Problem | Editorial | Submissions

Insert Interval

Difficulty: **Medium** Accuracy: **50.61%** Submissions: **32K+** Points: **4**

Geek has an array of non-overlapping intervals **intervals** where intervals[i] = [start$_i$, end$_i$] represent the start and the end of the i^{th} 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].

Input: intervals = [[1,2],[3,5],[6,7],[8,10],[12,16]], newInterval = [4,9]
Output: [[1,2], [3,10], [12,16]]
Explanation: The new interval [4,9] overlaps with [3,5],[6,7],[8,10].

Java (1.8) | Average Time: 30m | Start Timer

```
1  // } Driver Code Ends
40
41 class Solution {
42     public ArrayList<int[]> insertInterval(int[][] intervals, int[] newInterval) {
43         ArrayList<int[]> res = new ArrayList<>();
44         int i = 0;
45         int n = intervals.length;
46         while (i < n && intervals[i][1] < newInterval[0]) {
47             res.add(intervals[i]);
48             i++;
49         }
50         while (i < n && intervals[i][0] <= newInterval[1]) {
51             newInterval[0] = Math.min(newInterval[0], intervals[i][0]);
52             newInterval[1] = Math.max(newInterval[1], intervals[i][1]);
53             i++;
54         }
55         res.add(newInterval);
56         while (i < n) {
57             res.add(intervals[i]);
58             i++;
59         }
60         return res;
61     }
62 }
63
```

Custom Input | Compile & Run | Submit