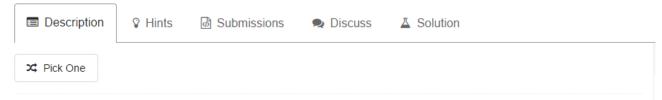
207. Course Schedule



There are a total of n courses you have to take, labeled from o to n-1.

Some courses may have prerequisites, for example to take course 0 you have to first take course 1, which is expressed as a pair:

[0,1]

Given the total number of courses and a list of prerequisite pairs, is it possible for you to finish all courses?

Example 1:

Example 2:

Note:

- The input prerequisites is a graph represented by a list of edges, not adjacency matrices. Read more about how a graph is represented.
- 2. You may assume that there are no duplicate edges in the input prerequisites.

这个题目是一个有向图检测环,分三步走。(BFS)

- 1. 计算每个点的入度
- 2. 找出入度为0的点 (表示它可以被执行)
- 3. 然后去除从它出去的边,其indegree减一,回到第二步

```
public class L207 {
public boolean canFinish(int numCourses, int[][] prerequisites) {
    if (numCourses <= 0) {</pre>
       return true;
    List<List<Integer>> fromTo = new ArrayList<List<Integer>>();
    for(int i = 0; i < numCourses; i ++) {
        fromTo.add(new ArrayList<Integer>());
    for(int [] edge : prerequisites) {
        fromTo.get(edge[1]).add(edge[0]);
    //计算每门课的indegree
    int [] inDegree = new int[numCourses];
    for(int [] edge : prerequisites) {
        inDegree[edge[0]] ++;
    //res存储的是有多少个已经解决了,并将indegree为0的课加入到que中
    List<Integer> res = new ArrayList<>();
    LinkedList<Integer> que = new LinkedList<>();
    for(int i = 0; i < inDegree.length; i ++) {</pre>
        if(inDegree[i] == 0) {
            que.add(i);
    //从que里面poll出来的课程,去掉他们的outdegree edge
    while (!que.isEmpty()) {
       int source = que.poll();
       res.add(source);
       for(int destination : fromTo.get(source)) {
           inDegree[destination] --;
           //若是终点的indegree减一后为0,就加到que中
           if(inDegree[destination] == 0) {
               que.add(destination);
       }
   return res.size() == numCourses;
}
```

}