```
/**
 * Definition for an interval.
 * public class Interval {
 *     int start;
 *     int end;
 *     Interval() { start = 0; end = 0; }
 *     Interval(int s, int e) { start = s; end = e; }
 * }
 */
class Solution {
    public int eraseOverlapIntervals(Interval[] intervals) {
        int len = intervals.length;
        if (len == 0)
            return 0;
        Arrays.sort(intervals, (a, b) -> a.end - b.end);
        int end = intervals[0].end, t = 0;

        for (int i = 1; i < len; i++) {
            if (intervals[i].start >= end) {
                end = intervals[i].end;
            } else {
                t++;
            }
        }
        return t;
    }
}

/**
 * Definition for an interval.
 * public class Interval {
 *     int start;
 *     int end;
 *     Interval() { start = 0; end = 0; }
 *     Interval(int s, int e) { start = s; end = e; }
 * }
 */
class Solution {
    public int eraseOverlapIntervals(Interval[] intervals) {
        int len = intervals.length;
        if (len == 0)
            return 0;
        Arrays.sort(intervals, (a, b) -> a.end - b.end);
        int cnt = 1, end = intervals[0].end;

        for (int i = 1; i < len; i++) {
            if (intervals[i].start >= end) {
                cnt++;
                end = intervals[i].end;
            }
        }
        return len - cnt;
    }
}
```