/\*

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.

For example:

2, [[1,0]]

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]

4, [[1,0],[2,0],[3,1],[3,2]]

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].

拓扑遍历！

\*/

class Solution {

public:

int findpoint(vector<int> &indegree,int num)

{

for(int i=0;i<num;i++)

{

if(indegree[i]==0)

{

indegree[i]--;

return i;

}

}

return -1;

}

vector<int> findOrder(int num, vector<pair<int, int>>& prer)

{

vector<vector<int>> p;

for(int i=0;i<num;i++)

{

vector<int> g(num,0);

p.push\_back(g);

}

vector<int> indegree(num,0);//入度

for(vector<pair<int, int>>::iterator it=prer.begin();it!=prer.end();it++)

{

p[it->first][it->second]=1;

indegree[it->second]++;

}

vector<int> ret;

for(int i=1;i<=num;i++)

{

int k=findpoint(indegree,num);

if(k==-1)

{

ret.clear();

return ret;

}

else

ret.push\_back(k);

for(int j=0;j<num;j++)

{

if(p[k][j]==1)

indegree[j]--;

}

}

reverse(ret.begin(),ret.end());

return ret;

}

};