[1,0]的意思是先做0，在做1

210. Course Schedule II

Description | Hints | Submissions | Discuss | Solution

Pick One

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**, return the ordering of courses you should take to finish all courses.

There may be multiple correct orders, you just need to return one of them. If it is impossible to finish all courses, return an empty array.

Example 1:

```
Input: 2, [[1,0]]
Output: [0,1]
Explanation: There are a total of 2 courses to take. To take course 1 you should have finished
             course 0. So the correct course order is [0,1] .
```

Example 2:

```
Input: 4, [[1,0],[2,0],[3,1],[3,2]]
Output: [0,1,2,3] or [0,2,1,3]
Explanation: There are a total of 4 courses to take. To take course 3 you should have finished both
             courses 1 and 2. Both courses 1 and 2 should be taken after you finished course 0.
             So one correct course order is [0,1,2,3]. Another correct ordering is [0,2,1,3] .
```

Note:

1. 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.

Seen this question in a real interview before? Yes No

```java
public class L210 {
      public int[] findOrder(int numCourses, int[][] prerequisites) {

          int [] map = new int[numCourses];
          for(int i = 0; i < prerequisites.length; i ++) { //计算每个点的入度
              map[prerequisites[i][0]] ++;
          }
          Queue<Integer> queue = new LinkedList<Integer>();//记录入度为0的点
          for(int i = 0; i < map.length; i ++) {
              if(map[i] == 0)
                  queue.add(i); //初始的入度为0的点
          }
          //用list的原因是为了找到一个顺序，从1到最后就是一个顺序
          List<Integer> res = new ArrayList<Integer>();
          int count = queue.size();

          while(!queue.isEmpty()) {
              int temp = queue.poll();
              res.add(temp);
              for(int i = 0; i < prerequisites.length; i ++) {
                  if(temp == prerequisites[i][1]) {
                      int t = prerequisites[i][0];
                      map[t] --;
                      if(map[t] == 0) {
                          queue.add(t);
                          count ++;
                      }
                  }
              }
          }
          if(count != numCourses) {
              int [] a = new int [0];
              return a;
          }else {
            int [] a = new int [res.size()];
            for(int i = res.size() - 1; i >= 0; i ++) { //这里需要反转一下，
                a[i] = res.get(i);
            }
            return a;
        }
      }
}
```