Пензенский государственный университет

Кафедра "Вычислительной техники"

**Отчёт**

По лабораторной №6

по дисциплине "Л и ОА в ИТ"

на тему "Поиск расстояний в 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 в графе.

**Лабораторное задание:**

Задание 1

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного 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++.

**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 lb\_6

{

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.WriteLine();

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] == 1 && DIST[i] == -1)

{

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

DIST[i] = DIST[v] + 1;

}

}

}

}

static void Main(string[] args)

{

Console.Write("Введите вершину для начала обхода:");

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

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

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

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

int[] DIST = new int[size];

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

{

DIST[i] = -1;

}

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);

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

}

}

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

{

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

{

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

}

Console.WriteLine();

}

BFSD(v, M, DIST, size);

Console.WriteLine("\n");

Console.Write("Расстояния: ");

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

{

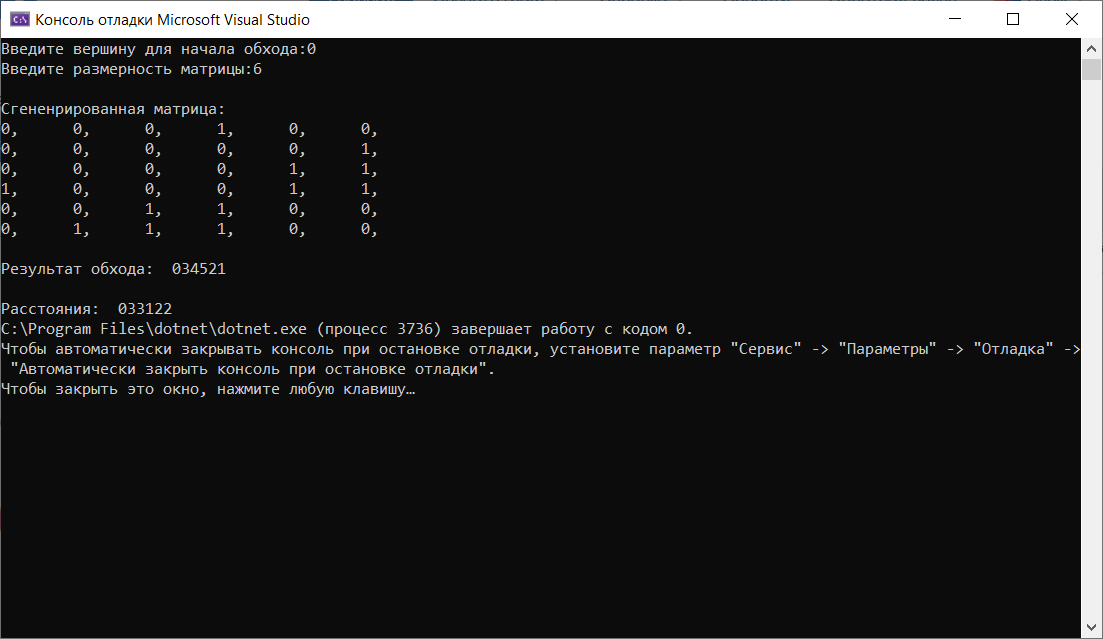
Console.Write(DIST[i]);

}

}

}

}

**Результат работы программы:**

**Вывод:** Научились выполнять поиск расстояний в графе.