Problem 163: Missing Ranges

Problem Information

Difficulty: Easy
Acceptance Rate: 35.48%
Paid Only: Yes
Tags: Array

Problem Description

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:

* `-109 <= lower <= upper <= 109` * `0 <= nums.length <= 100` * `lower <= nums[i] <= upper` * All the values of `nums` are **unique**.

Code Snippets

C++:

```
class Solution {
public:
vector<vector<int>> findMissingRanges(vector<int>& nums, int lower, int
upper) {

}
};
```

Java:

```
class Solution {
public List<List<Integer>> findMissingRanges(int[] nums, int lower, int
upper) {

}
}
```

Python3:

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