Пензенский государственный университет

Кафедра "Вычислительной техники"

**Отчёт**

По лабораторной №7

по дисциплине "Л и ОА в ИТ"

на тему "Поиск расстояний во взвешенном графе"

***Выполнили студенты группы 19вв1:***

*Григорьева Д. Д.*

*Мереняшева Е. А.*

***Приняли:***

*Митрохин М. А.*

*Юрова О. В.*

Пенза 2020

# Цель работы: выполнить поиск расстояний во взвешенном графе.

**Лабораторное задание:**

### **Задание 1**

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного взвешенного графа *G*. Выведите матрицу на экран.
2. Для сгенерированного графа осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием. При реализации алгоритма в качестве очереди используйте класс **queue** из стандартной библиотеки С++.

3. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для ориентированного взвешенного графа *G*. Выведите матрицу на экран и осуществите процедуру поиска расстояний, реализованную в соответствии с приведенным выше описанием.

### **Задание 2**

1. Модернизируйте программу так, чтобы получить возможность запуска программы с параметрами командной строки. В качестве параметра должны указываться тип графа (взвешенный или нет) и наличие ориентации его ребер (есть ориентация или нет).

**Теоретическая часть:**

Поиск расстояний – довольно распространенная задача анализа графов.

Для поиска расстояний можно использовать процедуры обхода графа. Для этого при каждом переходе в новую вершину необходимо запоминать, сколько шагов до нее мы сделали. При этом вектор, который хранил информацию о посещении вершин становится вектором расстояний. Довольно просто модернизировать для поиска расстояний в графе алгоритм обхода в ширину, т.к. этот алгоритм проходит вершины по уровням удаленности, то для не ориентированного графа для вершин каждого следующего уровня глубины расстояние от исходной вершины увеличивается на 1.

Удалённость в данном случае понимается как количество ребер, по которым необходимо прейти до достижения вершины.

**Практическая часть:**

**Листинг:**

using System;

using System.Collections.Generic;

namespace Program

{

class Program

{

static void BFSD(int v, int[,] matrix, int[] DIST, int size)

{

Queue<int> queue = new Queue<int>(); //Создаем новую очередь

queue.Enqueue(v); //Помещаем v в очередь

DIST[v] = 0;

Console.Write("Результат обхода: ");

while (queue.Count != 0)

{

v = queue.Dequeue();//Удаляем первый элемент из очереди

Console.Write(v);

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

{

if (matrix[v, i] != 0 && DIST[i] == -1)

{

queue.Enqueue(i); //Помещаем i в очередь

DIST[i] = DIST[v] + matrix[v,i];

}

}

}

Console.Write(" ");

}

static void Main(string[] args)

{

Console.Write("Введите размерность матрицы:");

int size = Convert.ToInt32(Console.ReadLine());

int[,] M = new int[size, size];

int[] DIST = new int[size];

Random random = new Random();

Console.WriteLine();

Console.WriteLine("Сгененрированная матрица:\t");

/\*for (int i = 1; i < size; i++)

{

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

{

M[i, j] = random.Next(2); // не взвешенный не направленный

M[j, i] = M[i, j];

}

}\*/

/\*for (int i = 1; i < size; i++)

{

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

{

M[i, j] = random.Next(10); // взвешенный не направленный

M[j, i] = M[i, j];

}

}\*/

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

{

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

{

M[i, j] = random.Next(10); // взвешенный направленный

M[i, i] = 0;

}

}

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

{

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

{

Console.Write($"{M[i, j]}, \t");

}

Console.WriteLine();

}

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

{

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

{

DIST[j] = -1;

}

BFSD(i, M, DIST, size);

Console.Write("Расстояния от вершины "+i+": ");

for (int t = 0; t < size; t++)

{

Console.Write(DIST[t]+" ");

}

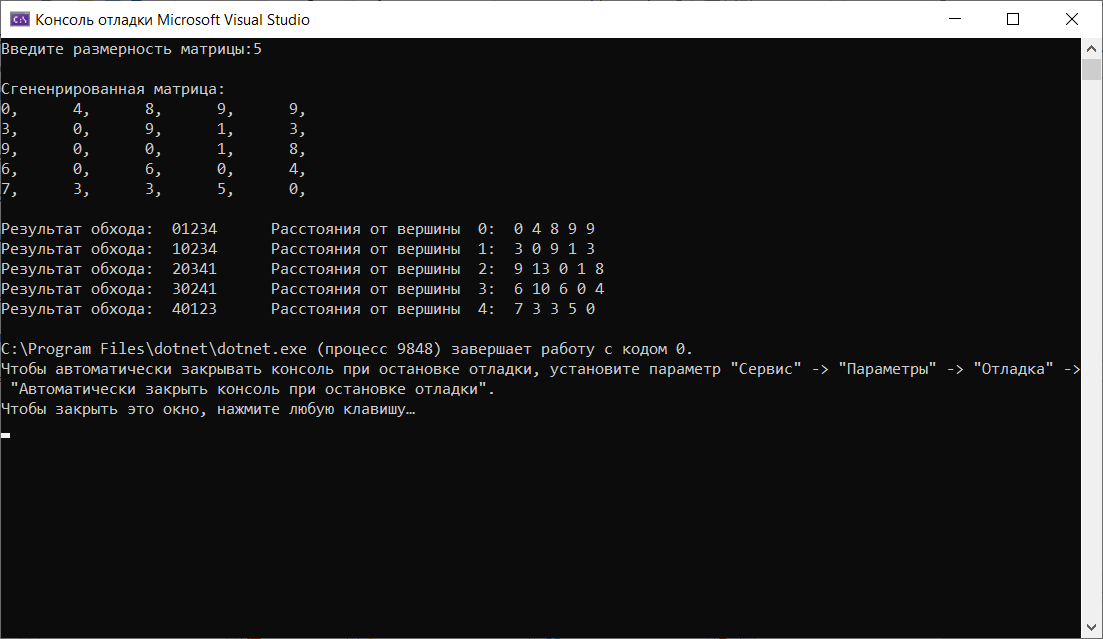
Console.WriteLine();

}

}

}

}

**Результат работы программы:**

**Вывод:** Научились выполнять поиск расстояний во взвешенном графе.