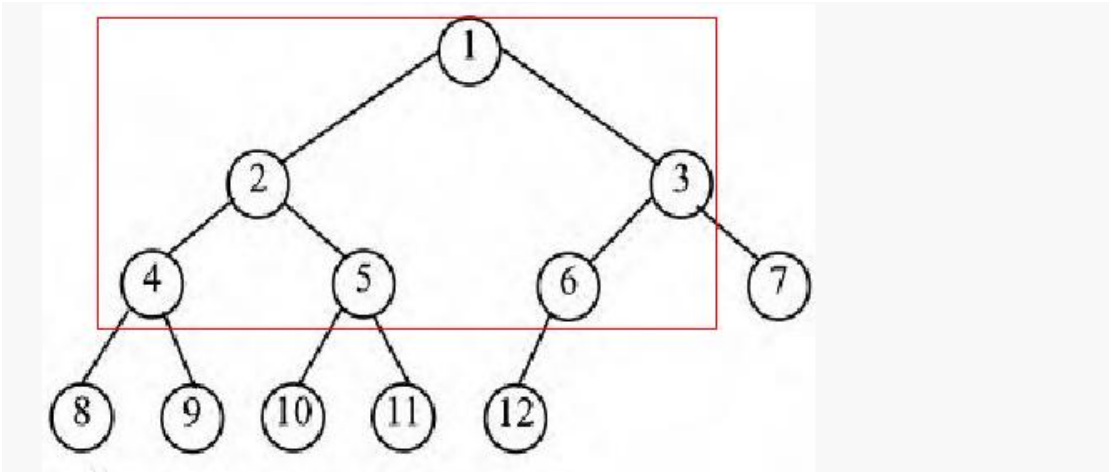
题目描述：
给定一个以顺序储存结构存储整数值的完全二叉树序列（最多 1000 个整数），请找出此完全二叉树的所有非叶子节点部分，然后采用后序遍历方式将此部分树（不包含叶子）输出。
1、只有一个节点的树，此节点认定为根节点（非叶子）。
2、此完全二叉树并非满二叉树，可能存在倒数第二层出现叶子或者无右叶子的情况



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.ArrayList;
import java.util.Scanner;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        ArrayList<Integer> list = new ArrayList<Integer>();
        while (in.hasNextInt()) { // 注意 while 处理多个 case
```

```
            int a = in.nextInt();
            list.add(a);
        }

        Integer[] treeArr = list.toArray(new Integer[0]);
        ArrayList<Integer> result = new ArrayList<Integer>();
        func(treeArr, 0, result);

        for (int j = 0; j < result.size(); j++) {
            if (j > 0) {
                System.out.print(" ");
            }
            Integer i = result.get(j);
            System.out.print(i);
        }
    }

    static void func(Integer[] treeArr, int i, ArrayList<Integer> result) {
        if (i >= treeArr.length) {
            return;
        }

        int leftIx = 2 * i + 1;
        if (leftIx >= treeArr.length && i > 0) {
            // 叶子节点
            return;
        }

        // if (leftIx < treeArr.length) {
        func(treeArr, leftIx, result);
        // }
        if (leftIx + 1 < treeArr.length) {
            func(treeArr, leftIx + 1, result);
        }
        result.add(treeArr[i]);
    }
}
```