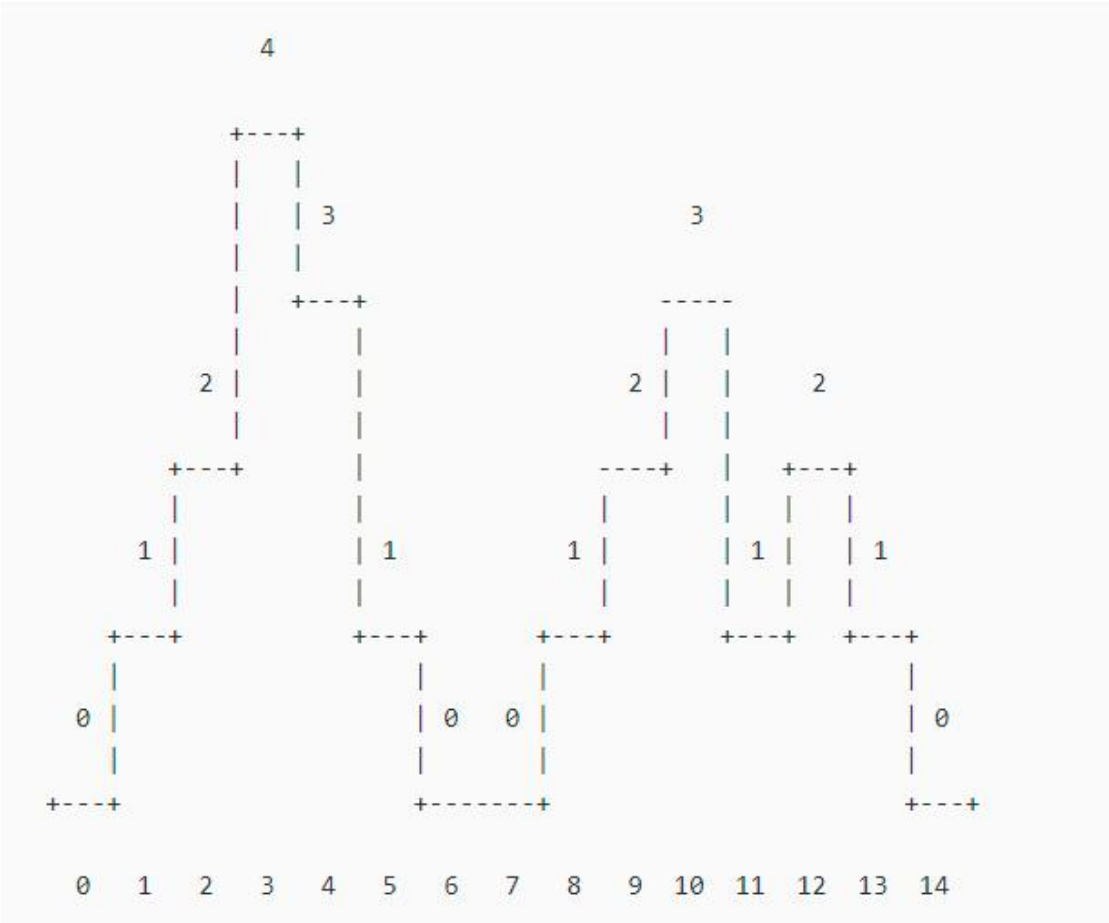
题目描述：

攀登者喜欢寻找各种地图，并且尝试攀登到最高的山峰。

地图表示为一维数组，数组的索引代表水平位置，数组的高度代表相对海拔高度。其中数组元素 `0` 代表地面。

例如 `[0,1,2,4,3,1,0,0,1,2,3,1,2,1,0]`, 代表如下图所示的地图，地图中有两个山脉位置分别为 `1,2,3,4,5` 和 `8,9,10,11,12,13`，最高峰高度分别为 `4,3`。最高峰位置分别为 `3,10`。

一个山脉可能有多座山峰(高度大于相邻位置的高度，或在地图边界且高度大于相邻的高度)。



登山时会消耗登山者的体力(整数)，上山时，消耗相邻高度差两倍的体力，下坡时消耗相邻高度差一倍的体力，平地不消耗体力，登山者体力消耗到零时会有生命危险。

例如，上图所示的山峰，从索引0，走到索引1，高度差为1，需要消耗2X1=2的体力，从索引2高度2走到高度4索引3需要消耗2X2=4的体力。如果是从索引3走到索引4则消耗1X1=1的体力。

登山者想要知道一张地图中有多少座山峰

示例1

输入：

```
[0,1,4,3,1,0,0,1,2,3,1,2,1,0]
```

输出：

```
3
```

说明：

山峰所在的索引分别为 2,10,12

```
import java.util.*;


public class Solution {
    /**
     * 返回地图中山峰的数量
     * @param hill_map int 整型一维数组 地图数组(长度大于 1)
     * @return int 整型
     */
    public int count_peaks (int[] hill_map) {
        // write code here
        if (hill_map.length < 1) {
            return 0;
        } else if (hill_map.length == 1) {
            return hill_map[0] > 0 ? 1 : 0;
        }

        int count = 0;

        for (int i = 0; i < hill_map.length; i++) {
            if (0 == i) {
                count = hill_map[0] > hill_map[1] ? 1 : 0;
```

```
                } else if (hill_map.length - 1 == i) {
                        count = hill_map[hill_map.length - 1] > hill_map[hill_map.length - 2] ? count
+ 1 : count;
                } else {
                        count = hill_map[i] > hill_map[i-1] && hill_map[i] > hill_map[i+1] ? count + 1 :
count;
                }
            }

            return count;
        }
}
```