Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
|  |
| Кафедра «Информатика» |
| кафедра |

**ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ**

|  |
| --- |
| Раскраска графов |
| Тема |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  |  |  | Р. Ю. Царев |
|  | |  |  |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ19-17/1б 031939175 | | |  |  |  | А. Д. Непомнящий |
|  | номер группы, зачетной книжки | | |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2021

1. Цель работы

Изучение жадного алгоритма раскраски графа.

1. Задачи

Написать программу, реализующую жадный алгоритм раскраски графа.

Предъявлены следующие требования к выполнению работы.

1. Строгое соответствие программы и результатов ее работы с полученным заданием.
2. Самостоятельные тестирование и отладка программы.
3. Предоставление демонстрационного примера и исходного текста программы для защиты.
4. Предоставление отчета по практическому заданию, содержащего описание реализованного алгоритма, программы, результатов работы программы (отчет необходимо загрузить на сайт курса).
5. Описание реализованного алгоритма

Реализован жадный алгоритм раскраски графа.

1. Описание программы

Для решения задачи была написана программа на языке C#. Было создано четыре класса. Vertex – класс, необходимый для хранения информации о вершине графа (имя вершины), Edge – класс, необходимый для хранения информации о ребре графа (вершины, соединяемые ребром), Graph – хранит коллекции ребер, вершин, класс VertexComparer, реализующий компаратор для вершин графа. В классе Graph описан метод GreedyColor(), реализующий алгоритм раскраски и возвращающий словарь вершина-цвет. Класс Program необходим для демонстрации примера работы.

Листинг 1 – Код в файле Vertex.cs

namespace Lab6

{

public class Vertex

{

public string Name { get; }

public Vertex(string name)

{

Name = name;

}

}

}

Листинг 2 – Код в файле Edge.cs

namespace Lab6

{

public class Edge

{

public Vertex Tail { get; }

public Vertex Head { get; }

public Edge(Vertex tail, Vertex head)

{

Tail = tail;

Head = head;

}

}

}

Листинг 3 – Код в файле Graph.cs

using System;

using System.Collections.Generic;

using System.Linq;

namespace Lab6

{

public class Graph

{

private List<Vertex> Vertices { get; }

Продолжение листинга 3

private List<Edge> Edges { get; }

private class VertexComparer : IComparer<Vertex>

{

public int Compare(Vertex x, Vertex y)

{

return string.Compare(x.Name, y.Name);

}

}

public Graph(bool[,] adjacencyMatrix)

{

Vertices = new List<Vertex>();

Edges = new List<Edge>();

for (var i = adjacencyMatrix.GetLowerBound(0);

i <= adjacencyMatrix.GetUpperBound(0);

i++)

{

Vertices.Add(new Vertex((i+1).ToString()));

}

CreateEdges(adjacencyMatrix);

}

private void CreateEdges(bool[,] adjacencyMatrix)

{

for (var i = 0; i < Vertices.Count; i++)

{

for (var j = 0; j < Vertices.Count; j++)

{

if (i == j || !adjacencyMatrix[i, j]) continue;

Edges.Add(new Edge(Vertices[i], Vertices[j]));

}

}

}

private List<Vertex> Neighbours(Vertex vertex)

{

return (from edge in Edges where edge.Tail == vertex select edge.Head).ToList();

Окончание листинга 3

}

public Dictionary<Vertex, int> GreedyColor()

{

Vertices.Sort(new VertexComparer());

Dictionary<Vertex, int> colors = new();

foreach (var vertex in Vertices)

{

List<int> neighbourColorUsage = new();

for (var i = 0; Vertices[i] != vertex; i++)

{

if (!Neighbours(vertex).Contains(Vertices[i]))

{

continue;

}

if (!colors.ContainsKey(Vertices[i]))

{

continue;

}

neighbourColorUsage.Add(colors[Vertices[i]]);

}

var newColor = 1;

while (neighbourColorUsage.Contains(newColor))

{

newColor++;

}

colors[vertex] = newColor;

neighbourColorUsage.Clear();

}

return colors;

}

}

}

Листинг 4 – Код в файле Program.cs

using System;

namespace Lab6

{

internal static class Program

{

private static void Main(string[] args)

{

bool[,] lengthMatrix =

{

{false,true,false,true,false,false,false},

{true,false,true,true,true,false,false},

{false,true,false,false,true,false,false},

{true,true,false,false,true,true,false},

{false,true,true,true,false,true,true},

{false,false,false,true,true,false,true},

{false,false,false,false,true,true,false}

};

var graph = new Graph(lengthMatrix); // Create Graph

var result = graph.GreedyColor();

foreach (var (key, value) in result)

{

Console.WriteLine($"Вершина {key.Name} - цвет {value}");

}

}

}

}

1. Результаты работы программы

На рисунке 1 приведен скриншот с результатами работы программы. На рисунке 2 приведен граф, к которому был применен алгоритм раскраски. Вершины графа раскрашены в соответствие с результатом работы программы.

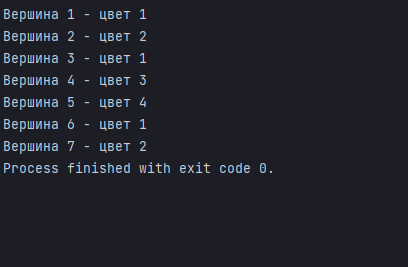


Рисунок 1 – Результаты работы программы

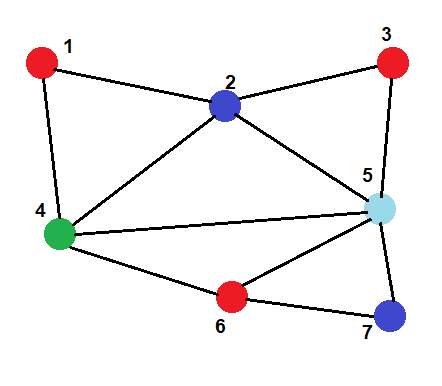


Рисунок 2 – Граф, к которому был применен алгоритм раскраски