# **Leetcode 57 – Insert Interval**

## Problem Understanding

You are given a list of **non-overlapping**, **sorted** intervals and a new interval.  
Insert the new interval in the correct position **and merge if needed** so that the final list remains:

* Sorted by start time
* With no overlapping intervals

### Example

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

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

Explanation: New interval [2,5] overlaps with [1,3], so we merge into [1,5].

## Optimized Java Solution

class Solution {

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

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

int i = 0;

int n = intervals.length;

// 1. Add intervals before newInterval

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

result.add(intervals[i]);

i++;

}

// 2. Merge overlapping intervals

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

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

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

i++;

}

result.add(newInterval); // Add the merged interval

// 3. Add remaining intervals

while (i < n) {

result.add(intervals[i]);

i++;

}

return result.toArray(new int[result.size()][]);

}

}

## Dry Run Using Table

### Input:

intervals = [[1,3], [6,9]]

newInterval = [2,5]

|  |  |  |
| --- | --- | --- |
| Step | Action | Result |
| 1 | [1,3] ends after [2,5] starts → merge | newInterval becomes [1,5] |
| 2 | [6,9] starts after newInterval ends | Add [6,9] as is |
| Final | Add merged [1,5] + [6,9] | [[1,5], [6,9]] |

✅ Output: [[1,5],[6,9]]

## Time / Space Complexity

|  |  |
| --- | --- |
| Metric | Value |
| Time | O(n) |
| Space | O(n) |

* One pass through the intervals
* Extra space for the result list

## Alternate Approaches

|  |  |  |  |
| --- | --- | --- | --- |
| Approach | Time | Space | Notes |
| ✅ Linear Scan + Merge | O(n) | O(n) | Best and standard approach |
| Brute Force + Sort + Merge | O(n log n) | O(n) | Insert then merge whole list (not optimal) |