import java.util.ArrayList;

import it.uniupo.graphLib.GraphInterface;

public class BFS

{

private GraphInterface graph; //grafo

private ArrayList<Integer> queue; //coda

private boolean found[]; //nodi scoperti

public BFS(GraphInterface graph)

{

this.graph=graph;

queue=new ArrayList<Integer>();

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

}

public ArrayList<Integer> getNodesInOrderToVisit(int source)

{

ArrayList<Integer> order=new ArrayList<Integer>();

found[source]=true;

queue.add(source);

while(!queue.isEmpty()) //finchè la coda è non vuota

{

int node=queue.remove(0);//estraggo un nodo

order.add(node);

for(int neighbour:graph.getNeighbors(node))//cerco i suoi vicini non scoperti

if(!found[neighbour])//e li inserisco in coda

{

queue.add(neighbour);

found[neighbour]=true;

}

}

return order;

}

public int getDistance(int source,int dest)

{

int father[]=new int[graph.getOrder()];

father[source]=-1;

queue.add(source);

found[source]=true;

while(!queue.isEmpty())

{

int node=queue.remove(0);

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

if(!found[neighbour])

{

father[neighbour]=node;

found[neighbour]=true;

queue.add(neighbour);

}

}

int dist=0;//calcolo la distanza

for(int node=father[dest];node!=-1;node=father[node])

dist++; //"risalendo" l'array di padri partendo dalla destinazione

return dist;

}

public GraphInterface getBfsTree(int source)

{

GraphInterface tree=graph.create();

found[source]=true;

queue.add(source);

while(!queue.isEmpty())

{

int node=queue.remove(0);

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

if(!found[neighbour])

{

queue.add(neighbour);

found[neighbour]=true;

tree.addEdge(node, neighbour);

}

}

return tree;

}

}

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

import static org.junit.Assert.assertEquals;

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

import org.junit.jupiter.api.Test;

import it.uniupo.graphLib.DirectedGraph;

import it.uniupo.graphLib.GraphInterface;

import it.uniupo.graphLib.UndirectedGraph;

class BFSTest

{

BFS dirBfs, undirBfs;

void init()

{

dirBfs=new BFS(new DirectedGraph("3;0 2;2 1;1 0"));

undirBfs=new BFS(new UndirectedGraph("3;0 1;1 2;2 0"));

}

@Test

void testBFS()

{

init();

assertNotNull(dirBfs);

assertNotNull(undirBfs);

}

@Test

void testGetNodesInOrderToVisit()

{

init();

assertEquals(dirBfs.getNodesInOrderToVisit(0).toString(),"[0, 2, 1]");

assertEquals(undirBfs.getNodesInOrderToVisit(0).toString(),"[0, 1, 2]");

}

@Test

void testGetDistance()

{

init();

assertEquals(dirBfs.getDistance(0,1),2);

assertEquals(dirBfs.getDistance(0,2),1);

}

@Test

void testGetBfsTree()

{

init();

GraphInterface dirBfsTree=dirBfs.getBfsTree(0);

GraphInterface undirBfsTree=undirBfs.getBfsTree(0);

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

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

assertFalse(dirBfsTree.hasEdge(0,1));

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

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

assertFalse(undirBfsTree.hasEdge(2,1));

}

}