Q1.
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 = [-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**.

Q2.
You are given a **0-indexed** integer array nums.

Swaps of **adjacent** elements are able to be performed on nums.

A **valid** array meets the following conditions:

- The largest element (any of the largest elements if there are multiple) is at the rightmost position in the array.
- The smallest element (any of the smallest elements if there are multiple) is at the leftmost position in the array.

Return the **minimum** swaps required to make nums a valid array.

Example 1:
Input: nums = [3,4,5,5,3,1]
Output: 6

Explanation: Perform the following swaps:
- Swap 1: Swap the 3rd and 4th elements, nums is then [3,4,5,3,5,1].
- Swap 2: Swap the 4th and 5th elements, nums is then [3,4,5,3,1,5].
- Swap 3: Swap the 3rd and 4th elements, nums is then [3,4,5,1,3,5].
- Swap 4: Swap the 2nd and 3rd elements, nums is then [3,4,1,5,3,5].
- Swap 5: Swap the 1st and 2nd elements, nums is then [3,1,4,5,3,5].
- Swap 6: Swap the 0th and 1st elements, nums is then [1,3,4,5,3,5].

It can be shown that 6 swaps is the minimum swaps required to make a valid array.

Example 2:
Input: nums = [9]
Output: 0

Explanation: The array is already valid, so we return 0.

Constraints:

- 1 <= nums.length <= 10^5
- 1 <= nums[i] <= 10^5

Q3.
Given an array of meeting time **intervals** where **intervals[i] = [**$start_i$**,** end_i**,]**, return the minimum number of conference rooms required.

Example 1:
Input: intervals = [[0,30],[5,10],[15,20]]
Output: 2

Example 2:
Input: intervals = [[7,10],[2,4]]
Output: 1

Constraints:

- 1 <= intervals.length <= 10^4
- 0 <= $start_i$< end_i<= 10^6