

Problem 2832: Maximal Range That Each Element Is Maximum in It

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given a

0-indexed

array

nums

of

distinct

integers.

Let us define a

0-indexed

array

ans

of the same length as

nums

in the following way:

`ans[i]`

is the

maximum

length of a subarray

`nums[l..r]`

, such that the maximum element in that subarray is equal to

`nums[i]`

Return

the array

`ans`

Note

that a

subarray

is a contiguous part of the array.

Example 1:

Input:

nums = [1,5,4,3,6]

Output:

[1,4,2,1,5]

Explanation:

For nums[0] the longest subarray in which 1 is the maximum is nums[0..0] so ans[0] = 1. For nums[1] the longest subarray in which 5 is the maximum is nums[0..3] so ans[1] = 4. For nums[2] the longest subarray in which 4 is the maximum is nums[2..3] so ans[2] = 2. For nums[3] the longest subarray in which 3 is the maximum is nums[3..3] so ans[3] = 1. For nums[4] the longest subarray in which 6 is the maximum is nums[0..4] so ans[4] = 5.

Example 2:

Input:

nums = [1,2,3,4,5]

Output:

[1,2,3,4,5]

Explanation:

For nums[i] the longest subarray in which it's the maximum is nums[0..i] so ans[i] = i + 1.

Constraints:

$1 \leq \text{nums.length} \leq 10$

5

$1 \leq \text{nums}[i] \leq 10$

5

All elements in

nums

are distinct.

Code Snippets

C++:

```
class Solution {  
public:  
    vector<int> maximumLengthOfRanges(vector<int>& nums) {  
        }  
    };
```

Java:

```
class Solution {  
public int[] maximumLengthOfRanges(int[] nums) {  
    }  
}
```

Python3:

```
class Solution:  
    def maximumLengthOfRanges(self, nums: List[int]) -> List[int]:
```

Python:

```
class Solution(object):  
    def maximumLengthOfRanges(self, nums):  
        """  
        :type nums: List[int]  
        :rtype: List[int]  
        """
```

JavaScript:

```
/**  
 * @param {number[]} nums
```

```
* @return {number[]}
*/
var maximumLengthOfRanges = function(nums) {
};

}
```

TypeScript:

```
function maximumLengthOfRanges(nums: number[]): number[] {
};

}
```

C#:

```
public class Solution {
public int[] MaximumLengthOfRanges(int[] nums) {
}

}
```

C:

```
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* maximumLengthOfRanges(int* nums, int numsSize, int* returnSize) {
}

}
```

Go:

```
func maximumLengthOfRanges(nums []int) []int {
}
```

Kotlin:

```
class Solution {
fun maximumLengthOfRanges(nums: IntArray): IntArray {
}

}
```

Swift:

```
class Solution {  
    func maximumLengthOfRanges(_ nums: [Int]) -> [Int] {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn maximum_length_of_ranges(nums: Vec<i32>) -> Vec<i32> {  
  
    }  
}
```

Ruby:

```
# @param {Integer[]} nums  
# @return {Integer[]}  
def maximum_length_of_ranges(nums)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param Integer[] $nums  
     * @return Integer[]  
     */  
    function maximumLengthOfRanges($nums) {  
  
    }  
}
```

Dart:

```
class Solution {  
    List<int> maximumLengthOfRanges(List<int> nums) {  
  
    }
```

```
}
```

Scala:

```
object Solution {  
    def maximumLengthOfRanges(nums: Array[Int]): Array[Int] = {  
  
    }  
}
```

Elixir:

```
defmodule Solution do  
  @spec maximum_length_of_ranges(nums :: [integer]) :: [integer]  
  def maximum_length_of_ranges(nums) do  
  
  end  
end
```

Erlang:

```
-spec maximum_length_of_ranges(Nums :: [integer()]) -> [integer()].  
maximum_length_of_ranges(Nums) ->  
.
```

Racket:

```
(define/contract (maximum-length-of-ranges nums)  
  (-> (listof exact-integer?) (listof exact-integer?))  
)
```

Solutions

C++ Solution:

```
/*  
 * Problem: Maximal Range That Each Element Is Maximum in It  
 * Difficulty: Medium  
 * Tags: array, stack  
 */
```

```

* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
vector<int> maximumLengthOfRanges(vector<int>& nums) {
    }

} ;

```

Java Solution:

```

/**
 * Problem: Maximal Range That Each Element Is Maximum in It
 * Difficulty: Medium
 * Tags: array, stack
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
public int[] maximumLengthOfRanges(int[] nums) {
    }

}

```

Python3 Solution:

```

"""
Problem: Maximal Range That Each Element Is Maximum in It
Difficulty: Medium
Tags: array, stack

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

```

```
class Solution:

def maximumLengthOfRanges(self, nums: List[int]) -> List[int]:
    # TODO: Implement optimized solution
    pass
```

Python Solution:

```
class Solution(object):

def maximumLengthOfRanges(self, nums):
    """
    :type nums: List[int]
    :rtype: List[int]
    """
```

JavaScript Solution:

```
/**
 * Problem: Maximal Range That Each Element Is Maximum in It
 * Difficulty: Medium
 * Tags: array, stack
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * @param {number[]} nums
 * @return {number[]}
 */
var maximumLengthOfRanges = function(nums) {

};
```

TypeScript Solution:

```
/**
 * Problem: Maximal Range That Each Element Is Maximum in It
 * Difficulty: Medium
 * Tags: array, stack
```

```

/*
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

function maximumLengthOfRanges(nums: number[]): number[] {
}

```

C# Solution:

```

/*
 * Problem: Maximal Range That Each Element Is Maximum in It
 * Difficulty: Medium
 * Tags: array, stack
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

public class Solution {
    public int[] MaximumLengthOfRanges(int[] nums) {
        return new int[0];
    }
}

```

C Solution:

```

/*
 * Problem: Maximal Range That Each Element Is Maximum in It
 * Difficulty: Medium
 * Tags: array, stack
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

/***

```

```
* Note: The returned array must be malloced, assume caller calls free().  
*/  
int* maximumLengthOfRanges(int* nums, int numSize, int* returnSize) {  
  
}
```

Go Solution:

```
// Problem: Maximal Range That Each Element Is Maximum in It  
// Difficulty: Medium  
// Tags: array, stack  
  
// Approach: Use two pointers or sliding window technique  
// Time Complexity: O(n) or O(n log n)  
// Space Complexity: O(1) to O(n) depending on approach  
  
func maximumLengthOfRanges(nums []int) []int {  
  
}
```

Kotlin Solution:

```
class Solution {  
    fun maximumLengthOfRanges(nums: IntArray): IntArray {  
  
    }  
}
```

Swift Solution:

```
class Solution {  
    func maximumLengthOfRanges(_ nums: [Int]) -> [Int] {  
  
    }  
}
```

Rust Solution:

```
// Problem: Maximal Range That Each Element Is Maximum in It  
// Difficulty: Medium  
// Tags: array, stack
```

```

// 
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn maximum_length_of_ranges(nums: Vec<i32>) -> Vec<i32> {

}
}

```

Ruby Solution:

```

# @param {Integer[]} nums
# @return {Integer[]}
def maximum_length_of_ranges(nums)

end

```

PHP Solution:

```

class Solution {

/**
 * @param Integer[] $nums
 * @return Integer[]
 */
function maximumLengthOfRanges($nums) {

}
}

```

Dart Solution:

```

class Solution {
List<int> maximumLengthOfRanges(List<int> nums) {

}
}

```

Scala Solution:

```
object Solution {  
    def maximumLengthOfRanges(nums: Array[Int]): Array[Int] = {  
        }  
        }  
    }
```

Elixir Solution:

```
defmodule Solution do  
  @spec maximum_length_of_ranges(nums :: [integer]) :: [integer]  
  def maximum_length_of_ranges(nums) do  
  
  end  
  end
```

Erlang Solution:

```
-spec maximum_length_of_ranges(Nums :: [integer()]) -> [integer()].  
maximum_length_of_ranges(Nums) ->  
.
```

Racket Solution:

```
(define/contract (maximum-length-of-ranges nums)  
  (-> (listof exact-integer?) (listof exact-integer?))  
  )
```