 0;

while ((t != null) && (i < d.Length))

{

Console.WriteLine(t);

d[i++] = t;

t = sr.ReadLine();

}

sr.Close();

}// чтение из файла

}

}

public class ge

{

public gv ConnectedVertex { get; }

public int EdgeWeight

{

get;

}

public ge(gv connectedVertex, int weight)

{

ConnectedVertex = connectedVertex;

EdgeWeight = weight;

Console.WriteLine("Затраты времени на ребре графа: {0}", weight);

// введение информация о затратах времени на ребре графа

}

}

public class gv

{

public string Name { get; }

public List<ge> Edges { get; }

public gv(string vertexName)

{

Name = vertexName;

Edges = new List<ge>();

}

public void AddEdge(ge newEdge)

{

Edges.Add(newEdge);

}

public void AddEdge(gv vertex, int edgeWeight)

{

AddEdge(new ge(vertex, edgeWeight));

}

public override string ToString() => Name;

}

public class Excel

{

public static double[,] GetArray(string filename, string listname, out int column)

{

Application xlApp = new Application(); //Excel

Workbook xlWB; //рабочая книга

Worksheet xlSht; //лист Excel

column = 0;

xlWB = xlApp.Workbooks.Open(filename); //название файла Excel

xlSht = (Worksheet)xlWB.Worksheets[listname]; //название листа или 1-й лист в книге xlSht = xlWB.Worksheets[1];

int iLastRow = xlSht.Cells[xlSht.Rows.Count, "C"].End[XlDirection.xlUp].Row; //последняя заполненная строка в столбце А

object[,] arrData = (object[,])xlSht.Range["B3:Z" + iLastRow].Value; //берём значения из диапазона в массив

for (var i = 1; i <= iLastRow - 2; i++)

{

for (var j = 1; j < arrData.GetLength(1); j++)

{

if (arrData[i, j] == null)

continue;

column++;

}

break;

}

double[,] table = new double[iLastRow - 2,

column];

for (var i = 1; i <= iLastRow - 2; i++)

{

for (var j = 1; j <= column; j++)

{

table[i - 1, j - 1] =

Convert.ToDouble(arrData[i, j]);

Console.Write("\t " + table[i - 1, j - 1]);

}

Console.Write("\n");

}

xlWB.Close(false); // закрываем книгу

xlApp.Quit(); // закрываем Excel

return table;

}

public static void ExportToExcel(string filename, string listname, double sum)

{

// Загрузить Excel, затем создать новую пустую рабочую книгу

Application excelApp = new Application();

// Сделать приложение Excel видимым

excelApp.Visible = true;

Workbook xlWB; //рабочая книга

Worksheet xlSht; //лист Excel

xlWB = excelApp.Workbooks.Open(filename, XlUpdateLinks.xlUpdateLinksNever, false); //название файла Excel

xlSht = (Worksheet)xlWB.Worksheets[listname];

// Установить заголовки столбцов в ячейках

xlSht.Cells[11, "A"] = "Кратчайший путь= ";

xlSht.Cells[11, "B"] = sum + 15;

excelApp.DisplayAlerts = false;

xlSht.SaveAs(string.Format(@"C:\Users\1\Downloads\komivoyazher.xlsx", Environment.CurrentDirectory));

excelApp.Quit();

Debug.WriteLine("запись в файл Excel");

Debug.Assert(false, "Произошла ошибка");

}

}

}