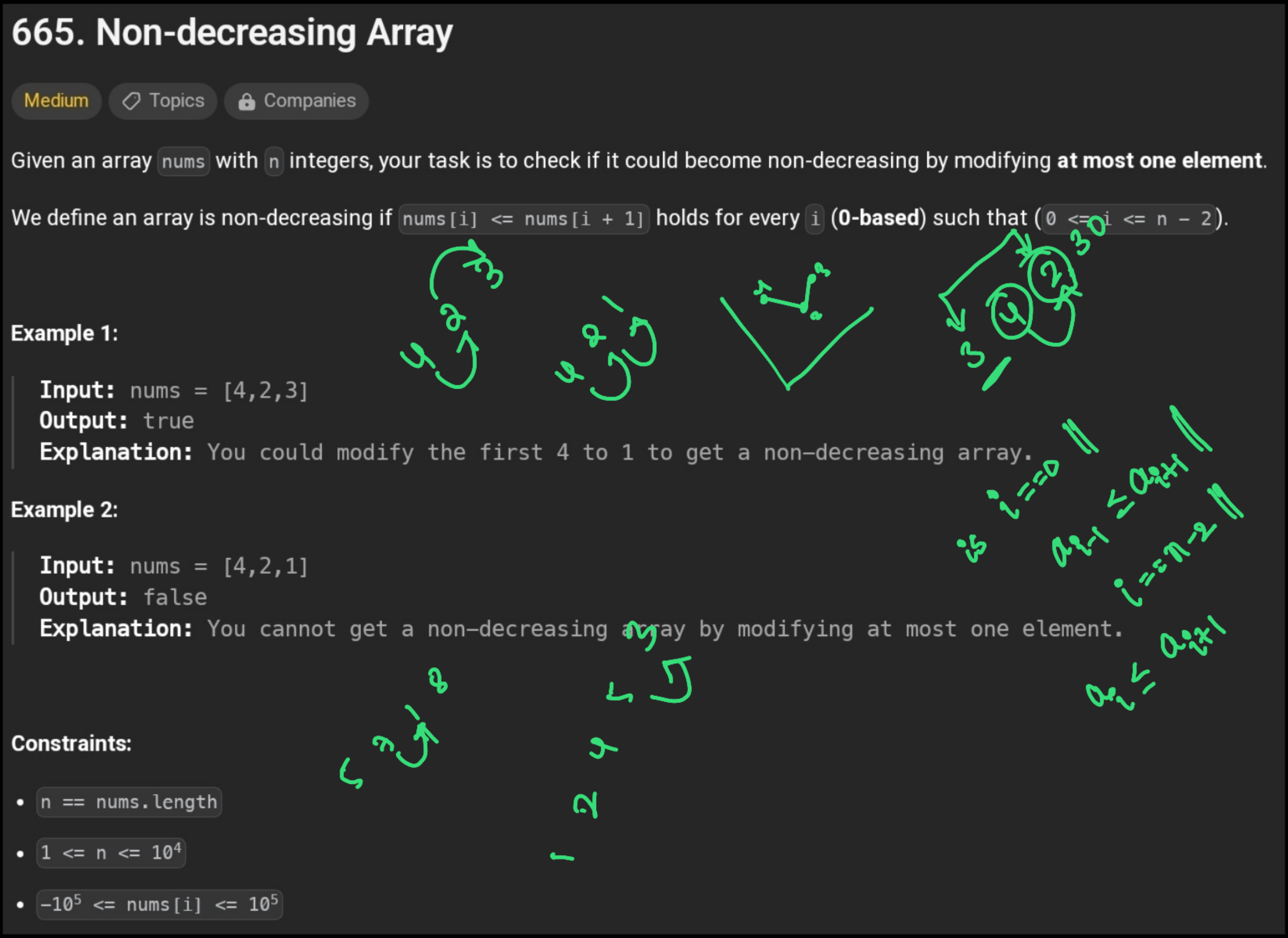


665. Non-decreasing Array

Medium · Topics · Companies

Given an array `nums` with `n` integers, your task is to check if it could become non-decreasing by modifying **at most one element**.

We define an array is non-decreasing if `nums[i] <= nums[i + 1]` holds for every `i` (**0-based**) such that (`0 <= i <= n - 2`).

Example 1:

```
Input: nums = [4,2,3]
Output: true
Explanation: You could modify the first 4 to 1 to get a non-decreasing array.
```

Example 2:

```
Input: nums = [4,2,1]
Output: false
Explanation: You cannot get a non-decreasing array by modifying at most one element.
```

Constraints:

- `n == nums.length`
- `1 <= n <= 10^4`
- `-10^5 <= nums[i] <= 10^5`

This problem is not simple as it seems.
we can come up with the solution by analyzing this example.

$$[\,3\ \ 4\ \ 2\ \ 3\,]$$

↑ here there is a dip

now there are 2 options
either we could replace 4 with 2
or we could replace 2 with 4

$[\,3\ \ 2\ \ 2\ \ 3\,]$
(or)
$[\,3\ \ 4\ \ 4\ \ 3\,]$

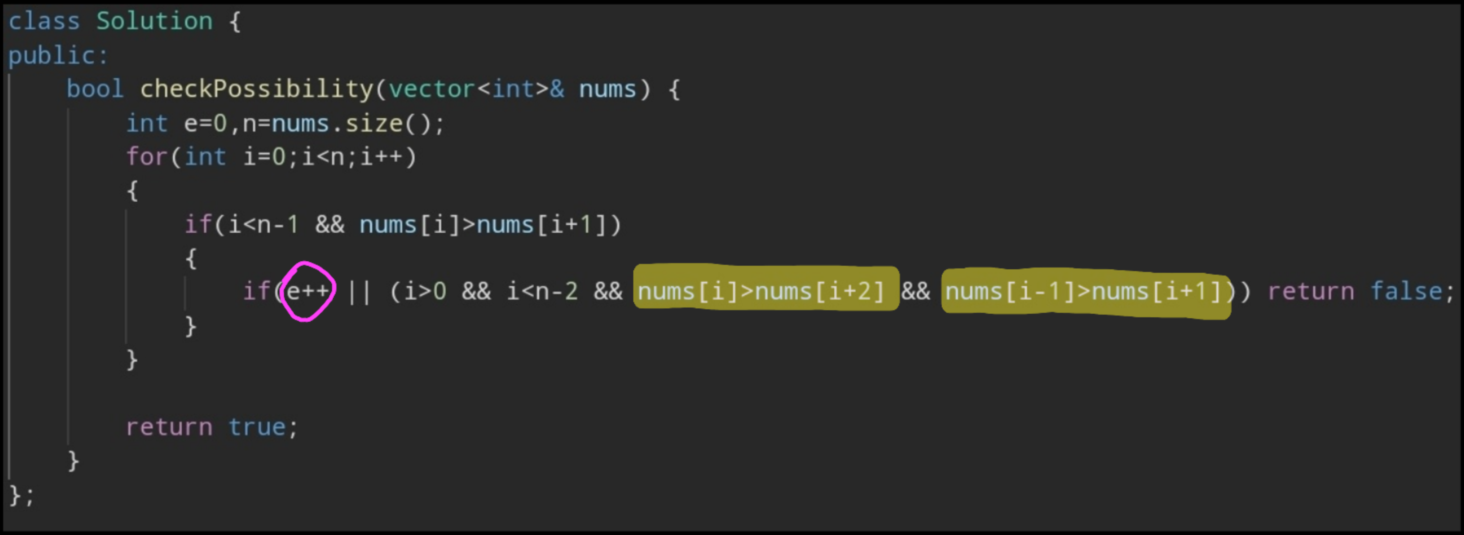
still not a non decreasing arrays

what if array is

[3 4 2 5]
    ↑
Replace 2 with 4
[3 4 4 5]

[1 4 2 3]
    ↑
Replace 4 with 2
[1 2 2 3]

So if any one of the above two is possible at a index ? then its fine to modify one element at that index so we can check on next indices.
If both are not possible at a index then it means even after modifying one element, still it cannot become a non decreasing array.



```
class Solution {
public:
    bool checkPossibility(vector<int>& nums) {
        int e=0,n=nums.size();
        for(int i=0;i<n;i++)
        {
            if(i<n-1 && nums[i]>nums[i+1])
            {
                if(e++ || (i>0 && i<n-2 && nums[i]>nums[i+2] && nums[i-1]>nums[i+1])) return false;
            }
        }

        return true;
    }
};
```

T(n): O(n)
S(n): O(1)