class Solution {

boolean doesIntervalsOverlap(int[] a, int[] b){

return Math.min(a[1],b[1])-Math.max(a[0],b[0])>=0;

}

int[] mergeIntervals(int[] a, int[] b){

int[] newInterval={Math.min(a[0],b[0]),Math.max(a[1],b[1])};

return newInterval;

}

public int[][] insertInterval(int[][] intervals, int[] newInterval){

boolean isIntervalInserted=false;

List<int[]> list=new ArrayList<>(Arrays.asList(intervals));

for(int i=0;i<intervals.length;i++){

if(newInterval[0]<intervals[i][0]){

list.add(i,newInterval);

isIntervalInserted=true;

break;

}

}

if(!isIntervalInserted){

list.add(newInterval);

}

return list.toArray(new int[list.size()][2]);

}

public int[][] insert(int[][] intervals, int[] newInterval) {

intervals=insertInterval(intervals,newInterval);

List<int[]> answer=new ArrayList<>();

for(int i=0;i<intervals.length;i++){

int[] currInterval={intervals[i][0],intervals[i][1]};

while(i<intervals.length && doesIntervalsOverlap(currInterval, intervals[i])){

currInterval=mergeIntervals(currInterval, intervals[i]);

i++;

}

i--;

answer.add(currInterval);

}

return answer.toArray(new int[answer.size()][2]);

}

public List<List<Integer>> removeInterval(int[][] intervals, int[] toBeRemoved) {

List<List<Integer>> ans = new ArrayList<>();

for (int[] interval : intervals) {

final int a = interval[0];

final int b = interval[1];

if (a >= toBeRemoved[1] || b <= toBeRemoved[0]) {

ans.add(Arrays.asList(a, b));

} else { // a < toBeRemoved[1] && b > toBeRemoved[0]

if (a < toBeRemoved[0])

ans.add(Arrays.asList(a, toBeRemoved[0]));

if (b > toBeRemoved[1])

ans.add(Arrays.asList(toBeRemoved[1], b));

}

}

return ans;

}

}