632. Smallest Range

DescriptionHintsSubmissionsDiscussSolution

DiscussPick One

- Difficulty:**Hard**
- Total Accepted:2K
- Total Submissions:4.6K
- Contributor:fallcree

You have `k` lists of sorted integers in ascending 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:[[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].
```

Note:

1. The given list may contain duplicates, so ascending order means >= here.

2. 1 <= k <= 3500

3. -105 <= value of elements <= 105.

4. **For Java users, please note that the input type has been changed to List<List<Integer>>. And after you reset the code template, you'll see this point.**

```
#include<iostream>
#include<sstream>
#include<stdio.h>
#include<vector>
#include<unordered_set>
#include<unordered_map>
#include<limits.h>
#include<set>
#include<random>
#include<ctime>
#include<stack>
#include<string>
#include<queue>
using namespace std;
typedef struct element
{
        int val;
        int row;
        int idx;
        element(int v, int r, int i)
        {
                val = v;
                row = r;
                idx = i;
        }
}element;

struct comp
{
        bool operator()(element a, element b)
        {
                return a.val>b.val;
        }
};

vector<int> smallestRange(vector<vector<int>>& nums) {
        priority_queue<element, vector<element>, comp> pq;
        int max_val = INT_MIN;
        for(int i=0;i<(int)nums.size();++i)
        {
                element elem(nums[i][0],i,0);
                pq.push(elem);
                max_val = max(max_val,elem.val);
        }
        int range = INT_MAX; int start = -1,end=-1;
        while(pq.size()==nums.size())
        {
                element curr = pq.top();
                pq.pop();
                if(max_val-curr.val<range)
                {
                        start = curr.val;
                        end = max_val;
                        range = max_val-curr.val;
                }
                if(curr.idx+1<(int)nums[curr.row].size())
                {
                        element tmp(nums[curr.row][curr.idx+1],curr.row,curr.idx+1);
                        pq.push(tmp);
                        if(tmp.val>max_val)
                        {
                                max_val = tmp.val;
                        }
                }
        }
        return {start,end};
}
```