1288. Remove Covered Intervals

Medium 579 24 Add to List Share

Given a list of `intervals`, remove all intervals that are covered by another interval in the list.

Interval `[a,b)` is covered by interval `[c,d)` if and only if `c <= a` and `b <= d`.

After doing so, return *the number of remaining intervals*.

Example 1:

```
Input: intervals = [[1,4],[3,6],[2,8]]
Output: 2
Explanation: Interval [3,6] is covered by [2,8], therefore it is removed.
```

Example 2:

```
Input: intervals = [[1,4],[2,3]]
Output: 1
```

Example 3:

```
Input: intervals = [[0,10],[5,12]]
Output: 2
```

Example 4:

```
Input: intervals = [[3,10],[4,10],[5,11]]
Output: 2
```

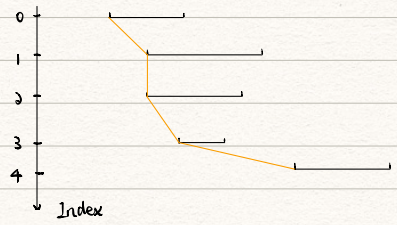
Example 5:

```
Input: intervals = [[1,2],[1,4],[3,4]]
Output: 1
```

Constraints:

- `1 <= intervals.length <= 1000`
- `intervals[i].length == 2`
- `0 <= intervals[i][0] < intervals[i][1] <= 10^5`
- All the intervals are **unique**.

1. Sort with start point.
   If start is same, sort the end in descending order.



2. Three cases
   I.

   Find the covered intervals.

   II.

   Merge two intervals into a big one.

   III.

   Two intervals don't cover.

```cpp
class Solution {
public:
    int removeCoveredIntervals(vector<vector<int>>& intervals) {
        // sort intervals
        sort(intervals.begin(), intervals.end(), compare);

        int left = (*intervals.begin())[0];
        int right = (*intervals.begin())[1];
        int res = intervals.size();

        for (auto it = intervals.begin() + 1; it != intervals.end(); it++) {
            // case 1
            if ((*it)[0] >= left && (*it)[1] <= right) {
                res--;
            }

            // case 2
            if ((*it)[0] >= left && (*it)[1] > right) {
                right = (*it)[1];
            }

            // case 3
            if ((*it)[0] >= right) {
                left = (*it)[0];
                right = (*it)[1];
            }

        }

        return res;
    }


private:
    static bool compare(const vector<int> interval_1, const vector<int>
interval_2) {
        if (interval_1[0] > interval_2[0]) {
            return false;
        } else if (interval_1[0] < interval_2[0]) {
            return true;
        } else if (interval_1[0] == interval_2[0]) {
            if (interval_1[1] <= interval_2[1]) {
                return false;
            }
        }
        return true;
    }
};
```