

Problem 1649: Create Sorted Array through Instructions

Problem Information

Difficulty: Hard

Acceptance Rate: 40.89%

Paid Only: No

Tags: Array, Binary Search, Divide and Conquer, Binary Indexed Tree, Segment Tree, Merge Sort, Ordered Set

Problem Description

Given an integer array `instructions`, you are asked to create a sorted array from the elements in `instructions`. You start with an empty container `nums`. For each element from **left to right** in `instructions`, insert it into `nums`. The **cost** of each insertion is the **minimum** of the following:

* The number of elements currently in `nums` that are **strictly less than** `instructions[i]`. *
The number of elements currently in `nums` that are **strictly greater than** `instructions[i]` .

For example, if inserting element `3` into `nums = [1,2,3,5]` , the **cost** of insertion is `min(2, 1)` (elements `1` and `2` are less than `3` , element `5` is greater than `3`) and `nums` will become `[1,2,3,3,5]` .

Return _the**total cost** to insert all elements from _`instructions` _into_`nums` . Since the answer may be large, return it **modulo** `109 + 7`

Example 1:

Input: instructions = [1,5,6,2] **Output:** 1 **Explanation:** Begin with nums = []. Insert 1 with cost min(0, 0) = 0, now nums = [1]. Insert 5 with cost min(1, 0) = 0, now nums = [1,5]. Insert 6 with cost min(2, 0) = 0, now nums = [1,5,6]. Insert 2 with cost min(1, 2) = 1, now nums = [1,2,5,6]. The total cost is 0 + 0 + 0 + 1 = 1.

Example 2:

****Input:**** instructions = [1,2,3,6,5,4] ****Output:**** 3 ****Explanation:**** Begin with nums = []. Insert 1 with cost min(0, 0) = 0, now nums = [1]. Insert 2 with cost min(1, 0) = 0, now nums = [1,2]. Insert 3 with cost min(2, 0) = 0, now nums = [1,2,3]. Insert 6 with cost min(3, 0) = 0, now nums = [1,2,3,6]. Insert 5 with cost min(3, 1) = 1, now nums = [1,2,3,5,6]. Insert 4 with cost min(3, 2) = 2, now nums = [1,2,3,4,5,6]. The total cost is 0 + 0 + 0 + 0 + 1 + 2 = 3.

****Example 3:****

****Input:**** instructions = [1,3,3,3,2,4,2,1,2] ****Output:**** 4 ****Explanation:**** Begin with nums = []. Insert 1 with cost min(0, 0) = 0, now nums = [1]. Insert 3 with cost min(1, 0) = 0, now nums = [1,3]. Insert 3 with cost min(1, 0) = 0, now nums = [1,3,3]. Insert 3 with cost min(1, 0) = 0, now nums = [1,3,3,3]. Insert 2 with cost min(1, 3) = 1, now nums = [1,2,3,3,3]. Insert 4 with cost min(5, 0) = 0, now nums = [1,2,3,3,3,4]. Insert 2 with cost min(1, 4) = 1, now nums = [1,2,2,3,3,3,4]. Insert 1 with cost min(0, 6) = 0, now nums = [1,1,2,2,3,3,3,4]. Insert 2 with cost min(2, 4) = 2, now nums = [1,1,2,2,2,3,3,3,4]. The total cost is 0 + 0 + 0 + 0 + 1 + 0 + 1 + 0 + 2 = 4.

****Constraints:****

* `1 <= instructions.length <= 105` * `1 <= instructions[i] <= 105`

Code Snippets

C++:

```
class Solution {
public:
    int createSortedArray(vector<int>& instructions) {
        }
};
```

Java:

```
class Solution {
    public int createSortedArray(int[] instructions) {
        }
}
```

Python3:

```
class Solution:  
    def createSortedArray(self, instructions: List[int]) -> int:
```