How Many Numbers Are Smaller Than the Current Number?

Given the array nums, for each nums[i] find out how many numbers in the array are smaller than it. That is, for each nums[i] you have to count the number of valid j's such that j != i and nums[j] < nums[i].

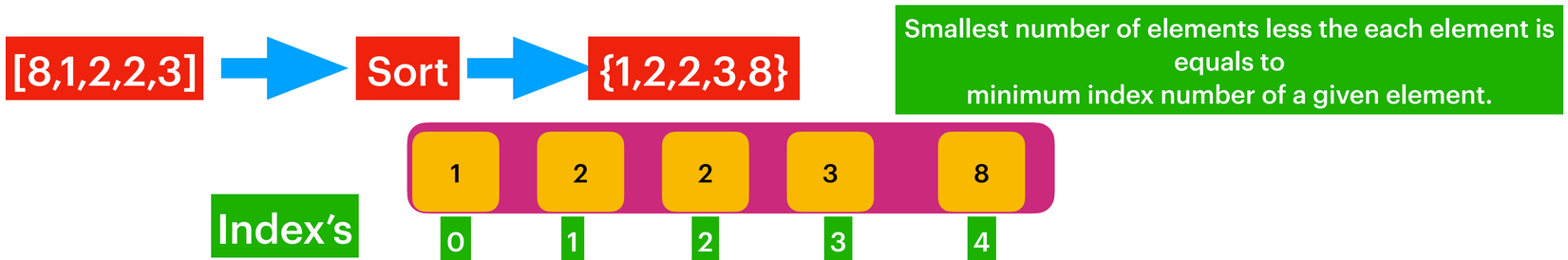
Return the answer in an array.

Input: nums = [8,1,2,2,3]
Output: [4,0,1,1,3]
Explanation:
For nums[0]=8 there exist four smaller numbers than it (1, 2, 2 and 3).
For nums[1]=1 does not exist any smaller number than it.
For nums[2]=2 there exist one smaller number than it (1).
For nums[3]=2 there exist one smaller number than it (1).
For nums[4]=3 there exist three smaller numbers than it (1, 2 and 2).

Input: nums = [6,5,4,8]
Output: [2,1,0,3]

Input: nums = [7,7,7,7]
Output: [0,0,0,0]

Constraints :
2 <= nums.length <= 500
0 <= nums[i] <= 100



inputArr:: [8,1,2,2,3,8,8] expected Output:: [4,0,1,1,3,4,4]

=>Time Complexity:

=> O(n) copy to Output Arr + nlog(n) sort + O(n) lookUpMap + O(n) copyResult to Output Array

=> O(3n) + nlog(n) = nlog(n)

Space Complexity:

=>O(n) copy + O(n) map = O(2n) = O(n)

outPutArr(copy) :: [8,1,2,2,3,8,8]

nlog(n) nums :: [1,2,2,3,8,8,8]

map ::

1 -> 0

2 -> 1

3 -> 3

8 -> 4

Each element is mapped to its smallest index number.

outPutArr(copy) :: [8,1,2,2,3,8,8]

traverse [4,0,1,1,3,4,4]

Design Unique Word Abbreviation

The abbreviation of a word is a concatenation of its first letter, the number of characters between the first and last letter, and its last letter. If a word has only two characters, then it is an abbreviation of itself.

For example:

dog --> d1g because there is one letter between the first letter 'd' and the last letter 'g'.
internationalization --> i18n because there are 18 letters between the first letter 'i' and the last letter 'n'.
it --> it because any word with only two characters is an abbreviation of itself.

Implement the ValidWordAbbr class:

ValidWordAbbr(String[] dictionary) Initializes the object with a dictionary of words.
boolean isUnique(string word) Returns true if either of the following conditions are met (otherwise returns false):
There is no word in dictionary whose abbreviation is equal to word's abbreviation.
For any word in dictionary whose abbreviation is equal to word's abbreviation, that word and word are the same.

Input
["ValidWordAbbr", "isUnique", "isUnique", "isUnique", "isUnique", "isUnique"]
[[["deer", "door", "cake", "card"]], ["dear"], ["cart"], ["cane"], ["make"], ["cake"]]
Output
[null, false, true, false, true, true]

Explanation
ValidWordAbbr validWordAbbr = new ValidWordAbbr(["deer", "door", "cake", "card"]);
validWordAbbr.isUnique("dear"); // return false, dictionary word "deer" and word "dear" have the same abbreviation "d2r" but are not the same.
validWordAbbr.isUnique("cart"); // return true, no words in the dictionary have the abbreviation "c2t".
validWordAbbr.isUnique("cane"); // return false, dictionary word "cake" and word "cane" have the same abbreviation "c2e" but are not the same.
validWordAbbr.isUnique("make"); // return true, no words in the dictionary have the abbreviation "m2e".
validWordAbbr.isUnique("cake"); // return true, because "cake" is already in the dictionary and no other word in the dictionary has "c2e" abbreviation.

Constraints :
1 <= dictionary.length <= 3 * 104
1 <= dictionary[i].length <= 20
dictionary[i] consists of lowercase English letters.
1 <= word.length <= 20
word consists of lowercase English letters.