import it.uniupo.graphLib.GraphInterface;

public class DFS

{

private GraphInterface graph;

private boolean found[];

public DFS(GraphInterface graph)

{

this.graph=graph;

found=new boolean[graph.getOrder()];

}

public GraphInterface getTree(int source)

{

GraphInterface tree=graph.create();

makeTree(tree,source);

return tree;

}

private void makeTree(GraphInterface tree, int source)

{

found[source]=true;

for(int neighbour:graph.getNeighbors(source))

if(!found[neighbour])

{

tree.addEdge(source,neighbour);

makeTree(tree,neighbour);

}

}

}

-----------------------------------------------------TEST------------------------------------------TEST-------------

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

import it.uniupo.graphLib.DirectedGraph;

import it.uniupo.graphLib.GraphInterface;

import it.uniupo.graphLib.UndirectedGraph;

class DFSTest

{

DFS dirDfs, undirDfs;

void init()

{

dirDfs=new DFS(new DirectedGraph("4;1 2;1 3;2 3;2 0;3 0;0 1"));

undirDfs=new DFS(new UndirectedGraph("4;1 2;1 3;2 3;2 0;3 0;0 1"));

}

@Test

void testDFS()

{

init();

assertNotNull(dirDfs);

assertNotNull(undirDfs);

}

@Test

void testGetTree()

{

init();

GraphInterface dirTree=dirDfs.getTree(0);

GraphInterface undirTree=undirDfs.getTree(0);

System.out.println(dirTree.toString());

System.out.println(undirTree.toString());

assertTrue(dirTree.hasEdge(0,1));

assertTrue(dirTree.hasEdge(1,2));

assertTrue(dirTree.hasEdge(2,3));

assertFalse(dirTree.hasEdge(3,0));

assertTrue(undirTree.hasEdge(0,2));

assertTrue(undirTree.hasEdge(1,2));

assertTrue(undirTree.hasEdge(1,3));

assertFalse(undirTree.hasEdge(3,0));

}

}