/\*

Given a collection of intervals, merge all overlapping intervals.

For example,

Given [1,3],[2,6],[8,10],[15,18],

return [1,6],[8,10],[15,18].

方法一：先按照start位置排序，然后挨个比较与身后的关系，如果可以合并就合并，然后删除后面一个。

方法二：建立一个新数组，然后往里面放，就可以不用删除。这样会快很多。

\*/

/\*\*

\* Definition for an interval.

\* struct Interval {

\* int start;

\* int end;

\* Interval() : start(0), end(0) {}

\* Interval(int s, int e) : start(s), end(e) {}

\* };

\*/

bool comp(Interval &a,Interval &b)

{

return a.start<b.start;

}

class Solution {

public:

vector<Interval> merge(vector<Interval>& intervals)

{

//way-1

/\*

if(intervals.size()==0)

return intervals;

sort(intervals.begin(),intervals.end(),comp);

for(int i=0;i<intervals.size()-1;i++)

{

if(intervals[i+1].start>intervals[i].end)

continue;

else if(intervals[i+1].start<=intervals[i].end)

{

intervals[i].end=(intervals[i+1].end>intervals[i].end)?intervals[i+1].end:intervals[i].end;

intervals.erase(intervals.begin()+i+1);

i--;

}

}

return intervals;

\*/

//way-2

if(intervals.size()==0)

return intervals;

sort(intervals.begin(),intervals.end(),comp);

vector<Interval> ret;

ret.push\_back(intervals[0]);

for(int i=1;i<intervals.size();i++)

{

if(intervals[i].start<=ret[ret.size()-1].end)

ret[ret.size()-1].end=max(ret[ret.size()-1].end,intervals[i].end);

else

ret.push\_back(intervals[i]);

}

return ret;

}

};