Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

ОТЧЕТ

по лабораторной работе №10

по курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Поиск расстояний во взвешенном графе»

Выполнили студенты группы 21вв1.1:

Кузнецов А.С.

Гришин С.В.

Приняли

Юрова О.В.

Акифьев И.В.

Пенза 2022

**Цель:** научиться осуществлять процедуру поиска расстояний во взвешенном на основе матрицы смежности.

**Методические указания:**

**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. Для каждого из вариантов сгенерированных графов (ориентированного и не ориентированного) определите радиус и диаметр.
2. Определите подмножества периферийных и центральных вершин.

**Задание 3\***

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

**Описание метода реализации:**

1)Создали матрицу смежности

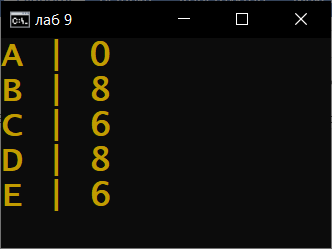
2)Осуществили процедуру поиска расстояния во взвешенном графе

**Результаты работы программы:**

1)Вывели матрицу



2)Осуществили процедуру поиска расстояния



**Вывод:** в ходе выполнения лабораторной работы, мы научились осуществлять процедуру поиска расстояний во взвешенном графе на основе матрицы смежности.

**Листинг:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab\_9

{

internal class Program

{

const int w = 30;

const int h = 7;

struct graf

{

public int[,] matrSM;

}//граф

//ПОИСК \/ \/ \/

static void BFSD(in int[,] matr, int v, ref int[] DIST, Queue<int> Q)

{

DIST[v] = 0;

while (Q.Count != 0)

{

v = Q.Dequeue();

for (int i = 1; i < DIST.Length; i++)

{

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

{

Q.Enqueue(i);

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

}

}

}

}

//ГЕНЕРАЦИЯ МАТРИЦЫ СМЕЖНОСТИ \/ \/ \/

static void matrSMgen(graf gr, int rand\_key)

{

Random rand = new Random(rand\_key);

int ras = Convert.ToInt32(Math.Sqrt(gr.matrSM.Length));

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

{

gr.matrSM[i, 0] = i;

gr.matrSM[0, i] = i;

}

gr.matrSM[0, 0] = -29;

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

{

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

{

gr.matrSM[i, j] = 1 + rand.Next(Convert.ToInt32(ras \* 1.5));

if (gr.matrSM[i, j] > 9) { gr.matrSM[i, j] = 0; }

gr.matrSM[j, i] = gr.matrSM[i, j];

}

}

}

//ВЫВОД МАТРИЦЫ СМЕЖНОСТИ \/ \/ \/

static void matrSMprint(graf gr)

{

if (Math.Sqrt(gr.matrSM.Length) < 5)

{

Console.WindowWidth = 11;

Console.WindowHeight = 11;

Console.SetBufferSize(Console.WindowWidth, Console.WindowHeight);

}

else

{

Console.WindowWidth = Convert.ToInt32(Math.Sqrt(gr.matrSM.Length)) \* 2 + 1;

Console.WindowHeight = Convert.ToInt32(Math.Sqrt(gr.matrSM.Length)) + 1;

Console.SetBufferSize(Console.WindowWidth, Console.WindowHeight);

}

int ras = Convert.ToInt32(Math.Sqrt(gr.matrSM.Length));

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

{

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

{

if (i == 0 || j == 0)

{

Console.Write(" " + Convert.ToChar(gr.matrSM[i, j] + 64));

}

else

{

if (true) { if (gr.matrSM[i, j] == 0) { sc(2); } else { sc(1); } }//ЦВЕТНАЯ ТАБЛИЦА

Console.Write(" " + gr.matrSM[i, j]);

sc(0);

}

}

Console.WriteLine();

}

}

//ИЗМЕНЕНИЕ ЦВЕТА КОНСОЛИ \/ \/ \/ (0 - жёлтый; 1 - зелённый; 2 - красный)

static void sc(byte mod)

{

switch (mod)

{

case 0:

Console.ForegroundColor = ConsoleColor.DarkYellow;

break;

case 1:

Console.ForegroundColor = ConsoleColor.DarkGreen;

break;

case 2:

Console.ForegroundColor = ConsoleColor.DarkRed;

break;

}

}

//ИЗМЕНЕНИЕ РАЗМЕРА КОНСОЛИ \/ \/ \/

static void consize(int weigh, int hight)

{

Console.SetWindowSize(weigh - 1, hight - 1);

Console.SetBufferSize(Console.WindowWidth, Console.WindowHeight);

Console.SetWindowSize(weigh, hight);

Console.SetBufferSize(Console.WindowWidth, Console.WindowHeight);

}

static void Main(string[] args)

{

Console.Title = "лаб 9";

sc(0);

Console.CursorVisible = false;

int mod = 0;

graf A = new graf();

consize(w, h);

MenuMain:

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" 1) Сгенерировать графы");

Console.WriteLine(" 2) Вывести матрицы графов");

Console.WriteLine(" 3) поиск расстояния");

Console.Write("\n■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

mod = Convert.ToInt32(Console.ReadKey().KeyChar);

if ((mod < 49 || mod > 54) && mod != 32)

{

Console.Clear();

consize(w, h);

goto MenuMain;

}

switch (mod)

{

case 49://СОЗДАНИЕ ГРАФА

{

int ras;

Console.Clear();

consize(w, 8);

ERROR1:

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.WriteLine(" Введите размер графа");

Console.WriteLine(" (от 0 до 14)\n\n\n");

Console.Write("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

Console.SetCursorPosition(15, Console.WindowHeight - 3);

if (int.TryParse(Console.ReadLine(), out ras) == false)

{

Console.Clear();

consize(w, 12);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(2);

Console.WriteLine(" введено некоректное число\n");

sc(0);

goto ERROR1;

}

if (ras < 0)

{

Console.Clear();

consize(w, 13);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(2);

Console.WriteLine(" Размер матрицы не может");

Console.WriteLine(" быть отрицательным!\n");

sc(0);

goto ERROR1;

}

if (ras > 14)

{

Console.Clear();

consize(w, 12);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(2);

Console.WriteLine(" превышен максимальный размер\n");

sc(0);

goto ERROR1;

}

ras++;

A.matrSM = new int[ras, ras];

matrSMgen(A, ras);

Console.Clear();

consize(w, h + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(1);

Console.WriteLine(" !граф успешно создан!\n");

sc(0);

goto MenuMain;

}

case 50://ВЫВОД ГРАФОВ

{

if (A.matrSM == null)

{

Console.Clear();

consize(w, h + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(2);

Console.WriteLine(" граф не сгенерирован\n");

sc(0);

goto MenuMain;

}

Console.Clear();

graf buf = A;

matrSMprint(buf);

Console.ReadKey();

Console.Clear();

consize(w, h);

goto MenuMain;

}

case 51://ПОИСК

{

if (A.matrSM == null)

{

Console.Clear();

consize(w, h + 4);

Console.WriteLine("■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■");

sc(2);

Console.WriteLine(" граф не сгенерирован\n");

sc(0);

goto MenuMain;

}

Console.Clear();

consize(15, Convert.ToInt32(Math.Sqrt(A.matrSM.Length)));

int[] DIST = new int[Convert.ToInt32(Math.Sqrt(A.matrSM.Length)) - 1];

for (int i = 0; i < DIST.Length; i++)

{

DIST[i] = -1;

}

Queue<int> Q = new Queue<int>();

Q.Enqueue(0);

BFSD(A.matrSM, 0, ref DIST, Q);

Console.WriteLine();

for (int i = 0; i < DIST.Length; i++)

{

Console.WriteLine(Convert.ToChar(A.matrSM[0, i + 1] + 64) + " | " + DIST[i]);

}

Console.ReadKey();

Console.Clear();

consize(w, h);

goto MenuMain;

}

}

}

}

}