LeetCode 56

Description

Given a collection of intervals, merge all overlapping intervals.
For example,
Given [1,3],[2,6],[8,10],[15,18],
return [1,6],[8,10],[15,18].

Thought

First, we need to make sure the intervals are sorted by their starting point. Then we need to compare the 1st interval's end with the 2nd interval's starting point. If we find overlapping or the 2nd interval's start point >= 1st interval's end point, then we need to update the 1st interval's end point to the 2nd interval's end point.

We run the same process as above using a for loop, and create a new result List to save the merged intervals.

Sorting takes O(nlogn) and iterations takes O(n), overall time complexity is O(nlogn);

Solution

```
public List<Interval> mergeIntervals(List<Interval) intervals) {
  //edge case
  if (intervals.size() <= 1) {
    return intervals;
  }

  //sort the intervals by starting point
  intervals.sort((a,b) -> Interval.compare(a.start, b.start));

  //create new list to save result
  List<Interval> result = new LinkedList<Interval>();
  int start = intervals.get(0).start;
  int end = intervals.get(0).end;

  //apply same process using for loop
  for (Interval interval: intervals) {
    if (interval.start <= end) { //if overlap, update previous end to current inter
      end = Math.max(end, interval.end);
    } else { //if not overlap, create new interval with current start and end and a
      result.add(new Interval(start, end));
      start = interval.star;
```

```
        end = interval.end;
      }
    }

    //one more step, add the last interval with current start and end
    result.add(new Interval(start, end));
    return result;
  }
```

Takeaways

- Visualize the problem
- Consider edge cases