Вариант №16

Нигамадянов Ф.М.

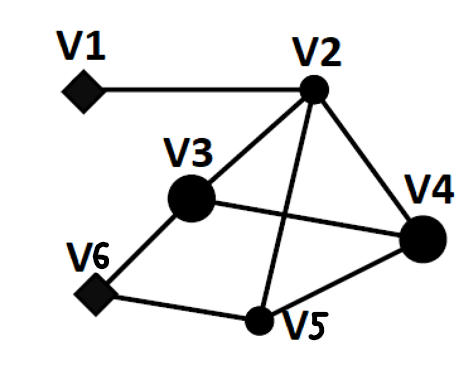
гр. 4210

Дата сдачи: 28.12.2021

Расчётно-графическая работа по дисциплине «Дискретная математика»

Задание №1

На некотором языке программирования разработать программу, определяющую с помощью матриц смежности в графе, задаваемом пользователем, расстояние и количество маршрутов интересующей длины между некоторой парой вершин. Привести листинг программы с комментариями. Продемонстрировать работу программы на приведённом (*p*, *q*)-графе ( p = |*V*|, *q* = |*X*| ) – привести входные и соответствующие им выходные данные по определению расстояния между вершинами ◆ и количеству маршрутов длины (*q* – p + 3) между вершинами ⚫.



Решение:  
  
using System;

namespace CGT\_discrete\_math\_CS

{

class WorkingWithGraph

{

static int num\_of\_vertexes; // Вершины графа, количество которых указывает пользователь

static int[][] original\_matrix; // Исходная матрица, вводимая пользователем

static int[][] resulting\_matrix; // Результирующая матрица, которая получается при возведении матрицы в степень

static string str; // Строка для сохранения введенных пользователем данных

static int num\_of\_edges = 0; // Количество ребер нашего графа

//

// Ниже приведена функция, которая умножает переданную матрицу на исходную матрицу

//

static int[][] multiply\_to\_original(int[][] matrix)

{

int sum = 0; // Поле для хранения суммы перемножения элементов строки на элементы столбца

int[][] answer = new int[num\_of\_vertexes][]; // Матрица, которая будет хранить рез-т умножения матриц

for (int i = 0; i < num\_of\_vertexes; i++) // Процесс умножения матриц

{

answer[i] = new int[num\_of\_vertexes]; // Выделение памяти для первой строки

for (int iter = 0; iter < num\_of\_vertexes; iter++) // Этот цикл нужен для переключения столбца

{

for (int j = 0; j < num\_of\_vertexes; j++)

{

sum += matrix[i][j] \* original\_matrix[j][iter]; // Суммируем соответствующие произведения эле-ов

}

answer[i][iter] = sum; // Соотв-му элементу приравниваем полученную сумму

sum = 0; // Обнуляем сумму

}

}

return answer; // Возврат результата умножения матриц

}

//

// Ниже приведена функция, которая ищет путь кратчайшей длины между двумя вершинами

//

static void min\_lehgth\_between\_vertexes(int Vi, int Vj)

{

int counter = 1; // Счетчик для отслеживания степени, в которую возведена матрица

if (resulting\_matrix[Vi - 1][Vj - 1] != 0) // если интерес-ий нас элемент не равен нулю

{

Console.WriteLine($"Кратчайшее растояние между верщинами {Vi} и {Vj} равно {counter}");

}

// Если же не равен нулю, то возводим матрицу в соотв. степень,

// пока инте-ий нас элемент не станет отличным от нуля

// Если элемент стал отличным от нуля, то это наш результат

else

{

while (resulting\_matrix[Vi - 1][Vj - 1] == 0)

{

resulting\_matrix = multiply\_to\_original(resulting\_matrix);

counter++;

}

Console.WriteLine($"Кратчайшее растояние между верщинами {Vi} и {Vj} равно {counter}");

resulting\_matrix = original\_matrix; // Возвращаешм результирующую матрицу в исходное состояние

}

}

//

// Ниже приведена функция, ищущая количество путей между интересующими нас вершинами с определенной длиной

//

static void num\_of\_def\_lehgth\_between\_vertexes(int a, int b)

{

int counter = 1; // Счетчик для отслеживания степени, в которую возведена матрица

int def\_length = num\_of\_edges - num\_of\_vertexes + 3; // Интресующая нас длина. См. исходный граф

if (counter == def\_length) // Если степень, в которую возведена матрица, равна интере-ей нас длине

{

Console.WriteLine($"Количество путей длины {def\_length} между верщинами {a} и {b} равно {resulting\_matrix[a - 1][b - 1]}");

}

else // если не равна, то возводим матрицу в степень, пока она не станет равна интере-ей нас длине

{

while (counter != def\_length)

{

resulting\_matrix = multiply\_to\_original(resulting\_matrix);

counter++;

}

Console.WriteLine($"Количество путей длины {def\_length} между верщинами {a} и {b} равно {resulting\_matrix[a - 1][b - 1]}");

resulting\_matrix = original\_matrix;

}

}

//

// Точка входа в программу. Функция Main

static void Main(string[] args)

{

Console.WriteLine("Введите количество вершин в графе!"); // Просим ввести ко-во вершин

num\_of\_vertexes = int.Parse(Console.ReadLine()); // Считываем ко-во вершин

original\_matrix = new int[num\_of\_vertexes][]; // Выделяем память под исходную матрицу

resulting\_matrix = new int[num\_of\_vertexes][]; // Выделяем память под результирующую матрицу

Console.WriteLine("Введите матрицу смежности");

for (int i = 0; i < num\_of\_vertexes; i++) // Построчно вводим матрцу

{

str = Console.ReadLine(); // Считываем строку

string[] mas\_str = str.Split(" "); // Отправляем элементы в массив с разделением по пробелу

original\_matrix[i] = new int[num\_of\_vertexes]; // Выделяем память под первую строку исходной матрицы

resulting\_matrix[i] = new int[num\_of\_vertexes]; // Выделяем память под первую строку рез-ей матрицы

for (int j = 0; j < num\_of\_vertexes; j++) // Присваиваем одному из массивов массива введенную нами строку

{

original\_matrix[i][j] = int.Parse(mas\_str[j]);

num\_of\_edges += original\_matrix[i][j]; // Добавляем ребра

}

resulting\_matrix[i] = original\_matrix[i]; // Результирующая матрица изначальна равна исходной

}

num\_of\_edges /= 2; // Делим ко-во ребер пополам, т.к. каждое ребро мы учли два раза

Console.WriteLine("Введите номера вершин, между которыми нужно найти путь минимальной длины");

str = Console.ReadLine(); // Считываем строку

string[] mas\_vertx = str.Split(" "); // Отправляем элементы в массив с разделением по пробелу

min\_lehgth\_between\_vertexes(int.Parse(mas\_vertx[0]), int.Parse(mas\_vertx[1])); // Вызываем функ-ю поиска пути минимальной длины между вершинами

Console.WriteLine("Введите номера вершин таких, между которыми нужно найти количество путей заданной длины");

str = Console.ReadLine(); // Считываем строку

mas\_vertx = str.Split(" "); // Отправляем элементы в массив с разделением по пробелу

num\_of\_def\_lehgth\_between\_vertexes(int.Parse(mas\_vertx[0]), int.Parse(mas\_vertx[1])); // Вызываем функ-ю поиска количества путей определенной длины

}

}

}

