[LeetCode](https://leetcode.com/problems/insert-interval/?envType=daily-question&envId=2024-03-17)

<https://github.com/AdityaKonda6/-50DaysOfCoding>

<https://leetcode.com/problems/insert-interval/?envType=daily-question&envId=2024-03-17>

<https://www.linkedin.com/in/aditya-adi-konda/>

Day 27 of [#50dayscodingchallenge](https://www.linkedin.com/feed/hashtag/?keywords=50dayscodingchallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633):  
[#leetcode](https://www.linkedin.com/feed/hashtag/?keywords=leetcode&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcodechallenge](https://www.linkedin.com/feed/hashtag/?keywords=leetcodechallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcodestreak](https://www.linkedin.com/feed/hashtag/?keywords=leetcodestreak&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcode2024](https://www.linkedin.com/feed/hashtag/?keywords=leetcode2024&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcode50day](https://www.linkedin.com/feed/hashtag/?keywords=leetcode50day&highlightedUpdateUrns=urn:li:activity:7166316239483461633)  
   
Just kicked off my coding journey with a fascinating problem - "Successfully solved LeetCode Problem “57. Insert Interval”  
   
✨ Task: You are given an array of non-overlapping intervals intervals where intervals[i] = [starti, endi] represent the start and the end of the ith interval and intervals is sorted in ascending order by starti. You are also given an interval newInterval = [start, end] that represents the start and end of another interval.

Insert newInterval into intervals such that intervals is still sorted in ascending order by starti and intervals still does not have any overlapping intervals (merge overlapping intervals if necessary).

Return intervals after the insertion.

Note that you don't need to modify intervals in-place. You can make a new array and return it.

Examples:

Example 1:

Input: intervals = [[1,3],[6,9]], newInterval = [2,5]

Output: [[1,5],[6,9]]

Example 2:

Input: intervals = [[1,2],[3,5],[6,7],[8,10],[12,16]], newInterval = [4,8]

Output: [[1,2],[3,10],[12,16]]

Explanation: Because the new interval [4,8] overlaps with [3,5],[6,7],[8,10].

Let's Connect:

If you find this problem intriguing or have insights to share, let's connect! I'm passionate about problem-solving, algorithmic thinking, and collaborative learning. Feel free to comment or reach out for engaging discussions and knowledge exchange.Unravel the mystery using your coding skills!

[#CodingChallenge](https://www.linkedin.com/feed/hashtag/?keywords=codingchallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#Algorithm](https://www.linkedin.com/feed/hashtag/?keywords=algorithm&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#LinkedInPost](https://www.linkedin.com/feed/hashtag/?keywords=linkedinpost&highlightedUpdateUrns=urn:li:activity:7166316239483461633) #Algorithm #Optimization #DataStructures #CodingChallenge  
  
Excited about the progress and challenges ahead!  
   
Make Sure You Follow My GitHub For Solutions: <https://github.com/AdityaKonda6/-50DaysOfCoding>  
  
  
Happy coding!

**Solution:-**

class Solution {

  public int[][] insert(int[][] intervals, int[] newInterval) {

    final int n = intervals.length;

    List<int[]> ans = new ArrayList<>();

    int i = 0;

    while (i < n && intervals[i][1] < newInterval[0])

      ans.add(intervals[i++]);

    // Merge overlapping intervals.

    while (i < n && intervals[i][0] <= newInterval[1]) {

      newInterval[0] = Math.min(newInterval[0], intervals[i][0]);

      newInterval[1] = Math.max(newInterval[1], intervals[i][1]);

      ++i;

    }

    ans.add(newInterval);

    while (i < n)

      ans.add(intervals[i++]);

    return ans.toArray(int[][] ::new);

  }

}

