## 435. Non-overlapping Intervals

Medium

Given an array of intervals `intervals` where `intervals[i] = [start_i, end_i]`, return *the minimum number of intervals you need to remove to make the rest of the intervals non-overlapping*.

**Example 1:**

```
Input: intervals = [[1,2],[2,3],[3,4],[1,3]]

Output: 1

Explanation: [1,3] can be removed and the rest of the intervals are non-overlapping.
```

**Example 2:**

```
Input: intervals = [[1,2],[1,2],[1,2]]

Output: 2

Explanation: You need to remove two [1,2] to make the rest of the intervals non-overlapping.
```

**Example 3:**

```
Input: intervals = [[1,2],[2,3]]

Output: 0

Explanation: You don't need to remove any of the intervals since they're already non-overlapping.
```

**Constraints:**

- `1 <= intervals.length <= 10^5`
- `intervals[i].length == 2`
- `-5 * 10^4 <= start_i < end_i <= 5 * 10^4`

Solution:

Let's Take one example:

[1,5] ,[3,7],[10,15]

Here in this example the the [1,5] and [3,7] are overlapping so we have to delete one of then so ans is 1 here.

Suppose Let's Take one more Example

[1,6],[3,9],[7,15]

In this intervlas

1----------6

3----------9

7----------15

In this above example you have to delete intervals to making the the intervals non-overallpping

So here it matters which interval should delete and also given in question that minimum interval

In above example if we delete [1,6] interval then it again [3,6] and [7,15] will overlap.

And in case if we delete [3,9] then other two are non-overlapping

Example 3:

[1,9] ,[3,6] ,[7,15]

In this Case we will prefer to delete first interval over second.

Code:

```
class Solution
{
   public:
      int eraseOverlapIntervals(vector<vector < int>> &intervals)
      {

         sort(intervals.begin(), intervals.end());
         int previous = 0;
         int ans = 0;
         int n = intervals.size();
         for (int i = 1; i < intervals.size(); i++)
         {

            if (intervals[i][0] < intervals[previous][1])
            {
```

```
            ans++;

            if (intervals[i][1] <= intervals[previous][1])
            {
                previous = i;
            }
        }
        else
        {
            previous = i;
        }
    }

    return ans;
  }
};
```