Intervals

need sort the first — merge interval

or second one — remove overlapping

overlap need update the interval. using vector

using sort lambda function sort(intervals.begin(), intervals.end(), [](vector<int> a, vector<int> b { return a[1] < a[2];});

57. Insert Interval

```
vector<vector<int>> insert(vector<int>& intervals, vector<int>& newInterval) {
      //sorted based on intervals.start..
     // compare the intervals.end < newInterval.start push to res
     // intervals.end > newInterval.start min(newInterval.start, intervals.start)
     // max(newInterval.end, intervals.end)
     // label newInterval insert;
    vector<vector<int>> insert(vector<vector<int>>& intervals, vector<int>& newInterval) {
         vector<vector<int>> res;
         bool ins = false;
         for (auto& interval : intervals) {
            if (interval[1] < newInterval[0]) res.push_back(interval);
            if (interval[0] > newInterval[1]) {
                if (!ins){
                    res.push_back(newInterval);
                    ins = true;
                }
                res.push_back(interval);
            }

            if (interval[1] >= newInterval[0]) {
              newInterval[0] = min(interval[0], newInterval[0]);
              newInterval[1] = max(interval[1], newInterval[1]);
            }

           }
        if (!ins) res.push_back(newInterval);
        return res;
    }
```

435. Non-overlapping Intervals

```
435. Non-overlapping Intervals
int eraseOverlapIntervals(vector<vector<int>>& intervals) {
        sort(intervals.begin(), intervals.end(), [](vector<int>& a, vector<int>& b)         {return a[1] < b[1];});
        int eraseN = 0;
        int end1 = intervals[0][1];
        for (int i = 1; i < intervals.size(); ++i) {
            vector<int> interval = intervals[i];
            if (interval[0] < end1)
                eraseN++;
            else
                end1 = interval[1];
        }
        return eraseN;
    }
```

252. Meeting Rooms

Given an array of meeting time intervals consisting of start and end times `[[s1,e1],[s2,e2],...]` (si< ei), determine if a person could attend all meetings.

```
//Check are there any overlapping interval in the meeting schedual
//sort last then check
#include<vector>
#include<algorithm>
using namespace std;

/*Definition of Interval:
 * class Interval {
 * public:
 *     int start, end;
 *     Interval(int start, int end) {
 *         this->start = start;
 *         this->end = end;
 *     }
 * }
 */

bool isAttendMeeting(vector<vector<int>> intervals) {
     sort( intervals.begin(), intervals.end(), [](vector<int> &a, vector<int> &b){
return a[1] < b[1]; });
      int endT = intervals[0][1];
      for (int i = 1; i < intervals.size(); ++i) {
      if (intervals[i][0] < endT)  return false;
           endT = interval[i][1];
}
     return true;
}
```

56. Merge Intervals

```
vector<vector<int>> merge(vector<vector<int>>& intervals) {
        //sorting
        //lambda function

        sort(intervals.begin(), intervals.end(),[](vector<int> &a, vector<int> &b)
             { return a[0] < b[0];
             });
        vector<vector<int>> res;
        res.push_back(intervals[0]);
        for (int i = 1; i < intervals.size(); ++i) {
            vector<int> interval = intervals[i];
            if (res.back()[1] >= interval[0])
                res.back()[1] = max(res.back()[1],interval[1]);
            else
                res.push_back(interval);
        }

        return res;
    }
```