

一、编程题

ACM：查找一个有向网络的头节点和尾节点

题目描述：给定一个有向图，图中可能包含有环，图使用二维矩阵表示，每一行的第一列表示起始节点，第二列表示终止节点，如[0, 1]表示从0到1的路径。每个节点用正整数表示。求这个数据的首节点与尾节点，题目给的用例会是一个首节点，但可能存在多个尾节点。同时，图中可能含有环。如果图中含有环，返回[-1]。

说明：入度为0是首节点，出度为0是尾节点。

输入描述：第一行为后续输入的键值对数量 $N \geq 0$ ，第二行为 $2N$ 个数字。每两个为一个起点，一个终点。如：

输出描述：输出一行头节点和尾节点。如果有多个尾节点，按从大到小的顺序输出。

补充说明：如果图有环，输出为-1。
所有输入均合法，不会出现不配对的数据

示例1
输入：4
0 1 0 2 1 2 2 3
输出：0 3
说明：

示例2
输入：2
0 1 0 2
输出：0 1 2
说明：

代码：

```
import sys
```

```
for line in sys.stdin:
    nodes = list(map(int,line.split()))
```

```
def accessible(node):
    global edges
    if node not in edges:
        return []
    for v in edges[node]:
        yield v
        yield from accessible(v)
```

```

edges = {}
for i in range(len(nodes)//2):
    if nodes[i*2] not in edges:
        edges[nodes[i*2]] = []
    edges[nodes[i*2]].append(nodes[i*2+1])

ring = False
for node in nodes:
    try:
        [i for i in accessible(node)]
    except RecursionError:
        ring=True

if ring:
    print('-1')
else:

    res = []
    ins = []
    for v in edges.values():
        ins.extend(v)

    for node in sorted(list(set(nodes))):
        if node in edges.keys() and node not in ins:
            res.append(str(node))
    for node in sorted(list(set(nodes))):
        if node in ins and node not in edges.keys():
            res.append(str(node))

    if len(res)==0:
        print('-1')
    else:
        print(' '.join(res))

```