Problem 739: Daily Temperatures

Problem Information

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

Problem Description

Given an array of integers

temperatures

represents the daily temperatures, return

an array

answer

such that

answer[i]

is the number of days you have to wait after the

i

th

day to get a warmer temperature

. If there is no future day for which this is possible, keep

answer[i] == 0

instead.

Example 1:

Input:

temperatures = [73,74,75,71,69,72,76,73]

Output:

[1,1,4,2,1,1,0,0]

Example 2:

Input:

temperatures = [30,40,50,60]

Output:

[1,1,1,0]

Example 3:

Input:

temperatures = [30,60,90]

Output:

[1,1,0]

Constraints:

1 <= temperatures.length <= 10

5

30 <= temperatures[i] <= 100

Code Snippets

C++:

```
class Solution {
public:
vector<int> dailyTemperatures(vector<int>& temperatures) {

}
};
```

Java:

```
class Solution {
public int[] dailyTemperatures(int[] temperatures) {

}
}
```

Python3:

```
class Solution:
def dailyTemperatures(self, temperatures: List[int]) -> List[int]:
```

Python:

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

JavaScript:

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

```
};
```

TypeScript:

```
function dailyTemperatures(temperatures: number[]): number[] {

};
```

C#:

```
public class Solution {
public int[] DailyTemperatures(int[] temperatures) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* dailyTemperatures(int* temperatures, int temperaturesSize, int*
returnSize) {

}
```

Go:

```
func dailyTemperatures(temperatures []int) []int {

}
```

Kotlin:

```
class Solution {
fun dailyTemperatures(temperatures: IntArray): IntArray {

}
}
```

Swift:

```
class Solution {
func dailyTemperatures(_ temperatures: [Int]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn daily_temperatures(temperatures: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[]} temperatures
# @return {Integer[]}
def daily_temperatures(temperatures)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $temperatures
* @return Integer[]
*/
function dailyTemperatures($temperatures) {

}
}
```

Dart:

```
class Solution {
List<int> dailyTemperatures(List<int> temperatures) {

}
}
```

Scala:

```
object Solution {
def dailyTemperatures(temperatures: Array[Int]): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec daily_temperatures(temperatures :: [integer]) :: [integer]
def daily_temperatures(temperatures) do

end
end
```

Erlang:

```
-spec daily_temperatures(Temperatures :: [integer()]) -> [integer()].
daily_temperatures(Temperatures) ->
.
```

Racket:

```
(define/contract (daily-temperatures temperatures)
(-> (listof exact-integer?) (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Daily Temperatures
* 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> dailyTemperatures(vector<int>& temperatures) {

}
};
```

Java Solution:

```
/**
* Problem: Daily Temperatures
* 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[] dailyTemperatures(int[] temperatures) {

}
}
```

Python3 Solution:

```
"""
Problem: Daily Temperatures
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 dailyTemperatures(self, temperatures: List[int]) -> List[int]:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Daily Temperatures
* 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[]} temperatures
* @return {number[]}
*/
var dailyTemperatures = function(temperatures) {

};
```

TypeScript Solution:

```
/**
* Problem: Daily Temperatures
* 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 dailyTemperatures(temperatures: number[]): number[] {

};
```

C# Solution:

```
/*
* Problem: Daily Temperatures
* 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[] DailyTemperatures(int[] temperatures) {

}
}
```

C Solution:

```
/*
* Problem: Daily Temperatures
* 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* dailyTemperatures(int* temperatures, int temperaturesSize, int*
returnSize) {
```

```
}
```

Go Solution:

```
// Problem: Daily Temperatures
// 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 dailyTemperatures(temperatures []int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun dailyTemperatures(temperatures: IntArray): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func dailyTemperatures(_ temperatures: [Int]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Daily Temperatures
// 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 daily_temperatures(temperatures: Vec<i32>) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} temperatures
# @return {Integer[]}
def daily_temperatures(temperatures)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $temperatures
* @return Integer[]
*/
function dailyTemperatures($temperatures) {

}
}
```

Dart Solution:

```
class Solution {
List<int> dailyTemperatures(List<int> temperatures) {

}
}
```

Scala Solution:

```
object Solution {
def dailyTemperatures(temperatures: Array[Int]): Array[Int] = {
```

```
}
}
```

Elixir Solution:

```
defmodule Solution do
@spec daily_temperatures(temperatures :: [integer]) :: [integer]
def daily_temperatures(temperatures) do

end
end
```

Erlang Solution:

```
-spec daily_temperatures(Temperatures :: [integer()]) -> [integer()].
daily_temperatures(Temperatures) ->
.
```

Racket Solution:

```
(define/contract (daily-temperatures temperatures)
(-> (listof exact-integer?) (listof exact-integer?))
)
```