56. Merge Intervals

Medium 6197 354 Add to List Share

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] overlaps,
merge them into [1,6].
```

Example 2:

```
Input: intervals = [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considered
overlapping.
```

Constraints:

- `1 <= intervals.length <= 10^4`
- `intervals[i].length == 2`
- `0 <= start_i <= end_i <= 10^4`

1. Sort

2. For merged interval x, x.start is the minimum stard, and x.end is the maximum end.

Time complexity: O(n log n)
Space complexity: O(n)

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

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

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

            // next merged interval
            if ((*it)[0] > right) {
                res.push_back(vector({left, right}));
                left = (*it)[0];
                right = (*it)[1];
            }

        }

        res.push_back(vector({left, right}));

        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;
    }
};
```