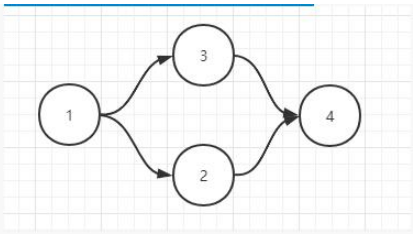
题目描述：

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>=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：

```
#include <iostream>
#include <iterator>
```

```
using namespace std;
#include <map>
#include <list>
#include <algorithm>
#include <set>
struct point{
    bool maybe_head = true;
    std::list<int> next;
};
std::map<int, point> mp;
std::set<int> path;
std::set<int> tail;
void step(int id)
{
    if(!path.emplace(id).second)//走过了 环
    {
        throw -1;
    }
    auto& p = mp.at(id);
    if(p.next.empty()) //is end
    {
        tail.emplace(id);
    }
    for(auto& it : p.next)
    {
        step(it);
    }
    path.erase(id);
}

int main() {
    int a, b;
    cin >> a;
    if (a == 0)
    {
        std::cout << "-1" << std::endl;
        return 0;
    }
    while (cin >> a >> b) { // 注意 while 处理多个 case
        mp[b].maybe_head = false;
        mp[a].next.emplace_back(b);
    }
    auto head = std::find_if(mp.begin(), mp.end(), [](auto& pt){return
pt.second.maybe_head;});
```

```
    if (head == mp.end())
    {
        std::cout << "-1";
        return 0;
    }
    try{
    step(head->first);
    }
    catch(...)
    {
        std::cout << "-1" << std::endl;
        return 0;
    }
    std::cout << head->first << " ";
    for(auto it = tail.begin(); it != tail.end(); it++)
    {
        std::cout << *it << " ";
    }
    std::cout << std::endl;
    return 0;
}
// 64 位输出请用 printf("%lld")
```