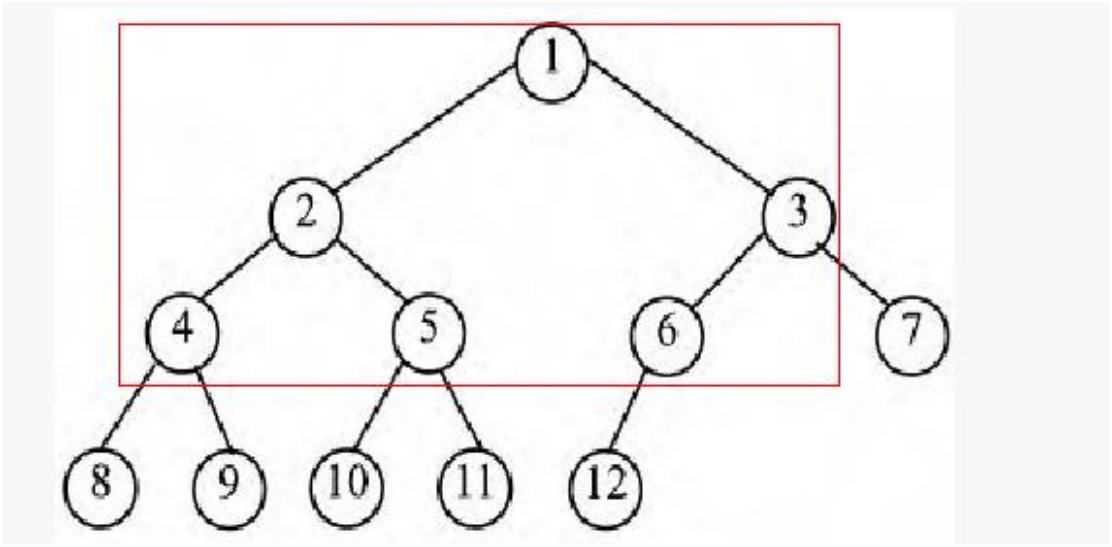
题目描述：
给定一个以顺序储存结构存储整数值的完全二叉树序列（最多 1000 个整数），请找出此完全二叉树的所有非叶子节点部分，然后采用后序遍历方式将此部分树（不包含叶子）输出。
1、只有一个节点的树，此节点认定为根节点（非叶子）。
2、此完全二叉树并非满二叉树，可能存在倒数第二层出现叶子或者无右叶子的情况




其他说明：二叉树的后序遍历是基于根来说的，遍历顺序为：左-右-根
输入描述：
一个通过空格分割的整数序列字符串
输出描述：
非叶子部分树结构的后序遍历结果
补充说明：
输出数字以空格分隔

示例 1
输入：
1 2 3 4 5 6 7
输出：
2 3 1
说明：
找到非叶子部分树结构，然后采用后续遍历输出

```
import java.util.Scanner;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner in=new Scanner(System.in);
        int[] arr = Arrays.stream(in.nextLine().split(" ")).mapToInt(Integer::parseInt).toArray();
        if(arr.length==1){
            System.out.println(arr[0]);
```

```
            return;
        }
        int[] nums=new int[arr.length+1];
        for (int i = 1; i < nums.length; i++) {
            nums[i]=arr[i-1];
        }
        //确定第一个叶子节点的位置,叶子节点没有左孩子
        int idx=1;
        while(idx*2<nums.length){
            idx++;
        }
        ArrayList<Integer> path = new ArrayList<>();
        print(nums,1,idx,path);
        StringJoiner sj=new StringJoiner(" ");
        path.stream().forEach(e->sj.add(String.valueOf(e)));
        System.out.println(sj.toString());

    }

    private static void print(int[] nums, int root, int end, List<Integer> path){
        //先打印左子树
        if (root>=end){
            return;
        }
        print(nums,root*2,end,path);
        print(nums,root*2+1,end,path);
        path.add(nums[root]);
    }
}
```