29、目录删除, 考点 or 实现——数据结构/二叉树

题目描述

某文件系统中有 N 个目录,每个目录都有一个独一无二的 ID。

每个目录只有一个父目录,但每个父目录下可以有零个或者多个子目录,目录结构呈树状结构。

假设,根目录的 ID 为 0,且根目录没有父目录,其他所有目录的 ID 用唯一的正整数表示,并统一编号。

现给定目录 ID 和其父目录 ID 的对应父子关系表[子目录 ID],以及一个待删除的目录 ID,请计算并返回一个 ID 序列,表示因为删除指定目录后剩下的所有目录,返回的ID序列以递增序输出。

注音

- 1、被删除的目录或文件编号一定在输入的 ID 序列中;
- 2、当一个目录删除时,它所有的子目录都会被删除。

输入描述

输入的第一行为父子关系表的长度 m;

接下来的 m 行为 m 个父子关系对;

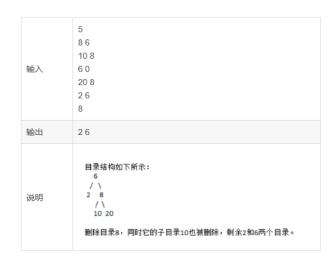
最后一行为待删除的 ID。

序列中的元素以空格分割,参见样例。

输出描述

输出一个序列,表示因为删除指定目录后,剩余的目录 ID。

用例



题目解析

本题咋看上去是让模拟N叉树结构,然后做节点删除操作,最后遍历N叉树。

但是这样的话思考的话,就太复杂了。

本题其实并不需要删除节点,也不需要遍历N叉树,我们可以在模拟N叉树的过程中,就统计节点,并排除要删除的节点的插入。

我首先,统计了所有父节点下的子节点,比如

86

108

60

可以统计为:

然后从根节点0开始遍历N叉树

然后从根节点0开始遍历N叉树

```
1 | let children = tree[0]
2 | for(let i=0; i<children.length; i++) {
3 | if(children[i] !== remove) {
4 | // 记录树节点
5 | // 递归处理children[i],即将children[i]当成父节点继续遍历查境其子节点,重复上面逻辑
6 | }
7 | }
```

Java算法源码

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;
import java.util.StringJoiner;
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int m = sc.nextInt();
    int[][] relations = new int[m][2];
     relations[i][0] = sc.nextInt();
      relations[i][1] = sc.nextInt();
    int del = sc.nextInt();
    System.out.println(getResult(m, relations, del));
  public static String getResult(int m, int[][] relations, int del) {
    HashMap<Integer, ArrayList<Integer>> tree = new HashMap<>();
    for (int[] relation : relations) {
     int child = relation[0];
     int father = relation[1]:
```

JS算法源码

```
let tree = {};

for (let i = 0; i < arr.length; i++) {
    let [child, father] = arr[i];
    tree[father] ? tree[father].push(child) : (tree[father] = [child]);
}

if (del === 0) return "";

const res = [];

dfs(tree, 0, del, res);

return res.sort((a, b) => a - b).join(" ");
}

function dfs(tree, node, del, res) {
    const children = tree[node];
    if (children)
    for (let i = 0; i < children.length; i++) {
        if (children[i] !== del) {
            res.push(children[i]);
            dfs(tree, children[i], del, res);
        }
}

for (let i = 0; i < children[i], del, res);
}
</pre>
```

Python算法源码

print(getResult())

36

```
m = int(input())
    relations = [list(map(int, input().split())) for _ in range(m)]
    remove = int(input())
    def dfs(tree, node, remove, res):
         if tree.get(node) is not None:
            children = tree[node]
 10
            for child in children:
                if child != remove:
                    res.append(child)
                     dfs(tree, child, remove, res)
    def getResult():
        tree = {}
         for child, father in relations:
20
            if tree.get(father) is None:
                 tree[father] = []
            tree[father].append(child)
24
         if remove == 0:
            return ""
27
         res = []
28
29
         dfs(tree, 0, remove, res)
30
31
         res.sort()
         return " ".join(map(str, res))
32
34
```