LeetCode 57

Description

Given a set of non-overlapping intervals, insert a new interval into the intervals (merge if necessary).

You may assume that the intervals were initially sorted according to their start times.

Example 1:
Given intervals [1,3],[6,9], insert and merge [2,5] in as [1,5],[6,9].

Example 2:
Given [1,2],[3,5],[6,7],[8,10],[12,16], insert and merge [4,9] in as [1,2],[3,10],[12,16].

This is because the new interval [4,9] overlaps with [3,5],[6,7],[8,10].

Thought

Break the intervals into 3 parts:
a. ones that's before the merge(when its end < start of merge_interval); b. ones that's after the merge(when its start > end of merge_interval);
c. those left needs to be merged.

Solution

```java
public List<Interval> insert(List<Interval> intervals, Interval newInterval) {
    List<Interval> result = new LinkedList<>();
    int i = 0;
    // add all the intervals ending before newInterval starts
    while (i < intervals.size() && intervals.get(i).end < newInterval.start)
        result.add(intervals.get(i++));
    // merge all overlapping intervals to one considering newInterval
    while (i < intervals.size() && intervals.get(i).start <= newInterval.end) {
        newInterval = new Interval( // we could mutate newInterval here also
                Math.min(newInterval.start, intervals.get(i).start),
                Math.max(newInterval.end, intervals.get(i).end));
        i++;
    }
    result.add(newInterval); // add the union of intervals we got
    // add all the rest
    while (i < intervals.size()) result.add(intervals.get(i++));
    return result;
}
```

Takeaways

- Breaking big problems into small problems