845. Longest Mountain in Array

link

开始想法, 存i位置比i小的数字和比i大的数字.

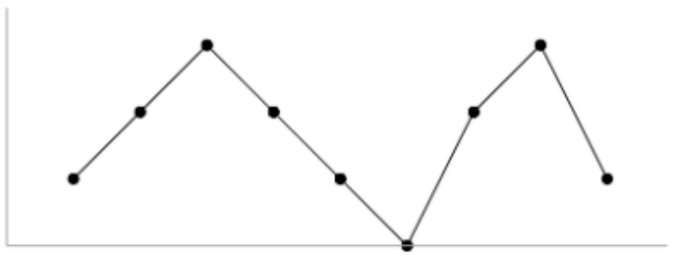
```
public int longestMountain(int[] arr) {
    int max = 0;
    Map<Integer, Integer> left = new HashMap<>();
    Map<Integer, Integer> right = new HashMap<>();
    left.put(0, 0);
    right.put(arr.length - 1, 0);
    for(int i = 1; i < arr.length; ++i){
        if(arr[i] > arr[i - 1]){
            left.put(i, left.get(i - 1) + 1);
        }else{
            left.put(i, 0);
        }
    }
    for(int i = arr.length - 1 - 1; i >= 0; --i){
        if(arr[i] > arr[i + 1])
            right.put(i, right.get(i + 1) + 1);
        else
            right.put(i, 0);

    }
    for(int i = 1; i < arr.length - 1; ++i){
        int leftCount = left.get(i);
        int rightCount = right.get(i);
        if(leftCount > 0 && rightCount > 0){
            max = Math.max(max, leftCount + rightCount + 1);
        }
    }
    return max;
}
```

实际上一个区间的开始只能在另外一个区间结束 space(O1)就可以了

Example

Here is a worked example on the array `A = [1, 2, 3, 2, 1, 0, 2, 3, 1]`:



```java
public int longestMountain(int[] A) {
    int res = 0, up = 0, down = 0;
    for (int i = 1; i < A.length; ++i) {
        if (down > 0 && A[i - 1] < A[i] || A[i - 1] == A[i]) up = down = 0;
        if (A[i - 1] < A[i]) up++;
        if (A[i - 1] > A[i]) down++;
        if (up > 0 && down > 0 && up + down + 1 > res) res = up + down + 1;
    }
    return res;
}
```