【华为OD机考 统一考试机试C卷】 山脉的个数/攀登者1（ C++ Java JavaScript python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

2023年11月份，华为官方已经将 华为OD机考：OD统一考试（A卷 / B卷）切换到 OD统一考试（C卷）和 OD统一考试（D卷） 。
真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境

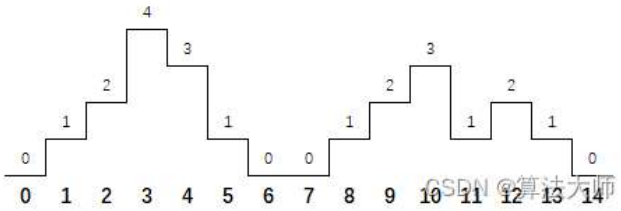
题目描述:山脉的个数(本题分值100)

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，数组的元素代表相对海拔高度。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)。
登山者想要知道一张地图中有多少座山峰。



输入描述

输入为一个整型数组，数组长度大于1。

输出描述

输出地图中山峰的数量。

用例1

输入

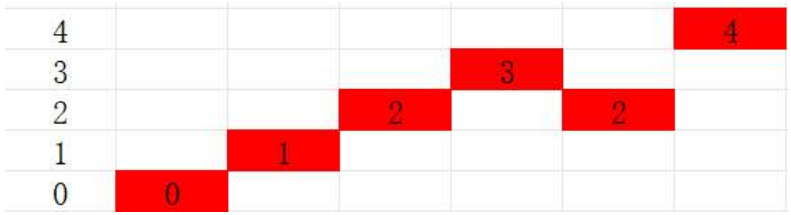
```
0, 1, 2, 3, 2, 4
```

输出

```
2
```

说明：

元素3和4 都是山峰，输出2.



用例2

输入

```
0,1,4,3,1,0,0,1,2,3,1,2,1,0
```

输出

```
3
```

说明 山峰所在索引分别为3，10，12

解题思路

如果当前元素是数组的第一个元素，并且大于下一个元素，或者是数组的最后一个元素，并且大于前一个元素，或者既不是第一个也不是最后一个元素，但大于前一个元素且大于后一个元素，则将计数器 `count` 加一。

C语言

```c
int count_peaks(int hill_map[] ){
    int size = sizeof(hill_map)  ; // 计算数组的长度
    int count = 0; // 初始化计数器为 0
    for(int i = 0; i < size; i++){ // 遍历数组 hill_map
        if(i == 0 && hill_map[i] > hill_map[i + 1]){ // 如果当前位置在数组的开头，并且当前元素大于下一个元素
            count++; // 计数器加一
        }
        if(i == size - 1 && hill_map[i] > hill_map[i - 1]){ // 如果当前位置在数组的末尾，并且当前元素大于前一个元素
            count++; // 计数器加一
```

```
        }
        if(i > 0 && i < size - 1 && hill_map[i] > hill_map[i - 1] && hill_map[i] > hill_map[i + 1]){ // 如果当前位置不在开头和末尾，并且当前元素大于前一个元素且大于后一个元素
            count++; // 计数器加一
        }
    }
    return count; // 返回计数器的值作为结果
}
```

C++

```

// 计算给定hill_map中峰值的函数
int count_peaks(std::vector<int> hill_map) {
    int count = 0; // 初始化计数器为0

    // 遍历hill_map数组
    for(int i = 0; i < hill_map.size(); i++) {

        // 如果当前位置在数组的开始位置，并且当前元素大于下一个元素
        if(i == 0 && hill_map[i] > hill_map[i+1]) {
            count++; // 计数器加一
        }

        // 如果当前位置在数组的结束位置，并且当前元素大于前一个元素
        if(i == hill_map.size()-1 && hill_map[i] > hill_map[i-1]) {
            count++; // 计数器加一
        }

        // 如果当前位置不在数组的开始或结束位置，并且当前元素大于前一个元素和下一个元素
        if(i > 0 && i < hill_map.size()-1 && hill_map[i] > hill_map[i-1] && hill_map[i] > hill_map[i+1]) {
            count++; // 计数器加一
        }
    }

    // 返回计数器的值作为结果
    return count;
}


```

Java

```java
public static int count_peaks(int[] hill_map){
    int count = 0; // 初始化计数器为 0
    for(int i = 0; i < hill_map.length; i++){ // 遍历数组 hill_map
        if(i == 0 && hill_map[i] > hill_map[i+1]){ // 如果当前位置在数组的开头，并且当前元素大于下一个元素
            count++; // 计数器加一
        }
        if(i == hill_map.length-1 && hill_map[i] > hill_map[i-1]){ // 如果当前位置在数组的末尾，并且当前元素大于前一个元素
            count++; // 计数器加一
        }
        if(i > 0 && i < hill_map.length-1 && hill_map[i] > hill_map[i-1] && hill_map[i] > hill_map[i+1]){ // 如果当前位置不在开头和末尾，并且当前元素大于前一个元素且大于后一个元素
            count++; // 计数器加一
        }
    }
    return count; // 返回计数器的值作为结果
}

```

JavaScript

```javascript
function count_peaks(hill_map) {
    let count = 0; // 初始化计数器为 0
    for(let i = 0; i < hill_map.length; i++){ // 遍历数组 hill_map
        if(i === 0 && hill_map[i] > hill_map[i+1]){ // 如果当前位置在数组的开头，并且当前元素大于下一个元素
            count++; // 计数器加一
        }
        if(i === hill_map.length-1 && hill_map[i] > hill_map[i-1]){ // 如果当前位置在数组的末尾，并且当前元素大于前一个元素
            count++; // 计数器加一
        }
        if(i > 0 && i < hill_map.length-1 && hill_map[i] > hill_map[i-1] && hill_map[i] > hill_map[i+1]){ // 如果当前位置不在开头和末尾，并且当前元素大于前一个元素且大于后一个元素
            count++; // 计数器加一
        }
    }
    return count; // 返回计数器的值作为结果
}

```

Python

```python
def count_peaks(hill_map):
    count = 0  # 初始化计数器为 0
    for i in range(len(hill_map)):  # 遍历数组 hill_map
        if i == 0 and hill_map[i] > hill_map[i + 1]:  # 如果当前位置在数组的开头，并且当前元素大于下一个元素
            count += 1  # 计数器加一
        if i == len(hill_map) - 1 and hill_map[i] > hill_map[i - 1]:  # 如果当前位置在数组的末尾，并且当前元素大于前一个元素
            count += 1  # 计数器加一

```

```
        if i > 0 and i < len(hill_map) - 1 and hill_map[i] > hill_map[i - 1] and hill_map[i] > hill_map[i + 1]:  # 如果当前位置不在开头和末尾，并且当前元素大于前一个元素且大于后一个元素
            count += 1  # 计数器加一
    return count  # 返回计数器的值作为结果
```

完整用例

用例1

0, 1, 2, 3, 2, 1, 0, 4, 3, 0

用例2

4, 1, 2, 3, 0

用例3

0, 1, 2, 3, 4

用例4

0, 0, 0, 0, 0

用例5

2, 2, 2, 2, 2

用例6

0, 1, 0, 2, 0, 3, 0

用例7

0, 1, 2, 5, 2, 1, 0

用例8

0, 1, 2, 2, 1, 0

用例9

0, 2, 0, 2, 1, 2, 0, 2, 0

用例10

0, 2, 0, 2, 1, 2, 0, 2, 2

文章目录

机考C卷真题
华为OD