|  |
| --- |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА – Российский технологический университет"**  **РТУ МИРЭА** |

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-3 «Управление и моделирование систем»

**ОТЧЕТ   
о выполнении лабораторной работы №2**

**«Алгоритмы на графах»**

**по дисциплине   
«Алгоритмы и структуры данных»**

**Вариант № 98**

Выполнил: студент 2 курса

группы БИСО-03-19

шифр 19Б1398

Тепсикоев С.А

Проверил:

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Москва 2020 г.

**Задание на лабораторную работу № 2.**

В рамках лабораторной работы №2 требуется реализовать в виде программы абстрактный тип данных «Граф» согласно варианту (Номер варианта – две последние цифры шифра студента, номера зачетной книжки) с учетом заданного представления графа. Операторы (операции) АТД «Граф» функционально должны выполнять следующие операции (названия операций – примерные):

1. FIRST(v) - возвращает индекс первой вершины, смежной с вершиной v. Если вершина v не имеет смежных вершин, то возвращается "нулевая" вершина.

2. NEXT(v, i)- возвращает индекс вершины, смежной с вершиной v, следующий за индексом i. Если i — это индекс последней вершины, смежной с вершиной v, то возвращается.

3. VERTEX(v, i) - возвращает вершину с индексом i из множества вершин, смежных с v.

4. ADD\_V(<имя>,<метка, mark>) - добавить УЗЕЛ.

5. ADD\_Е(v, w, c) - добавить ДУГУ (здесь c — вес, цена дуги (v,w))

6. DEL\_V(<имя>) - удалить УЗЕЛ.

7. DEL\_Е(v, w) – удалить ДУГУ.

8. EDIT\_V(<имя>, <новое значение метки или маркировки>) - изменить метку (маркировку) УЗЛА.

9.EDIT\_Е(v, w, <новый вес дуги>) - изменить вес ДУГИ.

Реализовать задание (заданный алгоритм) с использованием методов АТД «Граф». Оформление отчета не менее чем с двумя контрольными примерами, для каждого примера приводится рисунок (допускается скан рисунка «от руки» или изображение построенное c помощью графического или специализированного редактора) графа.

**Вариант № 98.**

**Алгоритм: Определить минимальное число красок, которыми можно раскрасить граф и вывести пример такой раскраски.**

**Способ представления графа: Матрица смежности**

**Теория об алгоритме**

Пусть

*G* = (*V, E*) –обыкновенный граф,

*k –натуральное число.*

Функция

*f*: V{1,…,*k*} называется *раскраской* графа.

Раскраска называется ***правильной***, если для любых смежных вершин x, yV справедливо неравенство *f*(x)*f*(y).

Число –число красок раскраски .

Правильность раскраски означает, что *смежные вершины раскрашиваются в разные цвета*.

Наименьшее число красок, необходимое для правильной раскраски графа G называется хроматическим числом графа G.

**Алгоритм раскраски графа**

Алгоритм состоит в следующем.

1.Упорядочиваем вершины графа

2.Вершину красим первой краской.

3.Предположим, что вершины ,…, уже раскрашены и на это использовано *m* красок. Если на раскрашенные вершины, смежные с + 1, использованы все краски, то + 1 раскрашиваем *m* + 1 краской. Если среди *m* красок найдется краска, которая не использована для вершин, смежных с + 1, то вершину + 1 красим этой краской.

**Листинг программы**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Graph

{

class Vertex

{

public int Number { get; set; }

public string Name { get; set; }

public Vertex(int number, string name )

{

Number = number;

Name = name;

}

}

class Edge

{

public Vertex From { get; set; }

public Vertex To { get; set; }

public int Weight { get; set; }

public Edge (Vertex from, Vertex to, int weight = 1)

{

From = from;

To = to;

Weight = weight;

}

}

class Graph

{

List<Vertex> Vertexes = new List<Vertex>();

List<Edge> Edges = new List<Edge>();

int[,] matrix;

int[] color;

public int VertexCount => Vertexes.Count;

public int EdgeCount => Edges.Count;

public void AddVertex(Vertex vertex)

{

Vertexes.Add(vertex);

}

public void AddEdge(Vertex from,Vertex to)

{

var edge = new Edge(from, to);

Edges.Add(edge);

}

public int [,] GetMatrix()

{

matrix = new int[VertexCount, VertexCount];

foreach (var edge in Edges)

{

var row = edge.From.Number - 1;

var column = edge.To.Number - 1;

matrix[row, column] = edge.Weight;

matrix[column, row] = matrix[row, column];

}

return matrix;

}

public int indexOfName(string NameSearch)

{

int c = 0;

foreach(var Vertex in Vertexes)

{

if(Vertex.Name == NameSearch)

return c;

c++;

}

return -1;

}

public int First(string v)

{

int[,] matrix = GetMatrix();

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

if (matrix[indexOfName(v), i] == 1) return i+1;

return -1;

}

public int Next(string v,int iFrom)

{

int[,] matrix = GetMatrix();

for (int i = iFrom; i < VertexCount; i++)

if (matrix[indexOfName(v), i] == 1) return i;

return -1;

}

public int Vertex(int ii)

{

var matrix = GetMatrix();

for (int i = ii; i < VertexCount; i++)

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

if (matrix[i, j] != 0)

return j+1;

return -1;

}

public void DelVertex(Vertex name)

{

Vertexes.Remove(name);

}

public void DelEdge(Vertex from, Vertex to)

{

foreach (var edge in Edges)

if(edge.From == from && edge.To == to)

Edges.Remove(edge);

}

public void EditVertex(string name, string newName)

{

foreach (var v in Vertexes)

if (v.Name == name)

v.Name = newName;

}

public void EditEdge(Vertex from, Vertex to, int newWeight)

{

foreach (var edge in Edges)

if (edge.From == from && edge.To == to)

edge.Weight = newWeight;

}

public bool isSafe(int v, int[,] matrix, int[] color, int c)

{

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

if (matrix[v, i] == 1 && c == color[i])

return false;

return true;

}

// Рекурсивная функция раскраски m цветами

public bool graphColoringUtil(int[,] matrix, int m, int[] color, int v)

{

//частный случай: если всем вершинам присвоен цвет, то вернуть true

if (v == VertexCount)

return true;

// Рассмотрим текущую вершину v и попробуем разные цвета

for (int c = 1; c <= m; c++)

{

// Проверка, правильно ли присвоен цвета c -> v

if (isSafe(v, matrix, color, c))

{

color[v] = c;

// повторение, для назначения цвета остальным вершинам

if (graphColoringUtil(matrix, m, color, v + 1))

return true;

// Если назначение цвета с не приводит к решению проблемы, то удалите его

color[v] = 0;

}

}

// Если никакой цвет не может быть назначен этой вершине, то вернуть false

return false;

}

/\* Функция для m -раскраски. Для решения используется graphColoringUtil().

\* Функция возвращает false, если m цветов не могут быть назначены,

\* в противном случае возвращает true и печатает назначения цветов всем вершинам.\*/

public bool graphColoring(int[,] matrix, int m)

{

// Инициализируйте все значения цвета как 0

color = new int[VertexCount];

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

color[i] = 0;

// Вызов graphColoringUtil() для 0 вершины

if (!graphColoringUtil(matrix, m, color, 0))

{

Console.WriteLine("Solution does not exist");

return false;

}

// Вывод результата

printSolution(color);

return true;

}

// Функция для печати цветов

public void printSolution(int[] color)

{

int c = 0;

foreach(var i in Vertexes)

{

Console.WriteLine($"Вершина {i.Name} окрашена в {color[c]} цвет");

c++;

}

Console.WriteLine();

Console.WriteLine($"Минимальное число красок, которыми можно раскрасить граф равно: {color.Max()}");

}

public void Print()

{

var matrix = new int[Vertexes.Count, Vertexes.Count];

foreach (var edge in Edges)

{

var row = edge.From.Number - 1;

var column = edge.To.Number - 1;

matrix[row, column] = edge.Weight;

matrix[column, row] = matrix[row, column];

}

Console.WriteLine("Матрица смежности");

Console.WriteLine();

Console.Write(" ");

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

{

Console.Write(i + 1 + " ");

}

Console.WriteLine();

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

{

Console.Write(" " + (i + 1) + " ");

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

{

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

}

Console.WriteLine();

}

Console.WriteLine();

}

}

class Program

{

static void Main(string[] args)

{

{

Console.WriteLine(" Граф 1");

Console.WriteLine();

var graph = new Graph();

var a = new Vertex(1, "a");

var b = new Vertex(2, "b");

var c = new Vertex(3, "c");

var d = new Vertex(4, "d");

var e = new Vertex(5, "e");

//var f = new Vertex(6,"f");

graph.AddVertex(a);

graph.AddVertex(b);

graph.AddVertex(c);

graph.AddVertex(d);

graph.AddVertex(e);

//graph.AddVertex(f);

graph.AddEdge(a, b);

graph.AddEdge(a, d);

graph.AddEdge(b, c);

graph.AddEdge(c, a);

graph.AddEdge(c, e);

graph.AddEdge(d, a);

graph.AddEdge(d, b);

graph.AddEdge(d, c);

graph.AddEdge(d, e);

graph.AddEdge(e, a);

graph.Print();

int m = graph.VertexCount; // Number of colors

graph.graphColoring(graph.GetMatrix(), m);

Console.WriteLine();

Console.WriteLine();

}//1 граф

{

Console.WriteLine(" Граф 2");

Console.WriteLine();

var graph = new Graph();

var a = new Vertex(1, "a");

var b = new Vertex(2, "b");

var c = new Vertex(3, "c");

var d = new Vertex(4, "d");

var e = new Vertex(5, "e");

graph.AddVertex(a);

graph.AddVertex(b);

graph.AddVertex(c);

graph.AddVertex(d);

graph.AddVertex(e);

graph.AddEdge(a, b);

graph.AddEdge(a, c);

graph.AddEdge(a, d);

graph.AddEdge(b, c);

graph.AddEdge(b, e);

graph.AddEdge(c, d);

graph.AddEdge(d, e);

graph.Print();

int m = graph.VertexCount; // Number of colors

graph.graphColoring(graph.GetMatrix(), m);

}//2 граф

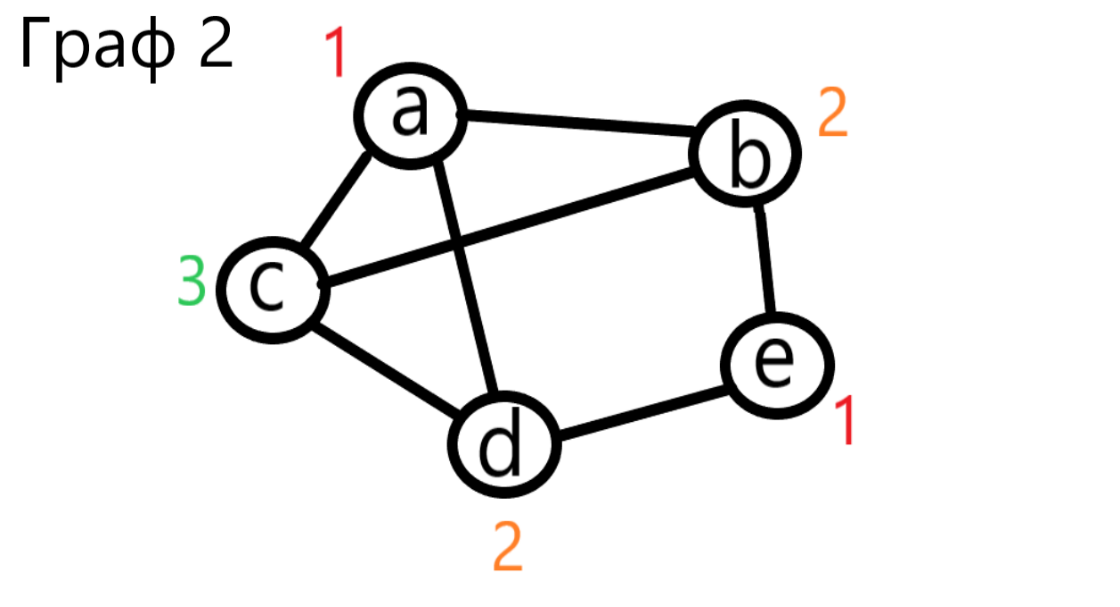
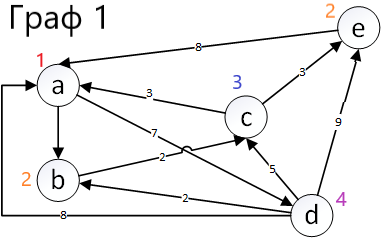
Console.ReadLine();

}

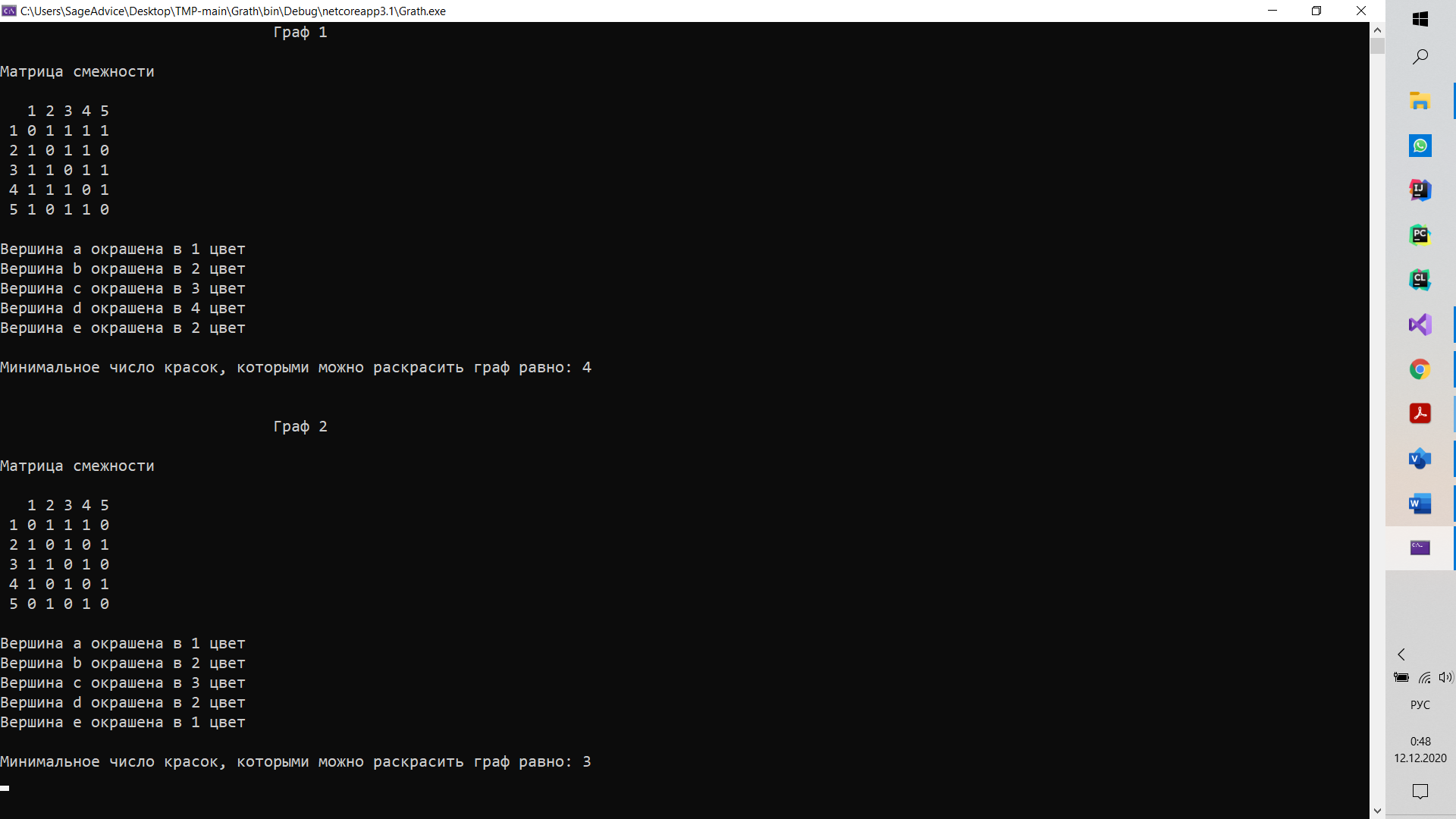
}

}

**Контрольные примеры**

****

**Скриншот работы программы:**



**Вывод:**

**Источники:**

https://www.geeksforgeeks.org/m-coloring-problem-backtracking-5/?ref=rp