There are N blocks, numbered from 0 to N-1, arranged in a row. A couple of frogs were sitting together on one block when they had a terrible quarrel. Now they want to jump away from one another so that the distance between them will be as large as possible. The distance between blocks numbered J and K, where $J \leq K$, is computed as $K - J + 1$. The frogs can only jump up, meaning that they can move from one block to another only if the two blocks are adjacent and the second block is of the same or greater height as the first. What is the longest distance that they can possibly create between each other, if they also chose to sit on the optimal starting block initially?

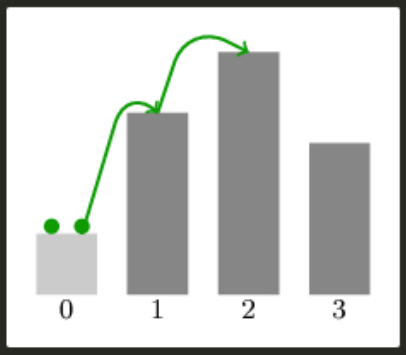
Write a function:

```
function solution(blocks);
```

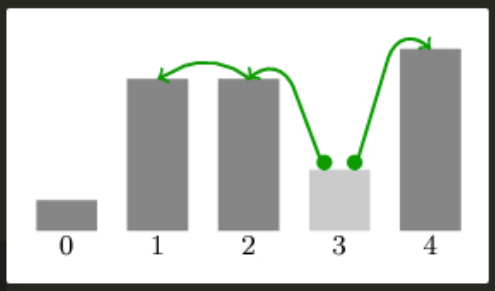
that, given an array `blocks` consisting of N integers denoting the heights of the blocks, returns the longest possible distance that two frogs can make between each other starting from one of the blocks.

Examples:

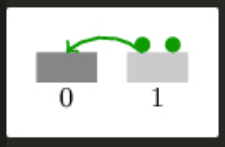
1. Given blocks = [2, 6, 8, 5], the function should return 3. If starting from blocks[0], the first frog can stay where it is and the second frog can jump to blocks[2] (but not to blocks[3]).



2. Given blocks = [1, 5, 5, 2, 6], the function should return 4. If starting from blocks[3], the first frog can jump to blocks[1], but not blocks[0], and the second frog can jump to blocks[4].



3. Given blocks = [1, 1], the function should return 2. If starting from blocks[1], the first frog can jump to blocks[0] and the second frog can stay where it is. Starting from blocks[0] would result in the same distance.



Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [2..200,000];
- each element of array `blocks` is an integer within the range [1..1,000,000,000].