LeetCode 57. Insert Interval

1. Problem Title & Link

- **57. Insert Interval**
- https://leetcode.com/problems/insert-interval/

2. Problem Statement (Short Summary)

We are given a list of **non-overlapping intervals** sorted by their start times, and a new interval.
We must insert the new interval into the correct position and **merge any overlapping intervals**, returning the final list of intervals.

3. Examples (Input → Output)

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

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

4. Constraints

- 0 <= intervals.length <= 10^4
- intervals[i].length == 2
- 0 <= start_i <= end_i <= 10^5
- intervals is sorted and non-overlapping
- newInterval.length == 2

5. Thought Process (Step by Step)

We handle in **3 phases**:

1. **Add all intervals that end before newInterval starts.**
2. **Merge all overlapping intervals with newInterval.**
   - Update newInterval = [min(start), max(end)].
3. **Add all intervals that start after newInterval ends.**

6. Pseudocode (Language-Independent)

```
result = []
for each interval in intervals:
    if interval.end < newInterval.start:
        add interval to result
```

```
    else if interval.start > newInterval.end:
        add newInterval to result
        add remaining intervals
        return result
    else:
        merge overlap:
        newInterval.start = min(newInterval.start, interval.start)
        newInterval.end   = max(newInterval.end, interval.end)

add newInterval to result
return result
```

7. Code Implementation

✅ **Python**

```
class Solution:
       def insert(self, intervals: List[List[int]], newInterval: List[int]) ->
List[List[int]]:
        res = []
        for interval in intervals:
            # case 1: interval ends before newInterval starts
            if interval[1] < newInterval[0]:
                res.append(interval)
            # case 2: interval starts after newInterval ends
            elif interval[0] > newInterval[1]:
                res.append(newInterval)
                res.extend(intervals[intervals.index(interval):])
                return res
            # case 3: overlap → merge
            else:
                newInterval[0] = min(newInterval[0], interval[0])
                newInterval[1] = max(newInterval[1], interval[1])
        res.append(newInterval)
        return res
```

✅ **Java**

```
class Solution {
    public int[][] insert(int[][] intervals, int[] newInterval) {
        List<int[]> res = new ArrayList<>();
        for (int[] interval : intervals) {
            if (interval[1] < newInterval[0]) {
                res.add(interval);
            } else if (interval[0] > newInterval[1]) {
                res.add(newInterval);
```

```
res.addAll(Arrays.asList(intervals).subList(Arrays.asList(intervals).indexOf(int
erval), intervals.length));
                return res.toArray(new int[res.size()][]);
            } else {
                newInterval[0] = Math.min(newInterval[0], interval[0]);
                newInterval[1] = Math.max(newInterval[1], interval[1]);
            }
        }
        res.add(newInterval);
        return res.toArray(new int[res.size()][]);
    }
}
```

8. Time & Space Complexity Analysis

- **Time:** O(n) (one pass through intervals)
- **Space:** O(n) (for result list)

9. Common Mistakes / Edge Cases

- Forgetting to add the merged interval at the end.
- Not handling empty intervals.
- Confusing between “before newInterval” and “after newInterval” conditions.

10. Variations / Follow-Ups

- Merge a **list of new intervals** instead of one.
- Handle unsorted intervals (requires sorting first).

11. Dry Run (Step by Step Execution)

👉 Input:

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

1. Start with result = []

- [1,2] → ends before 4 → add → result = [[1,2]]
- [3,5] → overlaps with [4,8] → merge → newInterval = [3,8]
- [6,7] → overlaps with [3,8] → merge → newInterval = [3,8]
- [8,10] → overlaps with [3,8] → merge → newInterval = [3,10]
- [12,16] → starts after 10 → add [3,10] first → result = [[1,2],[3,10]]

  Then add [12,16] → result = [[1,2],[3,10],[12,16]]

✅ Output:

Leetcode: 57

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