Problem 1299: Replace Elements with Greatest Element on Right Side

Problem Information

Difficulty: Easy
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Given an array

arr

, replace every element in that array with the greatest element among the elements to its right, and replace the last element with

-1

.

After doing so, return the array.

Example 1:

Input:

arr = [17,18,5,4,6,1]

Output:

[18,6,6,6,1,-1]

Explanation:

- index 0 --> the greatest element to the right of index 0 is index 1 (18). - index 1 --> the greatest element to the right of index 1 is index 4 (6). - index 2 --> the greatest element to the right of index 2 is index 4 (6). - index 3 --> the greatest element to the right of index 3 is index 4 (6). - index 4 --> the greatest element to the right of index 4 is index 5 (1). - index 5 --> there are no elements to the right of index 5, so we put -1.

Example 2:

Input:

arr = [400]

Output:

[-1]

Explanation:

There are no elements to the right of index 0.

Constraints:

1 <= arr.length <= 10

4

1 <= arr[i] <= 10

5

Code Snippets

C++:

```
class Solution {
public:
vector<int> replaceElements(vector<int>& arr) {

}
```

```
};
```

Java:

```
class Solution {
public int[] replaceElements(int[] arr) {

}
}
```

Python3:

```
class Solution:
def replaceElements(self, arr: List[int]) -> List[int]:
```

Python:

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

JavaScript:

```
/**
* @param {number[]} arr
* @return {number[]}
*/
var replaceElements = function(arr) {

};
```

TypeScript:

```
function replaceElements(arr: number[]): number[] {

};
```

C#:

```
public class Solution {
public int[] ReplaceElements(int[] arr) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* replaceElements(int* arr, int arrSize, int* returnSize) {

}
```

Go:

```
func replaceElements(arr []int) []int {

}
```

Kotlin:

```
class Solution {
fun replaceElements(arr: IntArray): IntArray {

}
}
```

Swift:

```
class Solution {
func replaceElements(_ arr: [Int]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn replace_elements(arr: Vec<i32>) -> Vec<i32> {

}
```

```
}
```

Ruby:

```
# @param {Integer[]} arr
# @return {Integer[]}
def replace_elements(arr)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer[]
*/
function replaceElements($arr) {

}
}
```

Dart:

```
class Solution {
List<int> replaceElements(List<int> arr) {

}
}
```

Scala:

```
object Solution {
def replaceElements(arr: Array[Int]): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec replace_elements(arr :: [integer]) :: [integer]
def replace_elements(arr) do

end
end
```

Erlang:

```
-spec replace_elements(Arr :: [integer()]) -> [integer()].
replace_elements(Arr) ->
.
```

Racket:

```
(define/contract (replace-elements arr)
(-> (listof exact-integer?) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Replace Elements with Greatest Element on Right Side
* Difficulty: Easy
* Tags: array
*
* 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> replaceElements(vector<int>& arr) {

}
};
```

Java Solution:

```
/**
* Problem: Replace Elements with Greatest Element on Right Side
* Difficulty: Easy
* Tags: array
*
* 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[] replaceElements(int[] arr) {

}
}
```

Python3 Solution:

```
"""
Problem: Replace Elements with Greatest Element on Right Side
Difficulty: Easy
Tags: array

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 replaceElements(self, arr: List[int]) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Replace Elements with Greatest Element on Right Side
* Difficulty: Easy
* Tags: array
*
* 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[]} arr
* @return {number[]}
*/
var replaceElements = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Replace Elements with Greatest Element on Right Side
* Difficulty: Easy
* Tags: array
*
* 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 replaceElements(arr: number[]): number[] {

};
```

C# Solution:

```
/*
* Problem: Replace Elements with Greatest Element on Right Side
* Difficulty: Easy
* Tags: array
*
```

```
* 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[] ReplaceElements(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Replace Elements with Greatest Element on Right Side
* Difficulty: Easy
* Tags: array
*
* 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* replaceElements(int* arr, int arrSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Replace Elements with Greatest Element on Right Side
// Difficulty: Easy
// Tags: array
//
// 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 replaceElements(arr []int) []int {
```

```
}
```

Kotlin Solution:

```
class Solution {
fun replaceElements(arr: IntArray): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func replaceElements(_ arr: [Int]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Replace Elements with Greatest Element on Right Side
// Difficulty: Easy
// Tags: array
//
// 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 replace_elements(arr: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @return {Integer[]}
def replace_elements(arr)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer[]
*/
function replaceElements($arr) {

}
}
```

Dart Solution:

```
class Solution {
List<int> replaceElements(List<int> arr) {

}
}
```

Scala Solution:

```
object Solution {
def replaceElements(arr: Array[Int]): Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec replace_elements(arr :: [integer]) :: [integer]
def replace_elements(arr) do

end
end
```

Erlang Solution:

```
-spec replace_elements(Arr :: [integer()]) -> [integer()].
replace_elements(Arr) ->
.
```

Racket Solution:

```
(define/contract (replace-elements arr)
(-> (listof exact-integer?) (listof exact-integer?))
)
```