Description | Editorial | Solutions | Submissions

163. Missing Ranges

Premium

Solved

Easy | Topics | Companies

You are given an inclusive range `[lower, upper]` and a **sorted unique** integer array `nums`, where all elements are within the inclusive range.

A number `x` is considered **missing** if `x` is in the range `[lower, upper]` and `x` is not in `nums`.

Return *the **shortest sorted** list of ranges that **exactly covers all the missing numbers***. That is, no element of `nums` is included in any of the ranges, and each missing number is covered by one of the ranges.

Example 1:

```
Input: nums = [0,1,3,50,75], lower = 0, upper = 99
Output: [[2,2],[4,49],[51,74],[76,99]]
Explanation: The ranges are:
[2,2]
[4,49]
[51,74]
[76,99]
```

Example 2:

```
Input: nums = [-1], lower = -1, upper = -1
Output: []
Explanation: There are no missing ranges since there are no missing numbers.
```

Constraints:

- `-10^9 <= lower <= upper <= 10^9`
- `0 <= nums.length <= 100`
- `lower <= nums[i] <= upper`
- All the values of `nums` are **unique**.

Seen this question in a real interview before? 1/5

Yes | No

Accepted 238.9K | Submissions 709.4K | Acceptance Rate 33.7%

Topics

Companies

Similar Questions

Discussion (28)

1.1K | 28

Code

Python3 • Auto

```python
class Solution:
    def findMissingRanges(self, nums: List[int], lower: int, upper: int) -> List[List[int]]:

        if len(nums) == 0:
            return [[lower, upper]]

        ans = []

        if nums[0]!= lower:
            ans.append([lower, nums[0]-1])

        for i in range(0, len(nums)-1):
            if nums[i]+1 != nums[i+1]:
                ans.append([nums[i]+1, nums[i+1]-1])

        if nums[-1]!= upper:
            ans.append([nums[-1]+1, upper])

        return ans

```

Saved

Ln 8, Col 1

Testcase | Test Result

Accepted Runtime: 60 ms

• Case 1 • Case 2

Input

nums =
[0,1,3,50,75]

lower =
0

upper =
99