/\*

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?

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 it is possible.

2, [[1,0],[0,1]]

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.

这个题的关键在于 就是图的思想，有没有环的存在

方法是使用拓扑排序思想，找入度为0的点，然后删除这个点以及出边，重复这个过程，如果找不到入度为0得点就说明有环！

使用indegree数组来存入度。

\*\*\*\*\*\*\*

如何获得一个图的拓扑排序：找到图中所有入度为0的点，放入序列，删除这些点和以这些点为出度的边，再找所有入度为0的点，依次循环。

如何通过拓扑排序判断图中是否有环：拓扑排序之后，若还剩有点，则表示有环。

\*\*\*\*\*\*\*

\*/

class Solution {

public:

int findpoint(vector<int> &indegree,int num)//找入度为0的点

{

int flag=0;

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

{

if(indegree[i]==0)

{

indegree[i]--;

return i;

}

}

return -1;

}

bool canFinish(int num, vector<pair<int, int>>& prer)

{

//形成邻接矩阵

vector<vector<int>> p;

vector<int> indegree(num,0);

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

{

vector<int> g(num,0);

p.push\_back(g);

}

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

{

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

indegree[it->second]++;

}

//就是看一个有向图是否形成环，return 0。

int cun=0;

while(cun!=num)

{

int k=findpoint(indegree,num);

if(k==-1) //没有入度为0的点

return false;

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

{

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

indegree[i]--;

}

cun++;

}

return true;

}

};