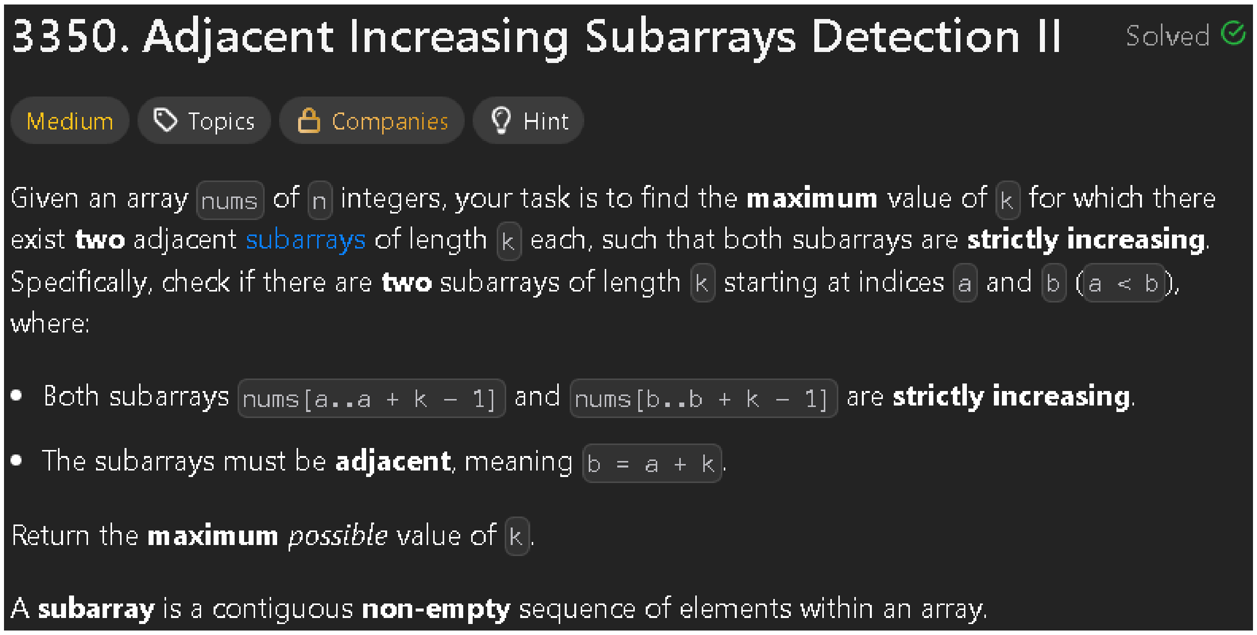


3350. Adjacent Increasing Subarrays Detection II

Solved

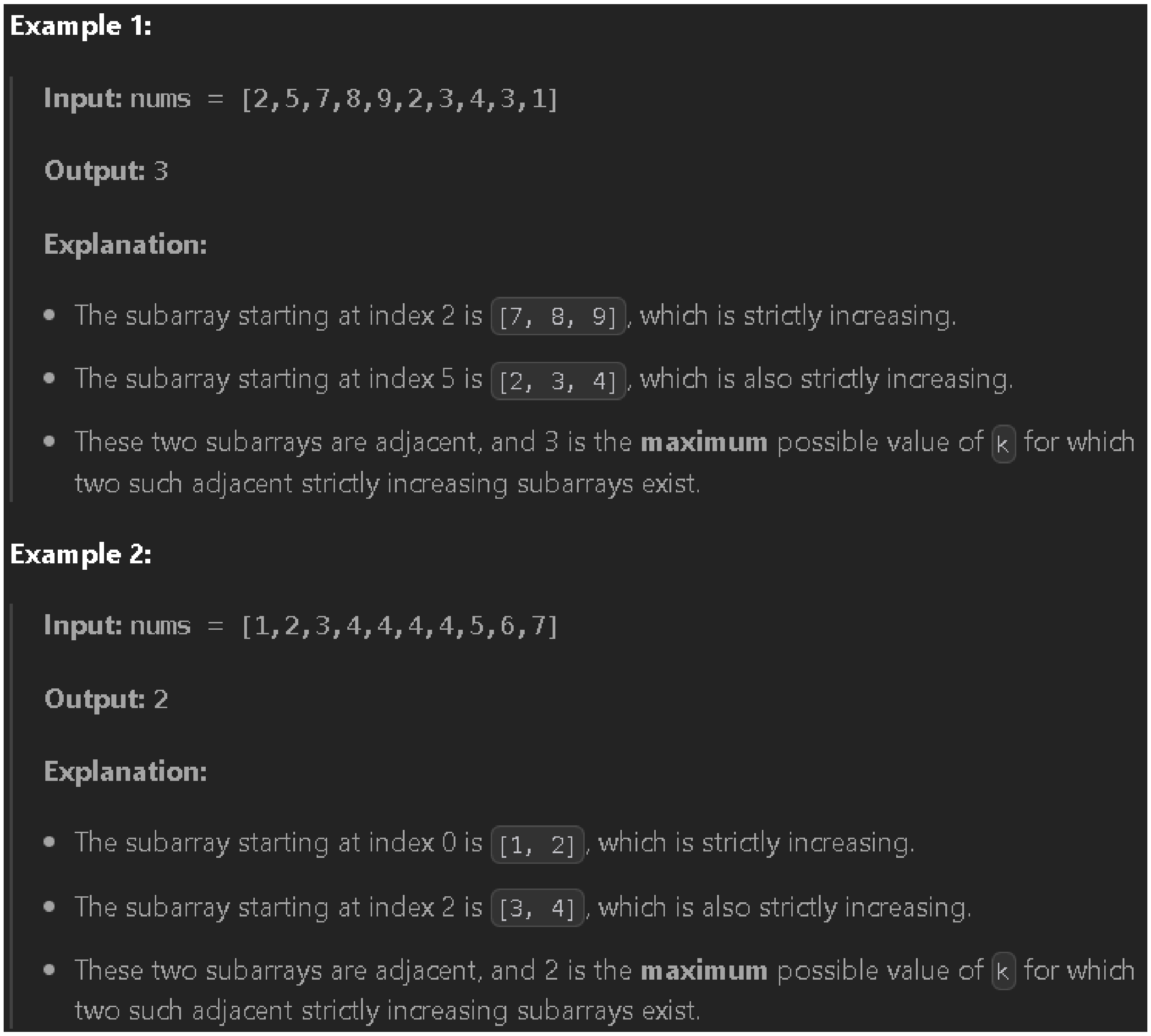
Medium   Topics   Companies   Hint

Given an array `nums` of `n` integers, your task is to find the **maximum** value of `k` for which there exist **two** adjacent subarrays of length `k` each, such that both subarrays are **strictly increasing**. Specifically, check if there are **two** subarrays of length `k` starting at indices `a` and `b` (`a < b`), where:

- Both subarrays `nums[a..a + k - 1]` and `nums[b..b + k - 1]` are **strictly increasing**.
- The subarrays must be **adjacent**, meaning `b = a + k`.

Return the **maximum** *possible* value of `k`.

A **subarray** is a contiguous **non-empty** sequence of elements within an array.


Example 1:

> **Input:** nums = [2,5,7,8,9,2,3,4,3,1]
>
> **Output:** 3
>
> **Explanation:**
>
> - The subarray starting at index 2 is `[7, 8, 9]`, which is strictly increasing.
> - The subarray starting at index 5 is `[2, 3, 4]`, which is also strictly increasing.
> - These two subarrays are adjacent, and 3 is the **maximum** possible value of `k` for which two such adjacent strictly increasing subarrays exist.

Example 2:

> **Input:** nums = [1,2,3,4,4,4,4,5,6,7]
>
> **Output:** 2
>
> **Explanation:**
>
> - The subarray starting at index 0 is `[1, 2]`, which is strictly increasing.
> - The subarray starting at index 2 is `[3, 4]`, which is also strictly increasing.
> - These two subarrays are adjacent, and 2 is the **maximum** possible value of `k` for which two such adjacent strictly increasing subarrays exist.

Python:

```
class Solution:
    def maxIncreasingSubarrays(self, nums: List[int]) -> int:
        n, Len, prev, k=len(nums), 1, 0, 0
        for i in range(1, n):
            if nums[i]>nums[i-1]: Len+=1
            else:
                k=max(k, Len//2, min(Len, prev))
                prev=Len
                Len=1
        return max(k, Len//2, min(Len, prev))
```

JavaScript:

```
var maxIncreasingSubarrays = function(nums) {
    let n = nums.length, up = 1, preUp = 0, res = 0;
    for (let i = 1; i < n; i++) {
        if (nums[i] > nums[i-1]) up++;
        else {
            preUp = up;
            up = 1;
        }
        let half = up >> 1;
        let m = Math.min(preUp, up);
        let candidate = Math.max(half, m);
        if (candidate > res) res = candidate;
    }
    return res;
};
```

Java:

```
class Solution {
    public int maxIncreasingSubarrays(List<Integer> nums) {
        int n = nums.size();
        int up = 1, preUp = 0, res = 0;
        for (int i = 1; i < n; i++) {
            if (nums.get(i) > nums.get(i - 1)) {
                up++;
            } else {
                preUp = up;
                up = 1;
            }
            int half = up >> 1;
            int min = preUp < up ? preUp : up;
            int candidate = half > min ? half : min;
            if (candidate > res) res = candidate;
        }
        return res;
    }
}
```