632. Smallest Range Covering Elements from K Lists

You have `k` lists of sorted integers in **non-decreasing order**. Find the **smallest** range that includes at least one number from each of the `k` lists.

We define the range `[a, b]` is smaller than range `[c, d]` if `b - a < d - c` **or** `a < c` if `b - a == d - c`.

Example 1:

```
Input: nums = [[4,10,15,24,26],[0,9,12,20],[5,18,22,30]]
Output: [20,24]
Explanation:
List 1: [4, 10, 15, 24,26], 24 is in range [20,24].
List 2: [0, 9, 12, 20], 20 is in range [20,24].
List 3: [5, 18, 22, 30], 22 is in range [20,24].
```

Example 2:

```
Input: nums = [[1,2,3],[1,2,3],[1,2,3]]
Output: [1,1]
```

Constraints:

- `nums.length == k`
- `1 <= k <= 3500`
- `1 <= nums[i].length <= 50`
- `-105 <= nums[i][j] <= 10`^{5}
- `nums[i]` is sorted in **non-decreasing** order.

Smallest Range Covering Elements from K Lists

```
/*
   k pointers
   min heap to get the min
   Time complexity: O(nlogk)
   Space complexity: O(k)
   n: total number of elements in all k lists
*/

typedef std::tuple<int, int, int> iii; // min_val,index of list, index of min_val

class Solution {
   public:
      std::vector<int> smallestRange(std::vector<std::vector<int>>& nums){
         int k=nums.size();
         int left=INT_MAX,right=INT_MIN;
         std::priority_queue<iii,std::vector<iii>,std::greater<iii>> min_heap;
         for(int i=0;i<k;++i){
            auto v=nums[i];
            left=std::min(left,v[0]);
            right=std::max(right,v[0]);
            min_heap.push({v[0],i,0});
         }
         std::vector<int> ans={left,right};
         while(!min_heap.empty()){
            auto [mi,list_idx,mi_idx]=min_heap.top();
            min_heap.pop();
            mi_idx++;
            if(mi_idx<nums[list_idx].size()){
               min_heap.push({nums[list_idx][mi_idx],list_idx,mi_idx});
               left=std::get<0>(min_heap.top());
               right=std::max(right,nums[list_idx][mi_idx]);
            }
            else break;
            if(right-left<ans[1]-ans[0]) ans={left,right};
         }
         return ans;
      }
};
```