Non-overlapping Intervals

Difficulty: **Medium** Accuracy: **51.92%** Submissions: **43K+** Points: **4**

Average Time: **30m**

Given a 2D array **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**.

Note: Two intervals are considered non-overlapping if the end time of one interval is less than or equal to the start time of the next interval.

Examples:

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.

Input: intervals[][] = [[1, 3], [1, 3], [1, 3]]
Output: 2
Explanation: You need to remove two [1, 3] to make the rest of the

C++ (12) Start Timer

```cpp
class Solution {
  public:
    int minRemoval(vector<vector<int>> &intervals) {
        // code here
        sort(intervals.begin(),intervals.end());
        int n=intervals.size();

        int last=intervals[0][1];

        int c=0;

        for(int i=1;i<n;i++)
        {
            if(intervals[i][0]<last)
            {
                c++;
                last=min(intervals[i][1],last);
            }
            else{last=intervals[i][1];}
        }
        return c;
    }
};
```

Custom Input Compile & Run Sub

Insert Interval

Difficulty: **Medium** Accuracy: **50.61%** Submissions: **55K+** Points: **4**

Average Time: **30m**

Geek has an array of non-overlapping intervals **intervals[][]** where **intervals[i]** = **[start$_i$, end$_i$]** represent the start and the end of the **ith** event and intervals is sorted in **ascending** order by **start$_i$** . He wants to add a new interval **newInterval[]** = **[newStart, newEnd]** where newStart and newEnd represent the start and end of this interval. Help Geek to **insert** newInterval into intervals such that intervals is still sorted in ascending order by **start$_i$** and intervals still does not have any overlapping intervals (merge overlapping intervals if necessary).

Examples:

Input: intervals[][] = [[1, 3], [4, 5], [6, 7], [8, 10]], newInterval[] = [5, 6]
Output: [[1, 3], [4, 7], [8, 10]]
Explanation: The newInterval [5, 6] overlaps with [4, 5] and [6, 7]. So, they are merged into one interval [4, 7].

C++ (12) Start Timer

```cpp
class Solution {
  public:
    vector<vector<int>> insertInterval(vector<vector<int>> &interval
                                       vector<int> &newInterval) {
        // code here
        int n=intervals.size();
        vector<vector<int>> ans;
        int i=0;

        while(i<n && intervals[i][1]<newInterval[0])
        {
            ans.push_back(intervals[i]);
            i++;
        }
        while(i<n && intervals[i][0]<=newInterval[1])
        {
            newInterval[0]=min(intervals[i][0],newInterval[0]);
            newInterval[1]=max(intervals[i][1],newInterval[1]);
            i++;
        }
        ans.push_back(newInterval);
        while(i<n)
        {
            ans.push_back(intervals[i]);
            i++;
        }
        return ans;
    }
```

Custom Input Compile & Run Sub