题目描述：

定义构造三叉搜索树规则如下：

每个节点都存有一个数，当插入一个新的数时，从根节点向下寻找，直到找到一个合适的空节点插入。

查找的规则是：

1. 如果数小于节点的数减去 500，则将数插入节点的左子树
2. 如果数大于节点的数加上 500，则将数插入节点的右子树
3. 否则，将数插入节点的中子树

给你一系列数，请按以上规则，按顺序将数插入树中，构建出一棵三叉搜索树，最后输出树的高度。

输入描述：

第一行为一个数 N，表示有 N 个数，1<=N<=10000
第二行为 N 个空格分隔的整数，每个数的范围为[1,10000]

输出描述：

输出树的高度(根节点的高度为 1)

示例 1

输入：

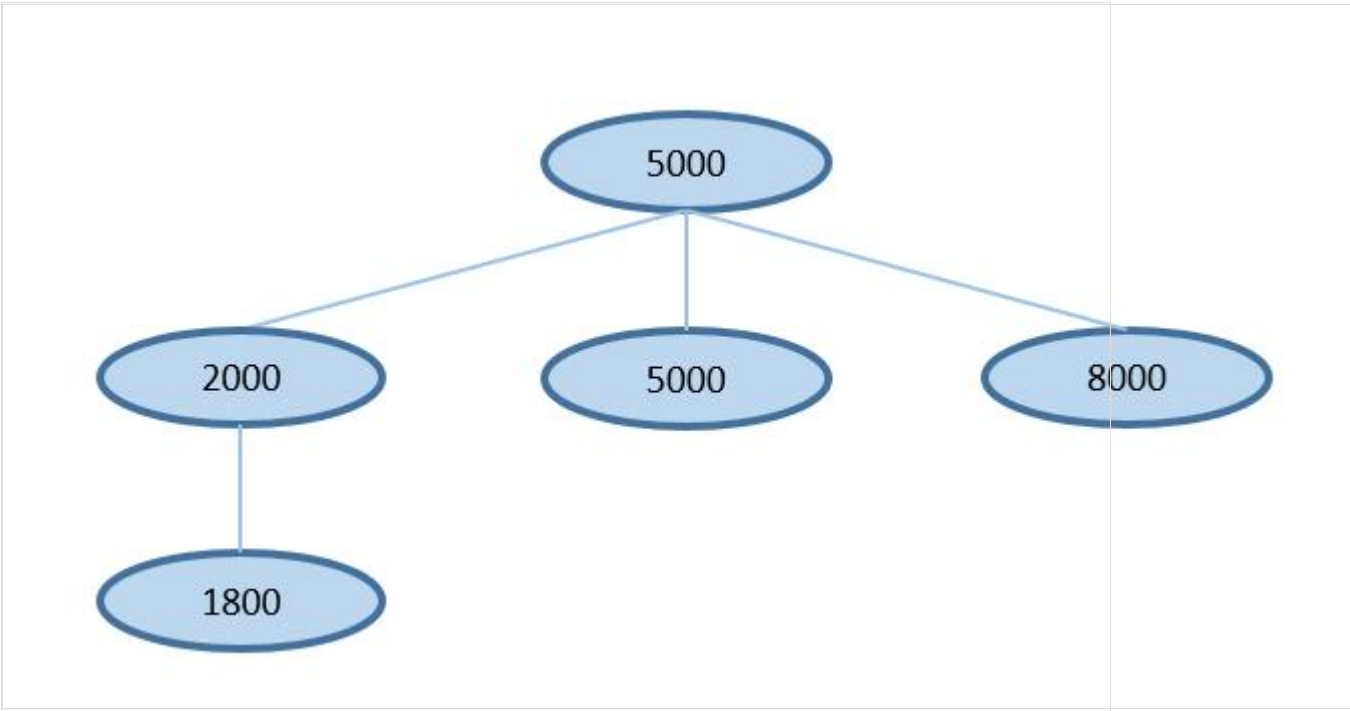
```
5
5000 2000 5000 8000 1800
```

输出：

```
3
```

说明：

最终构造出的树如下，高度为 3：



示例 2

输入：

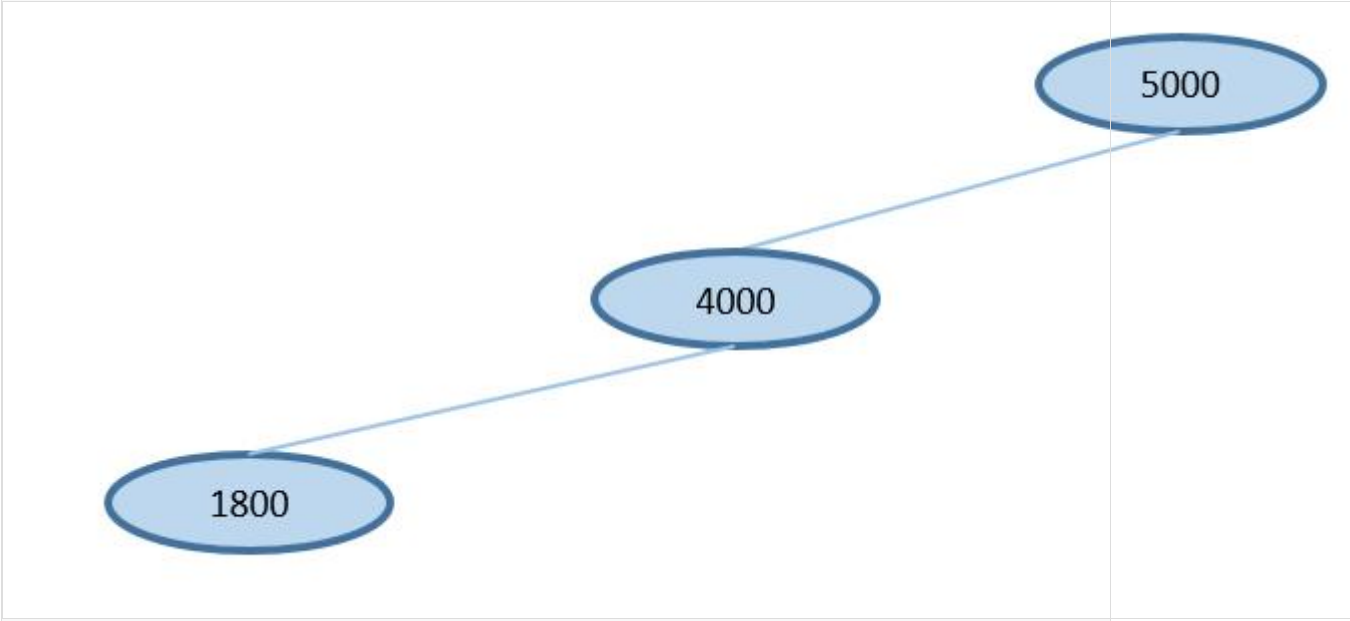
```
3
5000 4000 3000
```

输出：

```
3
```

说明：

最终构造出的树如下，高度为 3：




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

```
// 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);
        int count = in.nextInt();
                    int[] nums = new int[count];

        for(int i = 0;i < count;i ++){
            nums[i] = in.nextInt();
        }
        // 注意 hasNext 和 hasNextLine 的区别
        // while (in.hasNextLine()) { // 注意 while 处理多个 case
            // int count = Integer.parseInt(in.nextLine());
            // String[] numString = in.nextLine().split(" ");
            // int[] nums = new int[count];
            // for(int i = 0;i < count;i ++){
            //         nums[i] = Integer.parseInt(numString[i]);
            // }
            Node root = new Node();
            root.value = nums[0];

            for(int i = 1;i < count;i ++){
                insert(root,nums[i]);
            }

            Queue<Node> queue = new LinkedList<>();
            queue.add(root);
            int height = 0;
            while(!queue.isEmpty()){
                int size = queue.size();
                for(int i = 0;i < size;i ++){
                    Node node = queue.poll();
                    if(node.left != null){
                        queue.add(node.left);
                    }
                    if(node.middle != null){
                        queue.add(node.middle);
                    }
                    if(node.right != null){
                        queue.add(node.right);
                    }
                }
                height ++;
            }
```

```
                System.out.println(height);
            // }
        }

        public static void insert(Node node,int value){
            if(node.value - value > 500){
                if(node.left == null){
                    node.left = new Node();
                    node.left.value = value;
                }else{
                    insert(node.left,value);
                }
            }else if(node.value - value < -500){
                if(node.right == null){
                    node.right = new Node();
                    node.right.value = value;
                }else{
                    insert(node.right,value);
                }
            }else{
                if(node.middle == null){
                    node.middle = new Node();
                    node.middle.value = value;
                }else{
                    insert(node.middle,value);
                }
            }
        }
}

class Node{
        int value;
        Node left;
        Node right;
        Node middle;
}
```