Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное учреждение высшего образования  
«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

Дискретная математика

Семестр 4

Лабораторная работа 3

Выполнил  
студент группы РИС-22-2б  
Баяндин К. С.

Проверила  
старший преподаватель кафедры ИТАС  
Рустамханова Г.И.

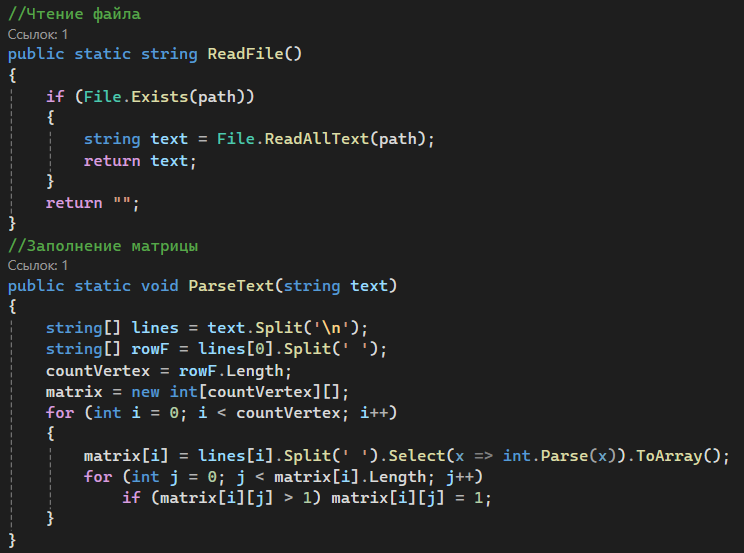
Пермь, 2024

Постановка задачи

Вводится матрица смежности из файла. Требуется найти минимальную раскраску графа

Алгоритмы работы программы

Для считывания матрицы смежности из файла в двумерную матрицу разработан следующий алгоритм:



*Рисунок 1 – считывание из файла*

В программе разработан алгоритм минимальной раскраски графа:

1. Сначала создается массив colors, который будет содержать цвета для каждой вершины графа. Размер этого массива равен количеству вершин в графе.

2. Затем создается переменная maxColor, которая будет содержать текущий максимальный использованный цвет.

3. Затем происходит цикл по всем вершинам графа. Внутри этого цикла происходит следующее:

- Определяется минимальный неиспользованный цвет для текущей вершины. Это происходит внутренним циклом, который проверяет все цвета, использованные для смежных вершин.

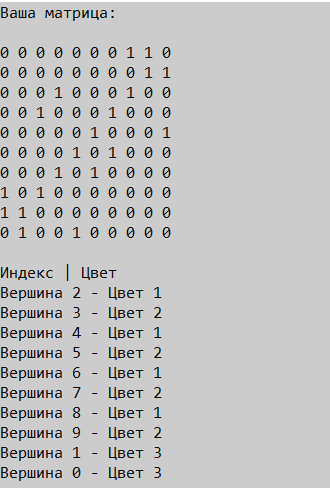
- Если найденный цвет не использовался для смежных вершин, то он присваивается текущей вершине.

- Также обновляется значение maxColor, если текущий найденный цвет больше предыдущего maxColor.

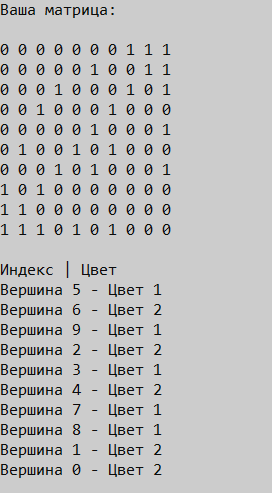
4. После завершения цикла возвращается массив colors, который содержит цвета для каждой вершины графа.

В результате работы программы выводится корректная раскраска графа.

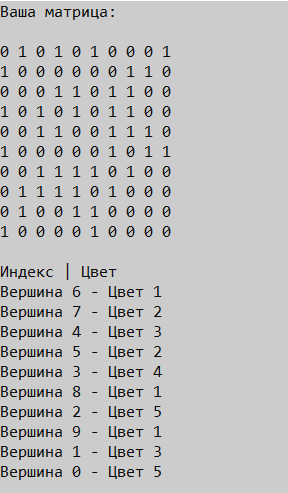
**Результаты работы программы:**

****

*Рисунок 2 – 1 текстовый файл*



*Рисунок 3 – 2 текстовый файл*



*Рисунок 4 – 3 текстовый файл*

**Листинг программы:**

**Program.cs**

using coloring\_graphs;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace coloring\_graphs

{

public class Program

{

readonly static string path = @"C:\Users\bayan\source\repos\Discrete-mathematics-2-course\coloring\_graphs\g33.txt";

static int countVertex;

static int[][]? matrix;

static List<Vertex> masVertex = new();

public static void Main()

{

//считывание матрицы

string text = ReadFile();

ParseText(text);

Console.WriteLine("Ваша матрица:\n");

PrintMatrix();

//Парсинг вершин

MatrixToList();

//Раскраска

Coloring();

//Вывод результата

Console.WriteLine();

Console.WriteLine("Индекс | Цвет ");

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

{

Console.WriteLine($"Вершина {masVertex[i].Number} - Цвет {masVertex[i].Color}");

}

}

//Проверка на соседей определённого цвета color

public static bool NotColorNeighbors(int color, List<int> neighbors, int countVertex, List<Vertex> masVertex)

{

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

if (neighbors.Contains(masVertex[j].Number) && masVertex[j].Color == color)

return false;

return true;

}

//Печать матрицы

public static void PrintMatrix()

{

if (matrix != null)

{

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

{

for (int j = 0; j < matrix.Length; j++)

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

Console.WriteLine();

}

}

}

//Выбор нераскрашенного элемента с максимальным количеством связей

public static int MaxRowNumber()

{

int maxRow = 0;

if (matrix != null)

{

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

if (matrix[i].Sum() > maxRow)

maxRow = i;

}

return maxRow;

}

//Чтение файла

public static string ReadFile()

{

if (File.Exists(path))

{

string text = File.ReadAllText(path);

return text;

}

return "";

}

//Заполнение матрицы

public static void ParseText(string text)

{

string[] lines = text.Split('\n');

string[] rowF = lines[0].Split(' ');

countVertex = rowF.Length;

matrix = new int[countVertex][];

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

{

matrix[i] = lines[i].Split(' ').Select(x => int.Parse(x)).ToArray();

for (int j = 0; j < matrix[i].Length; j++)

if (matrix[i][j] > 1) matrix[i][j] = 1;

}

}

//выбор вершин из матрицы

public static void MatrixToList()

{

if (matrix != null)

{

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

{

int numerRow = MaxRowNumber();

List<int> neighbors = new();

for (int j = 0; j < matrix[numerRow].Length; j++)

if (matrix[numerRow][j] == 1)

{

neighbors.Add(j);

matrix[numerRow][j] = 0;

}

masVertex.Add(new Vertex(numerRow, neighbors));

}

}

}

//Раскраска

public static void Coloring()

{

int useColor = 1;

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

{

if (masVertex[i].Color == 0)

{

masVertex[i].Color = useColor;

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

if (masVertex[j].Color == 0 && NotColorNeighbors(useColor, masVertex[j].Neighbors, countVertex, masVertex))

{

masVertex[j].Color = useColor;

}

useColor++;

}

}

}

}

}

**Vertex.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace coloring\_graphs

{

public class Vertex

{

public int Number { get; set; }

public List<int> Neighbors { get; set; }

public int Color { get; set; }

public Vertex(int number, List<int> neighbors)

{

Number = number;

Neighbors = neighbors;

Color = 0;

}

}

}