广度优先搜索

1. 实验题目

利用广度优先搜索算法（BFS）搜索goal

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，它的原理与队列相似，先进先出，就是从左到右搜索同一层数的节点，然后将搜索的节点的子节点也按照从左到右的顺序输入队列，直到搜索到goal。

1. 实验目的

编写一个广度优先算法，搜索到目标节点的最短路径。

1. 实验代码

**package** bfs;

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.concurrent.LinkedBlockingQueue;

**public** **class** BFS {

**public** **static** **void** main(String[] args) {

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

hashMap.put("A",**new** String[]{"B","C","D"});

hashMap.put("B",**new** String[]{"A","E","F"});

hashMap.put("E",**new** String[]{"B"});

hashMap.put("F",**new** String[]{"B"});

hashMap.put("C",**new** String[]{"A","G"});

hashMap.put("D",**new** String[]{"A","G","GOAL"});

hashMap.put("G",**new** String[]{"D","C"});

hashMap.put("GOAL",**new** String[]{"D"});

Node target = *findTarget*("A","GOAL",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;

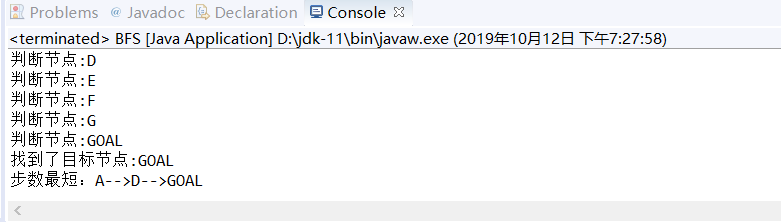
**this**.parent = parent;

}

}

}

## 五实验截图



#### 六实验总结

通过这次实验，我懂得了广度优先搜索的基本原理和如何进行广度优先搜索，要充分利用先进先出的特点和利用辅助方法记录已经检测完的节点防止重复搜索。