Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа № 6

По дисциплине ЯПИС

За шестой семестр

Тема: «Метод поиска в графе. Рекурсивная и не  
 рекурсивная реализации алгоритма DFS»

Вариант 8

Выполнил:

Студент 3 курса

Группы ИИ-16

Журавлёв В.А.

Проверил:

Слинко Е.В.

Брест, 2021

Цель: Сформировать представление о методе и алгоритме поиска в глубину на графе и его применении при решении задач (задачи класса сложности Р)

Задание

8. Разработайте и напишите программу нахождения количества связных компонент графа, используя поиск в глубину для определения всех вершин, принадлежащих одной связанной компоненте (определяется путь, связывающий между собой любую пару вершин). Рассмотрите случай, когда граф оказывается несвязным.

Код программы:

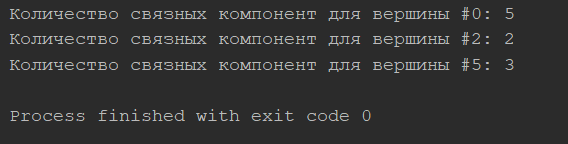
**Graph.java**

import javafx.util.Pair;  
  
import java.util.HashSet;  
import java.util.Set;  
  
public class Graph {  
  
 private Set<Integer>[] edges;  
  
 public Graph(int size) {  
 edges = new HashSet[size];  
 for (int i = 0; i < size; i++) {  
 edges[i] = new HashSet<>();  
 }  
 }  
  
 public void addEdge(Pair<Integer, Integer> edge) {  
 edges[edge.getKey()].add(edge.getValue());  
 edges[edge.getValue()].add(edge.getKey());  
 }  
  
 public int DFS(int startVertex) {  
 boolean[] visited = new boolean[edges.length];  
 return DFS(startVertex, visited);  
 }  
  
 private int DFS(int vertex, boolean[] visited) {  
 visited[vertex] = true;  
 Set<Integer> edge = edges[vertex];  
 return edge.stream()  
 .filter(edgeVertex -> !visited[edgeVertex])  
 .map(edgeVertex -> DFS(edgeVertex, visited))  
 .reduce((x, y) -> x + y)  
 .orElse(0) + 1;  
 }  
  
}

**Main.java**

import javafx.util.Pair;  
  
public class Main {  
 public static void main(String[] args) {  
 Graph graph = new Graph(10);  
 graph.addEdge(new Pair<>(0, 1));  
 graph.addEdge(new Pair<>(1, 6));  
 graph.addEdge(new Pair<>(6, 3));  
 graph.addEdge(new Pair<>(6, 4));  
 graph.addEdge(new Pair<>(6, 1));  
 graph.addEdge(new Pair<>(3, 1));  
 graph.addEdge(new Pair<>(7, 5));  
 graph.addEdge(new Pair<>(7, 8));  
 graph.addEdge(new Pair<>(2, 9));  
 System.*out*.println("Количество связных компонент для вершины #" + 0 + ": " + graph.DFS(0));  
 System.*out*.println("Количество связных компонент для вершины #" + 2 + ": " + graph.DFS(2));  
 System.*out*.println("Количество связных компонент для вершины #" + 5 + ": " + graph.DFS(5));  
 }  
}

Результат работы программы:



**Вывод:**Сформировал представление о методе и алгоритме поиска в глубину на графе и его применении при решении задач (задачи класса сложности Р).