Description | Solution | Discuss (785) | Submissions

163. Missing Ranges

Easy 666 2306 Add to List Share

You are given an inclusive range `[lower, upper]` and a **sorted unique** integer array `nums`, where all elements are in 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 **smallest sorted** list of ranges that **cover every missing number exactly***. That is, no element of `nums` is in any of the ranges, and each missing number is in one of the ranges.

Each range `[a,b]` in the list should be output as:

- `"a->b"` if `a != b`
- `"a"` if `a == b`

Example 1:

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

Example 2:

```
Input: nums = [], lower = 1, upper = 1
Output: ["1"]
Explanation: The only missing range is [1,1], which becomes "1".
```

Example 3:

```
Input: nums = [], lower = -3, upper = -1
Output: ["-3->-1"]
Explanation: The only missing range is [-3,-1], which becomes "-3->-1".
```

Example 4:

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

Example 5:

```
Input: nums = [-1], lower = -2, upper = -1
Output: ["-2"]
```

Constraints:

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

Accepted 145,987 | Submissions 486,205

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Facebook | 11 Amazon | 7

Related Topics

Array

Similar Questions

Summary Ranges Easy

Java | Autocomplete

```java
class Solution {
    public List<String> findMissingRanges(int[] nums, int lower, int upper) {
        List<String> result = new ArrayList<>();
        int prev = lower - 1;
        for (int i = 0; i <= nums.length; i++) {
            int curr = (i < nums.length) ? nums[i] : upper + 1;
            if (prev + 1 <= curr - 1) {
                result.add(formatRange(prev + 1, curr - 1));
            }
            prev = curr;
        }
        return result;
    }

    // formats range in the requested format
    private String formatRange(int lower, int upper) {
        if (lower == upper) {
            return String.valueOf(lower);
        }
        return lower + "->" + upper;
    }
}
```