Topological Sorting

- Course Schedule II(图的拓扑排序)

```
public class Solution {
    public int[] findOrder(int numCourses, int[][] prerequisites) {
        // write your code here
        if(numCourses == 0){
            return null;
        }
        List[] edges = new ArrayList[numCourses];//all the neighbors here
        int[] degree = new int[numCourses];
        for(int i = 0; i < numCourses; i++){
            edges[i] = new ArrayList<Integer>(); // add list for each node
        }

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

        Queue<Integer> queue = new LinkedList<>();
        for(int i = 0; i < numCourses; i++){
            if(degree[i] == 0){
                queue.add(i);
            }
        }
        int count  = 0;
        int[] order = new int [numCourses];
        while(!queue.isEmpty()){
            int course = queue.poll();
            order[count] = course;
            count++;

            for(int i = 0; i < edges[course].size(); i++){//loop all the neighbors of the node and add the suitable nodes to queue
                int n = (int)edges[course].get(i);
                degree[n]--;
                if(degree[n] == 0){
                    queue.add(n);
                }
            }
        }
        if(count == numCourses){//compare the amount to judge if there is a cycle
            return order;
        }
        return new int[0];
    }
}
```