广度优先搜索算法

171491106 安南 硬件一班

1. 实验题目

广度优先搜索算法（BFS）

1. 算法简介

广度优先搜索算法（Breadth-First Search，BFS）是一种盲目搜寻法，目的是系统地展开并检查图中的所有节点，以找寻结果。换句话说，它并不考虑结果的可能位置，彻底地搜索整张图，直到找到结果为止。

1. 实验目的

成功编写一个广度优先算法，找到从起点到终点的最短路径。

1. 实验代码

package 广度优先;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.concurrent.LinkedBlockingQueue;

public class 算法 {

public static void main(String[] args) {

HashMap<String,String[]> hashMap=new HashMap<>();

hashMap.put("AN",new String[]{"GUO","CHANG","NING"});

hashMap.put("GUO",new String[]{"AN","ADI","GONG"});

hashMap.put("ADI",new String[]{"GUO"});

hashMap.put("GONG",new String[]{"GUO"});

hashMap.put("CHANG",new String[]{"AN","TONG"});

hashMap.put("NING",new String[]{"AN","TONG","YEAH"});

hashMap.put("TONG",new String[]{"NING","CHANG"});

hashMap.put("YEAH",new String[]{"NING"});

Node target = findTarget("AN","YEAH",hashMap);

//打印出最短路径的各个节点信息

printSearPath(target);

}

static void printSearPath(Node target) {

if (target != null) {

System.out.print("找到了目标节点:" + target.id + "\n");

List<Node> searchPath = new ArrayList<Node>();

searchPath.add(target);

Node node = target.parent;

while(node!=null) {

searchPath.add(node);

node = node.parent;

}

String path = "";

for(int i=searchPath.size()-1;i>=0;i--) {

path += searchPath.get(i).id;

if(i!=0) {

path += "-->";

}

}

System.out.print("步数最短："+path);

} else {

System.out.print("未找到了目标节点");

}

}

static Node findTarget(String startId,String targetId,HashMap<String,String[]> map) {

List<String> hasSearchList = new ArrayList<String>();

LinkedBlockingQueue<Node> queue=new LinkedBlockingQueue<>();

queue.offer(new Node(startId,null));

while(!queue.isEmpty()) {

Node node = queue.poll();

if(hasSearchList.contains(node.id)) {

continue;

}

System.out.print("判断节点:" + node.id +"\n");

if (targetId.equals(node.id)) {

return node;

}

hasSearchList.add(node.id);

if (map.get(node.id) != null && map.get(node.id).length > 0) {

for (String childId : map.get(node.id)) {

queue.offer(new Node(childId,node));

}

}

}

return null;

}

static class Node{

public String id;

public Node parent;

public Node(String id,Node parent) {

this.id = id;

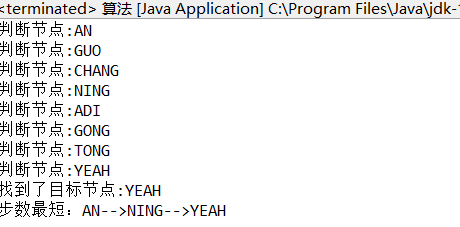
this.parent = parent;

}

}

}

1. 实验结果



1. 实验总结

BFS最适合这种给定初始状态跟目标状态，要求从初始状态到目标状态的最短路径的问题。