Министерство науки и высшего образования Российской Федерации

Федеральное государственное образовательное учреждение

высшего образования

**ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра систем автоматизации управления

**Дисциплина: Алгоритмы и структуры данных**

**Отчёт по лабораторной работе №10**

**Тема: «Обходы графов. Поиск точек сочленения и мостов в графе. Построение паросочетания в графе»**

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил студент  группы ИТб 2302-02-20 |  | Ердяков Роман Александрович |  |
|  |  |
| Проверила |  | Кашина Елена Вячеславовна |  |

СОДЕРЖАНИЕ

[1 Задание 3](#_Toc200883455)

[2 Тестирование (задача 1) 4](#_Toc200883456)

[3 Программа на C# (задача 1): 5](#_Toc200883457)

[4 Тестирование (задача 2) 6](#_Toc200883458)

[5 Программа на C# (задача 2): 7](#_Toc200883459)

[6 Тестирование (задача 3) 9](#_Toc200883460)

[7 Программа на C# (задача 3): 10](#_Toc200883461)

[8 Тестирование (задача 4) 13](#_Toc200883462)

[9 Программа на C# (задача 4): 14](#_Toc200883463)

[10 Вывод 16](#_Toc200883464)

# Задание

Задача 1.

Написать программу, которая с помощью обхода в глубину определяет число компонент связности.

Задача 2.

Написать программу, которая с помощью обхода в ширину определяет наличие циклов в орграфе.

Задача 3.

Написать программу для определения точек сочленения и мостов в графе.

Задача 4.

Написать программу, которая реализует метод построения (нахождения) паросочетания в графе. Использовать метод Куна для избыточных моделей (построения наибольшего паросочетания) и серию обходов в ширину для двудольных графов. Изобразить полученный граф на листочке, выписать доли графа и полученное паросочетание.

# Тестирование (задача 1)

Примеры выполнения программы представлены на рисунке 1 и 2.

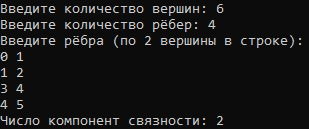


Рисунок 1 – Экранная форма программы

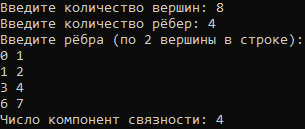


Рисунок 2 – Экранная форма программы

# Программа на C# (задача 1):

class Program

{

static List<int>[] graph;

static bool[] visited;

static void Main()

{

Console.Write("Введите количество вершин: ");

int n = int.Parse(Console.ReadLine());

Console.Write("Введите количество рёбер: ");

int m = int.Parse(Console.ReadLine());

graph = new List<int>[n];

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

graph[i] = new List<int>();

Console.WriteLine("Введите рёбра (по 2 вершины в строке):");

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

{

string[] input = Console.ReadLine().Split();

int u = int.Parse(input[0]);

int v = int.Parse(input[1]);

graph[u].Add(v);

graph[v].Add(u);

}

visited = new bool[n];

int components = 0;

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

{

if (!visited[i])

{

DFS(i);

components++;

}

}

Console.WriteLine($"Число компонент связности: {components}");

}

static void DFS(int v)

{

visited[v] = true;

foreach (int neighbor in graph[v])

{

if (!visited[neighbor])

DFS(neighbor);

}

}

}

# Тестирование (задача 2)

Примеры выполнения программы представлены на рисунках 3, 4.

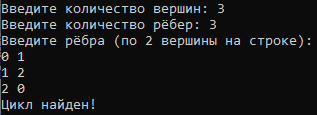


Рисунок 3 – Экранная форма программы

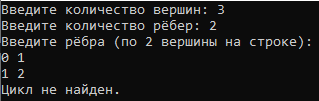


Рисунок 4 – Экранная форма программы

# Программа на C# (задача 2):

class Program

{

static void Main()

{

Console.Write("Введите количество вершин: ");

int vertices = int.Parse(Console.ReadLine());

Console.Write("Введите количество рёбер: ");

int edges = int.Parse(Console.ReadLine());

List<int>[] graph = new List<int>[vertices];

int[] inDegree = new int[vertices];

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

graph[i] = new List<int>();

Console.WriteLine("Введите рёбра (по 2 вершины на строке):");

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

{

string[] input = Console.ReadLine().Split();

int from = int.Parse(input[0]);

int to = int.Parse(input[1]);

graph[from].Add(to);

inDegree[to]++;

}

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

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

{

if (inDegree[i] == 0)

queue.Enqueue(i);

}

int visitedCount = 0;

while (queue.Count > 0)

{

int current = queue.Dequeue();

visitedCount++;

foreach (var neighbor in graph[current])

{

inDegree[neighbor]--;

if (inDegree[neighbor] == 0)

queue.Enqueue(neighbor);

}

}

if (visitedCount == vertices)

Console.WriteLine("Цикл не найден.");

else

Console.WriteLine("Цикл найден!");

}

}

# Тестирование (задача 3)

Примеры выполнения программы представлены на рисунках 5, 6.

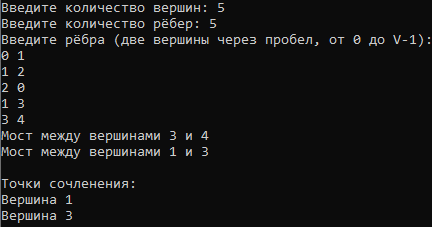


Рисунок 5 – Экранная форма программы

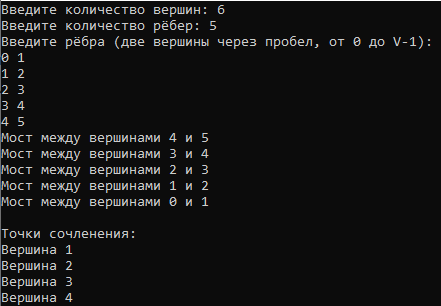


Рисунок 6 – Экранная форма программы

# Программа на C# (задача 3):

class Graph

{

private int V;

private List<int>[] adj;

private int time = 0;

public Graph(int v)

{

V = v;

adj = new List<int>[v];

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

adj[i] = new List<int>();

}

public void AddEdge(int u, int v)

{

adj[u].Add(v);

adj[v].Add(u);

}

public void FindArticulationPointsAndBridges()

{

bool[] visited = new bool[V];

int[] disc = new int[V];

int[] low = new int[V];

int[] parent = new int[V];

bool[] isArticulation = new bool[V];

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

{

parent[i] = -1;

visited[i] = false;

}

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

{

if (!visited[i])

DFS(i, visited, disc, low, parent, isArticulation);

}

Console.WriteLine("\nТочки сочленения:");

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

if (isArticulation[i])

Console.WriteLine($"Вершина {i}");

}

private void DFS(int u, bool[] visited, int[] disc, int[] low, int[] parent, bool[] isArticulation)

{

visited[u] = true;

disc[u] = low[u] = ++time;

int children = 0;

foreach (int v in adj[u])

{

if (!visited[v])

{

children++;

parent[v] = u;

DFS(v, visited, disc, low, parent, isArticulation);

low[u] = Math.Min(low[u], low[v]);

if (parent[u] == -1 && children > 1)

isArticulation[u] = true;

if (parent[u] != -1 && low[v] >= disc[u])

isArticulation[u] = true;

if (low[v] > disc[u])

Console.WriteLine($"Мост между вершинами {u} и {v}");

}

else if (v != parent[u])

{

low[u] = Math.Min(low[u], disc[v]);

}

}

}

}

class Program

{

static void Main()

{

Console.Write("Введите количество вершин: ");

int V = int.Parse(Console.ReadLine());

Console.Write("Введите количество рёбер: ");

int E = int.Parse(Console.ReadLine());

Graph g = new Graph(V);

Console.WriteLine("Введите рёбра (две вершины через пробел, от 0 до V-1):");

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

{

var input = Console.ReadLine().Split();

int u = int.Parse(input[0]);

int v = int.Parse(input[1]);

g.AddEdge(u, v);

}

g.FindArticulationPointsAndBridges();

}

}

# Тестирование (задача 4)

Примеры выполнения программы представлены на рисунках 7, 8.

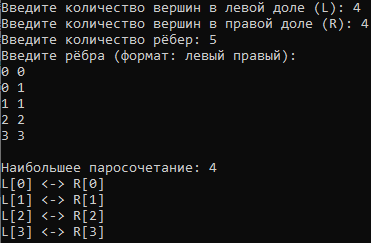


Рисунок 7 – Экранная форма программы

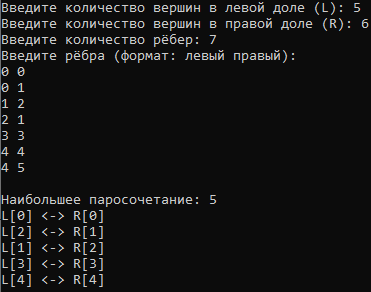


Рисунок 8 – Экранная форма программы

# Программа на C# (задача 4):

using System;

using System.Collections.Generic;

class Program

{

static int n, m;

static List<int>[] graph;

static int[] match;

static bool[] used;

static void Main()

{

Console.Write("Введите количество вершин в левой доле (L): ");

n = int.Parse(Console.ReadLine());

Console.Write("Введите количество вершин в правой доле (R): ");

m = int.Parse(Console.ReadLine());

Console.Write("Введите количество рёбер: ");

int e = int.Parse(Console.ReadLine());

graph = new List<int>[n];

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

graph[i] = new List<int>();

Console.WriteLine("Введите рёбра (формат: левый правый):");

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

{

var parts = Console.ReadLine().Split();

int u = int.Parse(parts[0]);

int v = int.Parse(parts[1]);

graph[u].Add(v);

}

match = new int[m];

for (int i = 0; i < m; i++) match[i] = -1;

int result = 0;

for (int v = 0; v < n; v++)

{

used = new bool[n];

if (TryKuhn(v))

result++;

}

Console.WriteLine($"\nНаибольшее паросочетание: {result}");

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

{

if (match[i] != -1)

Console.WriteLine($"L[{match[i]}] <-> R[{i}]");

}

}

static bool TryKuhn(int v)

{

if (used[v]) return false;

used[v] = true;

foreach (int u in graph[v])

{

if (match[u] == -1 || TryKuhn(match[u]))

{

match[u] = v;

return true;

}

}

return false;

}

}

# Вывод

В ходе лабораторной работы были успешно реализованы алгоритмы анализа графов, включая поиск компонент связности, циклов, точек сочленения и максимального паросочетаний.