Merge Intervals

Given an array of `intervals` where `intervals[i] = [start_i, end_i]`, merge all overlapping intervals, and return *an array of the non-overlapping intervals that cover all the intervals in the input.*

Example 1:

```
Input: intervals = [[1,3],[2,6],[8,10],[15,18]]

Output: [[1,6],[8,10],[15,18]]

Explanation: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].
```

A **simple approach** is to start from the first interval and compare it with all other intervals for overlapping, if it overlaps with any other interval, then remove the other interval from the list and merge the other into the first interval. Repeat the same steps for remaining intervals after first. This approach cannot be implemented in better than O(n^2) time.

Approach 1

```
class Solution {
  public int[][] merge(int[][] intervals) {
    Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
    LinkedList<int[]> merged = new LinkedList<>();
    for (int[] interval : intervals) {
      // if the list of merged intervals is empty or if the current
      // interval does not overlap with the previous, simply append it.
      if (merged.isEmpty() || merged.getLast()[1] < interval[0]) {
        merged.add(interval);
      }
      // otherwise, there is overlap, so we merge the current and previous intervals.
      else {
        merged.getLast()[1] = Math.max(merged.getLast()[1], interval[1]);
      }
    }
    return merged.toArray(new int[merged.size()][]);
  }                          }
```

Approach 2 without Collections

```
class Solution {
    public int[][] merge(int[][] intervals) {
       Arrays.sort(intervals,(a,b)->Integer.compare(a[0],b[0]));
      int n=intervals.length;
        int j=0;
        int l=intervals[0][0];
        int r=intervals[0][1];
        for(int i=1;i<n;i++)
        {
            if(r>=intervals[i][0]&&r<=intervals[i][1])  {
                r=intervals[i][1];
            }
            else if(r<intervals[i][0])  {
                intervals[j][0]=l;
                intervals[j][1]=r;
                l=intervals[i][0];
                r=intervals[i][1];
                j++;
            }
        }
         intervals[j][0]=l;
        intervals[j][1]=r;
        int[][] res=new int[j+1][2];
        for(int i=0;i<j+1;i++) {
            res[i][0]=intervals[i][0];
            res[i][1]=intervals[i][1];
        }
        return res;
    }
}
```