207. Course Schedule

There are a total of n courses you have to take, labeled from 0 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:

```
Input: 2, [[1,0],[0,1]]
Output: false
Explanation: There are a total of 2 courses to take.
To take course 1 you should have finished course 0,
and to take course 0 you should also have finished
course 1. So it is impossible.
```

```
class Solution {
    public boolean canFinish(int numCourses, int[][] prerequisites) {

        // save the edges of each point
        ArrayList[] edges = new ArrayList[numCourses];
        int[] indegree = new int[numCourses];
        Queue<Integer> queue = new LinkedList<>();
        int count = 0;

        for (int i = 0; i < numCourses; i++)
            edges[i] = new ArrayList<>();

        for (int i = 0; i < prerequisites.length; i++) {
            indegree[prerequisites[i][1]]++;
            edges[prerequisites[i][0]].add(prerequisites[i][1]);
        }

        // start with indegree which is 0
        for (int i = 0; i < indegree.length; i++) {
            if (indegree[i] == 0) {
                queue.add(i);
                count++;
            }
        }

        while(!queue.isEmpty()) {
            int curr = (int)queue.poll();

            // traverse all edges of curr
            // reduce the indegree of each edge by 1
            // if indegree is 0 the add to the queue
            for(int i = 0; i < edges[curr].size(); i++) {
                int pointer = (int)edges[curr].get(i);
                indegree[pointer]--;
                if (indegree[pointer] == 0) {
                    queue.add(pointer);
                    count++;
                }
            }
        }

       return count == numCourses;
    }
}
```