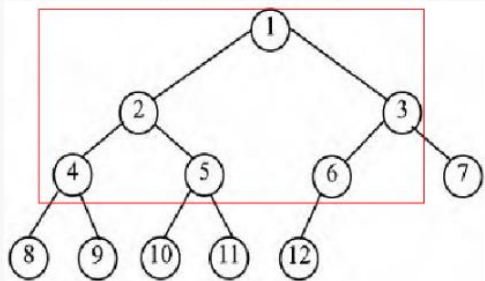


完全二叉树非叶子部分后序遍历

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

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.List;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String str = scanner.nextLine();
        String[] start = str.split(" ");
        int length = start.length;
        List<Integer> temp = new ArrayList<>();
        int i = 0;
        for (String s : start) {
            temp.add(Integer.parseInt(s));
            i++;
            if(i * 2 + 1 >= length){
                break;
            }
        }
        List<Integer> res = new ArrayList<>();
```

```

        postTravel(res, 0, temp);
        String result = "";
        for (int j = 0; j < res.size(); j++) {
            result += res.get(j);
            if(j < res.size() - 1){
                result += " ";
            }
        }
        System.out.println(result);
    }

    private static void postTravel(List list, int index, List temp){
        if(index >= temp.size()){
            return;
        }
        postTravel(list, 2 * index + 1, temp);
        postTravel(list, 2 * index + 2, temp);
        list.add(temp.get(index));
    }
}

```