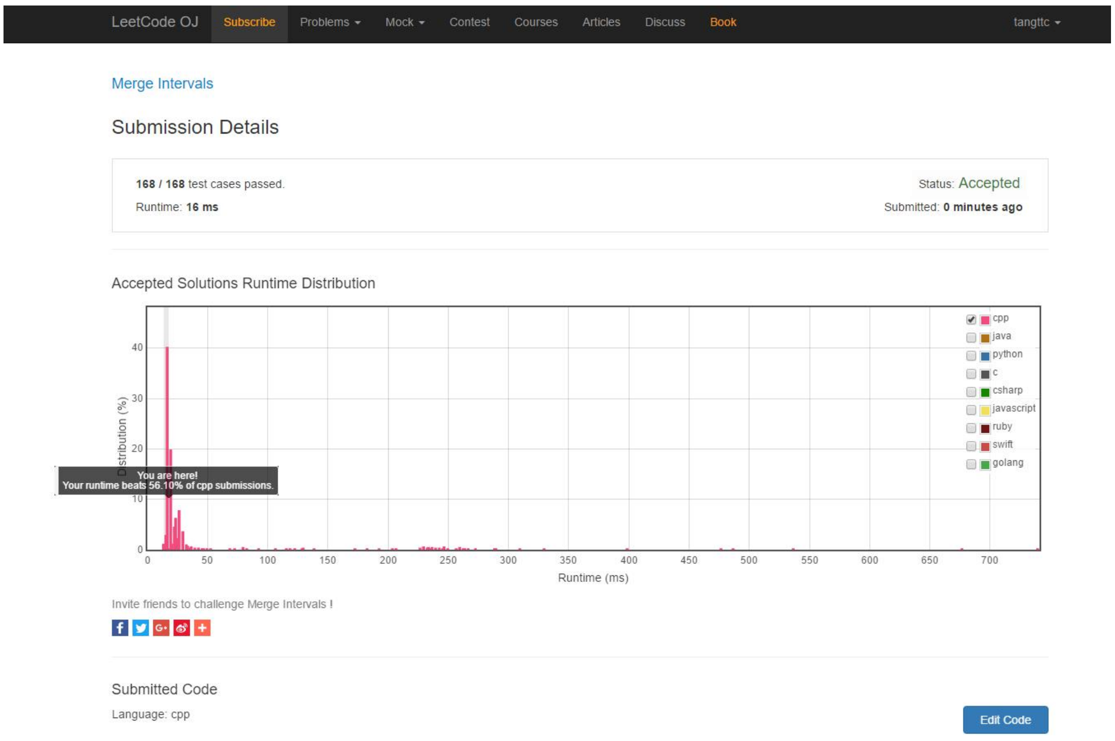
PA2 唐天成 515030910287

Leetcode 56: Merge Intervals



Merge Intervals

Submission Details

168 / 168 test cases passed.

Runtime: **16 ms**

Status: Accepted

Submitted: **0 minutes ago**

Accepted Solutions Runtime Distribution

You are here!
Your runtime beats 56.10% of cpp submissions.

Invite friends to challenge Merge Intervals !

Submitted Code

Language: cpp

Code:C++

```
/**
 * Definition for an interval.
 * struct Interval {
 *       int start;
 *       int end;
 *       Interval() : start(0), end(0) {}
 *       Interval(int s, int e) : start(s), end(e) {}
 * };
 */

#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

int cmp(const Interval &Interval_1, const Interval &Interval_2) { // sort according to the start of the struct
    if (Interval_1.start < Interval_2.start) {
        return 1;
    }
    else {
        return 0;
    }
```

```
}

class Solution {
public:
        vector<Interval> merge(vector<Interval>& intervals) {
                sort(intervals.begin(), intervals.end(), cmp); // first sort the vector
                vector<Interval> result;
                int size = intervals.size();
                if (size > 0) {
                         int last_start = intervals[0].start; // store the start and end of the interval that will
be the result
                        int max_end = intervals[0].end;
                        for (int i = 1 ;i < size; i++){
                                int start = intervals[i].start;
                                int end = intervals[i].end;
                                if (start == last_start){
                                        if (end > max_end){
                                                max_end = end;
                                        }
                                }
                                else if (start > last_start){
                                        if (start <= max_end){
                                                if (end > max_end){
                                                        max_end = end;
                                                }
                                        }
                                          else { // end updating the information of the current interval, and then
store it to vector<Interval> result
                                                Interval add(last_start,max_end);
                                                result.push_back(add);
                                                last_start = start; // updating last_start and max_end
                                                max_end = end;
                                        }
                                }
                                if (i == size - 1){ // if reaches the end of the vector, then store the information
                                        Interval last_add(last_start,max_end);
                                        result.push_back(last_add);
                                }
                        }
                }
                if (size == 1){ // if there is only one interval
                        Interval last_add(intervals[0].start,intervals[0].end);
                        result.push_back(last_add);
                }
                return result;
        }
};
```