БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Лабораторная работа № 13

по дисциплине «Структуры и алгоритмы обработки данных»

Вариант № 4

Выполнил студент: Ефименко Павел Викторович,

группа 981063

Минск 2020

**Условие:**

Представить ориентированный граф, состоящий из 7-10 вершин, с помощью матрицы смежности, а затем выполнить следующие операторы над его элементами. Указать вершину-источник, а затем решить следующие задачи.

1.   Кратчайшие пути от вершины-источника до всех вершин орграфа на основе алгоритма Дейкстры.

2.   Кратчайшие расстояния между каждой парой вершин орграфа на основе алгоритма Флойда.

**Листинг:**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Lr13

{

class Node

{

public string Name { get; }

public List<Node> Children { get; }

// Ветвь.

public Node(string name)

{

Name = name;

Children = new List<Node>();

}

// Добавление "ребенка" ветви.

public Node AddChildren(Node node, bool bidirect = true)

{

Children.Add(node);

if (bidirect)

{

node.Children.Add(this);

}

return this;

}

// Метод отображения посещённой ноды.

public void Handler()

{

Console.WriteLine($"visited {this.Name}");

}

}

/// <summary>

/// Класс, отображающий глубокий поиск.

/// </summary>

class DepthFirstSearch

{

private HashSet<Node> visited;

private LinkedList<Node> path;

private Node goal;

private bool limitWasReached;

// Алгоритм поиска в глубину.

public LinkedList<Node> DFS(Node start, Node goal)

{

visited = new HashSet<Node>();

path = new LinkedList<Node>();

this.goal = goal;

// Вызываем метод ниже.

DFS(start);

if (path.Count > 0)

{

path.AddFirst(start);

}

return path;

}

// Действие над отдельной ветвью.

private bool DFS(Node node)

{

//пишем в консоль что посетили ноду

node.Handler();

// если это не та что мы ищем, заканчиваем поиск

if (node == goal)

{

return true;

}

visited.Add(node);

//поиск дочерних элементов (которые мы еще не посещали)

foreach (var child in node.Children.Where(x => !visited.Contains(x)))

{

if (DFS(child)) // рекурсия

{

path.AddFirst(child);

return true;

}

}

return false;

}

//Поиск с ограничением глубины, отличается только указанием максимальной глубины поиска

public LinkedList<Node> DLS(Node start, Node goal, int limit)

{

visited = new HashSet<Node>();

path = new LinkedList<Node>();

limitWasReached = true;

this.goal = goal;

DLS(start, limit);

if (path.Count > 0)

{

path.AddFirst(start);

}

return path;

}

//действия над отдельной нодой

private bool DLS(Node node, int limit)

{

//пишем в консоль что посетили ноду

node.Handler();

// если это не та что мы ищем, заканчиваем поиск

if (node == goal)

{

return true;

}

if (limit == 0)

{

limitWasReached = false;

return false;

}

visited.Add(node);

//поиск дочерних элементов (которые мы еще не посещали)

foreach (var child in node.Children.Where(x => !visited.Contains(x)))

{

if (DLS(child, limit - 1)) // рекурсия

{

path.AddFirst(child);

return true;

}

}

return false;

}

}

class Program

{

static void Main(string[] args)

{

var n01 = new Node("01");

var n02 = new Node("02");

var n03 = new Node("03");

var n04 = new Node("04");

var n05 = new Node("05");

var n06 = new Node("06");

var n07 = new Node("07");

var n08 = new Node("08");

var n09 = new Node("09");

var n10 = new Node("10");

var n11 = new Node("11");

var n12 = new Node("12");

var n13 = new Node("13");

var n14 = new Node("14");

var n15 = new Node("15");

// Добавляем потомка.

n01.AddChildren(n02).AddChildren(n03);

n02.AddChildren(n05);

n03.AddChildren(n04);

n04.AddChildren(n05, false).AddChildren(n10, false).AddChildren(n11, false);

n06.AddChildren(n01, false);

n07.AddChildren(n03, false).AddChildren(n08);

n09.AddChildren(n08).AddChildren(n10);

n11.AddChildren(n12).AddChildren(n13);

n12.AddChildren(n13);

n14.AddChildren(n15);

var search = new DepthFirstSearch();

var path = search.DLS(n06, n13, 6);

PrintPath(path);

}

private static void PrintPath(LinkedList<Node> path)

{

Console.WriteLine();

if (path.Count == 0)

{

Console.WriteLine("You shall not pass!");

}

else

{

Console.WriteLine(string.Join(" -> ", path.Select(x => x.Name)));

}

Console.Read();

}

}

}

**Выполнение:**

