156. Merge Intervals

* [Description](http://www.lintcode.com/en/problem/merge-intervals/#description)
* [Notes](http://www.lintcode.com/en/problem/merge-intervals/#note)
* [Testcase](http://www.lintcode.com/en/problem/merge-intervals/#testcase)
* [Judge](http://www.lintcode.com/en/problem/merge-intervals/#judge)

[Discuss](http://www.lintcode.com/discuss)

Given a collection of intervals, merge all overlapping intervals.

Have you met this question in a real interview?

Yes

**Example**

Given intervals => merged intervals:

[ [

(1, 3), (1, 6),

(2, 6), => (8, 10),

(8, 10), (15, 18)

(15, 18) ]

]

[**Challenge**](http://www.lintcode.com/en/problem/merge-intervals/#challenge)

O(*n* log *n*) time and O(1) extra space.

<http://www.lintcode.com/en/problem/merge-intervals/#>

***/\*\****

***\* Definition of Interval:***

***\* public classs Interval {***

***\*     int start, end;***

***\*     Interval(int start, int end) {***

***\*         this.start = start;***

***\*         this.end = end;***

***\*     }***

***\* }***

***\*/***

**public** **class** Solution {

***/\*\****

***\* @param intervals: interval list.***

***\* @return: A new interval list.***

***\*/***

**void** quicksort(List<Interval> vector, **int** primero, **int** ultimo)

        {

**int** i, j, central;

**int** pivote;

            central = (primero + ultimo) / 2;

            pivote = vector.get( central).start;

            i = primero;

            j = ultimo;

**do**

            {

**while** (vector.get(i).start < pivote) i++;

**while** (vector.get(j).start > pivote) j--;

**if** (i <= j)

                {

                    Interval temp;

                    temp = vector.get(i);

*//vector[i] = vector[j];*

*//vector[j] = temp;*

                    vector.set(i, vector.get(j));

                    vector.set(j, temp);

                    i++;

                    j--;

                }

            } **while** (i <= j);

**if** (primero < j)

            {

                quicksort(vector, primero, j);

            }

**if** (i < ultimo)

            {

                quicksort(vector, i, ultimo);

            }

        }

**public** List<Interval> merge(List<Interval> intervals)

        {

**if** (intervals == **null** || intervals.isEmpty()) **return** intervals;

            quicksort(intervals, 0, intervals.size() - 1);

**int** i = 0;

            List<Interval> res = **new** ArrayList();

**while**(i < intervals.size())

            {

                Interval a = intervals.get(i);

                Interval insertar = a;

**while** (i +1 < intervals.size() && intervals.get(i+1).start <= a.end)

                {

                    insertar = **new** Interval(Math.min( a.start, intervals.get(i+1).start) , Math.max(a.end, intervals.get(i + 1).end));

                    a = insertar;

                    i++;

                }

                res.add(insertar);

                i++;

            }

**return** res;

        }

}