2406. Divide Intervals Into Minimum Number of Groups

You are given a 2D integer array `intervals` where `intervals[i] = [lefti, righti]` represents the **inclusive** interval `[lefti, righti]`.

You have to divide the intervals into one or more **groups** such that each interval is in **exactly** one group, and no two intervals that are in the same group **intersect** each other.

Return *the* ***minimum*** *number of groups you need to make*.

Two intervals **intersect** if there is at least one common number between them. For example, the intervals `[1, 5]` and `[5, 8]` intersect.

Example 1:

```
Input: intervals = [[5,10],[6,8],[1,5],[2,3],[1,10]]
Output: 3
Explanation: We can divide the intervals into the following groups:
- Group 1: [1, 5], [6, 8].
- Group 2: [2, 3], [5, 10].
- Group 3: [1, 10].
It can be proven that it is not possible to divide the intervals into fewer than 3
groups.
```

Example 2:

```
Input: intervals = [[1,3],[5,6],[8,10],[11,13]]
Output: 1
Explanation: None of the intervals overlap, so we can put all of them in one group.
```

Constraints:

- `1 <= intervals.length <=` 10^5
- `intervals[i].length == 2`
- `1 <= lefti <= righti <=` 10^6

2406. Divide Intervals Into Minimum Number of Groups

```
/*
    Brute force
    Time complexity: O(nlogn+n^2)=O(n^2)
    Space complexity: O(n)
*/
class Solution {
    public:
        int minGroups(std::vector<std::vector<int>>& intervals){
            int n=intervals.size();

            std::sort(intervals.begin(),intervals.end());

            std::vector<int> groups;

            for(auto& interval: intervals){
                int left=interval[0];
                int right=interval[1];
                int j=0;
                while(j<groups.size()&&groups[j]>=left) j++;
                if(j<groups.size()) groups[j]=right;
                else groups.push_back(right);
            }
            return groups.size();
        }
};
```

2406. Divide Intervals Into Minimum Number of Groups

```
/*
    Line sweep with sorting (map)
    Time complexity: O(nlogn)
    Space complexity: O(n)
*/
class Solution {
    public:
        int minGroups(std::vector<std::vector<int>>& intervals){
            std::map<int,int> dp;
            for(auto& interval: intervals){
                int left=interval[0];
                int right=interval[1];
                dp[left]++;
                dp[right+1]--;
            }

            int ans=0;
            int prefix_sum=0;
            for(auto& [k,v]: dp) {
                prefix_sum+=v;
                ans=std::max(ans,prefix_sum);
            }

            return ans;
        }
};
```

2406. Divide Intervals Into Minimum Number of Groups

```
/*
    Line sweep without sorting (vector)
    Time complexity: O(n+(max_right-min_left+1))
    Space complexity: O(max_right+2)=O(max_right)
*/
class Solution {
    public:
        int minGroups(std::vector<std::vector<int>>& intervals){
            int min_left=INT_MAX;
            int max_right=INT_MIN;
            for(auto& interval: intervals){
                min_left=std::min(min_left,interval[0]);
                max_right=std::max(max_right,interval[1]);
            }

            std::vector<int> dp(max_right+2,0);

            for(auto& interval: intervals){
                int left=interval[0];
                int right=interval[1];
                dp[left]++;
                dp[right+1]--;
            }

            int ans=0;
            int tmp_ans=0;
            for(auto& v: dp) {
                tmp_ans+=v;
                ans=std::max(ans,tmp_ans);
            }

            return ans!=0?ans:1;
        }
};
```