📌 Problem Statement – Smallest Range Covering Elements from K Lists Leetcode-632

We are given **k sorted lists** of integers in non-decreasing order.

We have to find the **smallest range [a, b]** such that **at least one number from each list** is inside that range.

Comparison rule for ranges:

- `[a,b]` is smaller than `[c,d]` if `(b-a) < (d-c)`
- If `(b-a) == (d-c)`, the smaller range is the one with the smaller `a`.

Example

Input

```
nums = [
  [4,10,15,24,26],
  [0,9,12,20],
  [5,18,22,30]
]
```

Output

```
[20, 24]
```

Constraints

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

💡 Approach (Mera Version)

1. Har list ka **first element** priority queue (min-heap) me daal.
2. Saath me `mx` rakho jo ab tak ka max element store kare.
3. Heap se min element nikalke:
   - Range `[mn, mx]` ka size check karo.
   - Agar naya range chhota hai to `start` aur `end` update karo.
4. Min element ke list ka **next element** heap me daal do.
5. `mx` ko update karo agar naya element bada hai.
6. Jaise hi koi list ka pointer end tak pahunchta hai → break.

🖥️ Your Exact Code

```
class Solution {
public:
    typedef pair<int,pair<int,int>> pel;
    vector<int> smallestRange(vector<vector<int>>& nums) {
        vector<int>ans;
        priority_queue<pel,vector<pel>,greater<pel>>pq;
        int mx = INT_MIN;
        for(int i=0;i<nums.size();i++){
            mx = max(nums[i][0],mx);
            pq.push({nums[i][0],{i,0}});
        }
        int start = pq.top().first;
        int end = mx;
```

```
        while(true){
            int mn = pq.top().first;
            int row = pq.top().second.first;
            int column = pq.top().second.second;
            pq.pop();
            if(mx-mn < end-start){
                start = mn;
                end = mx;
            }
            if(column+1 >= nums[row].size()) break;
            mx = max(nums[row][column+1],mx);
            pq.push({nums[row][column+1],{row,column+1}});

        }
        ans.push_back(start);
        ans.push_back(end);
        return ans;
    }
};
```

📜 Dry Run – Example 1

Input:

```
nums = [
  [4, 10, 15, 24, 26],
  [0, 9, 12, 20],
  [5, 18, 22, 30]
]
```

Step 1: Initialization

Push first element of each list into heap:

```
Heap = [(0,{1,0}), (4,{0,0}), (5,{2,0})]
mx = 5
start = pq.top().first = 0
end = mx = 5
```

Step 2: Iterations

Iteration 1

- mn=0, row=1, col=0
- Range = 5-0 = 5 → No smaller than current (5)
- Push (9,{1,1}), mx=9

Iteration 2

- mn=4, row=0, col=0
- Range = 9-4 = 5 → No update
- Push (10,{0,1}), mx=10

Iteration 3

- mn=5, row=2, col=0
- Range = 10-5 = 5 → No update
- Push (18,{2,1}), mx=18

Iteration 4

- mn=9, row=1, col=1
- Range = 18-9 = 9 → No update
- Push (12,{1,2}), mx=18

Iteration 5

- mn=10, row=0, col=1
- Range = 18-10 = 8 → No update
- Push (15,{0,2}), mx=18

Iteration 6

- mn=12, row=1, col=2
- Range = 18-12 = 6 → No update
- Push (20,{1,3}), mx=20

Iteration 7

- mn=15, row=0, col=2
- Range = 20-15 = 5 → No update
- Push (24,{0,3}), mx=24

Iteration 8

- mn=18, row=2, col=1
- Range = 24-18 = 6 → No update
- Push (22,{2,2}), mx=24

Iteration 9

- mn=20, row=1, col=3
- Range = 24-20 = 4 → Smaller → start=20, end=24 ✅
- No next element in row 1 → break

✅ **Final Answer**:

```
[20, 24]
```
