题目：实验三：深度优先遍历

时间：2020年11月18日星期三

代码：

package exp03;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Stack;

/\*\*

\* @author zhenwen

\* @date 2020/11/13

\*/

public class Algorithm {

private static int t = 0;

private static int s = 0;

private static HashMap<String, Integer> discoverTime;

private static HashMap<String, Integer> finishTime;

public static void main(String[] args) {

Graph graph = createCompleteGraph(5);

graph.print();

traverse(graph, graph.getVertices().get(0));

}

private static Graph createCompleteGraph(int vertexCount) {

List<Vertex> vertices = new ArrayList<>();

List<Edge> edges = new ArrayList<>();

String alphaBet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

for (int i = 0; i < vertexCount; i++) {

vertices.add(new Vertex(alphaBet.charAt(i) + ""));

}

// 完全图

for (Vertex vp : vertices) {

for (Vertex vn : vertices) {

if (vp.equals(vn)) {

continue;

}

edges.add(new Edge(vp, vn));

}

}

return new Graph(vertices, edges);

}

private static void dfs(Graph g, Vertex v) {

v.setVisited(true);

System.out.println(v.getName());

for (Edge e : g.getEdges(v)) {

if (!e.getNext().getVisited()) {

dfs(g, e.getNext());

}

}

}

private static void traverse(Graph g, Vertex v) {

Stack<Vertex> stack = new Stack<>();

stack.push(v);

while (!stack.empty()) {

v = stack.pop();

if (!v.getVisited()) {

v.setVisited(true);

System.out.println(v.getName());

for (Edge e : g.getEdges(v)) {

stack.push(e.getNext());

}

}

}

}

private static void dfsTimestamp(Graph g, Vertex v) {

v.setVisited(true);

discoverTime.put(v.getName(), t);

t += 1;

for (Edge e : g.getEdges()) {

if (!e.getNext().getVisited()) {

dfsTimestamp(g, e.getNext());

}

}

finishTime.put(v.getName(), s);

s += 1;

}

}

class Graph {

private List<Vertex> vertices;

private List<Edge> edges;

public Graph() {

}

public Graph(List<Vertex> vertices, List<Edge> edges) {

this.vertices = vertices;

this.edges = edges;

}

public List<Vertex> getVertices() {

return vertices;

}

public void setVertices(List<Vertex> vertices) {

this.vertices = vertices;

}

public List<Edge> getEdges() {

return edges;

}

public List<Edge> getEdges(Vertex v) {

List<Edge> edges = new ArrayList<>();

for (Edge e : getEdges()) {

if (e.getPrior().equals(v)) {

edges.add(e);

}

}

return edges;

}

public void setEdges(List<Edge> edges) {

this.edges = edges;

}

@Override

public String toString() {

return "Graph{" +

"vertices=" + vertices +

", edges=" + edges +

'}';

}

public void print() {

System.out.println(toString());

}

}

class Edge {

private Vertex prior;

private Vertex next;

private Boolean isVisited;

public Edge() {

}

public Edge(Vertex prior, Vertex next) {

this(prior, next, false);

}

public Edge(Vertex prior, Vertex next, Boolean isVisited) {

this.prior = prior;

this.next = next;

this.isVisited = isVisited;

}

public Vertex getPrior() {

return prior;

}

public void setPrior(Vertex prior) {

this.prior = prior;

}

public Vertex getNext() {

return next;

}

public void setNext(Vertex next) {

this.next = next;

}

public Boolean getVisited() {

return isVisited;

}

public void setVisited(Boolean visited) {

isVisited = visited;

}

@Override

public String toString() {

return prior + " -> " + next;

}

}

class Vertex {

private String name;

private Boolean isVisited;

public Vertex() {

this(null, false);

}

public Vertex(String name) {

this(name, false);

}

public Vertex(String name, Boolean isVisited) {

this.name = name;

this.isVisited = isVisited;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Boolean getVisited() {

return isVisited;

}

public void setVisited(Boolean visited) {

isVisited = visited;

}

@Override

public String toString() {

return name;

}

}

运行结果及截图：

