452. Minimum Number of Arrows to Burst Balloons

There are a number of spherical balloons spread in two-dimensional space. For each balloon, provided input is the start and end coordinates of the horizontal diameter. Since it's horizontal, y-coordinates don't matter and hence the x-coordinates of start and end of the diameter suffice. Start is always smaller than end. There will be at most 104 balloons.

An arrow can be shot up exactly vertically from different points along the x-axis. A balloon with xstart and xend bursts by an arrow shot at x if xstart ≤ x ≤ xend. There is no limit to the number of arrows that can be shot. An arrow once shot keeps travelling up infinitely. The problem is to find the minimum number of arrows that must be shot to burst all balloons.

**Example:**

**Input:**

[[10,16], [2,8], [1,6], [7,12]]

**Output:**

2

**Explanation:**

One way is to shoot one arrow for example at x = 6 (bursting the balloons [2,8] and [1,6]) and another arrow at x = 11 (bursting the other two balloons).

題目解析：有幾個大小不一的氣球放一條直線上，氣球可以重疊。參數中，每個數組就是氣球覆蓋的起始點和結束點。用箭射氣球。問最少可以用多少支箭射爆所有氣球。

思路：

氣球數組按照起始點排序，這裡用了升序。一開始記錄第一個下氣球的end，如果下一個氣球B起始點小於等於end，氣球B也可以之前的一起爆。這時，end=Math.min(end, end of 氣球B)。

如果發現某個氣球起始點大於end，這個氣球必須用第二支箭射。這時，end=這個氣球的end，arrowcount++；

public int findMinArrowShots(int[][] points) {

if(points==null || points.length==0 || points[0].length==0) return 0;

Arrays.sort(points, new Comparator<int[]>() {

public int compare(int[] a, int[] b) {

if(a[0]==b[0]) return a[1]-b[1];

else return a[0]-b[0];

}

});

int minArrows = 1;

int arrowLimit = points[0][1];

for(int i=1;i<points.length;i++) {

int[] baloon = points[i];

if(baloon[0]<=arrowLimit) {

arrowLimit=Math.min(arrowLimit, baloon[1]);

} else {

minArrows++;

arrowLimit=baloon[1];

}

}

return minArrows;

}

435. Non-overlapping Intervals

給你幾個數組，當中最多幾個，區間不重疊。題目問需要剔除多少個。所以用Intervals.length-count。

思路跟上題452類似，我將Interval按照起始點排序。

如果下一個與上一個重疊，即A.end>B.start，A和B中留下end較小的一個(greedy)，剔除較大的一個。

如果下一個與上一個不重疊，就count++，end=B.end.

public class Solution {

public int eraseOverlapIntervals(Interval[] intervals) {

if(intervals.length<=1)return 0;

int len=intervals.length;

Arrays.sort(intervals,new Comparator<Interval>(){

public int compare(Interval a, Interval b){

return a.start-b.start;

}

});

int end=intervals[0].end;

int count=1;

for(int i=1;i<len;i++){

if(intervals[i].start<end)

{

end=Math.min(end,intervals[i].end);

}

else{

count++;

end=intervals[i].end;

}

}

return len-count;

}

}

134. Gas Station

cost[i]是station I 到station i+1的油耗，gas[i]是station[i]的加油站油量。汽車沒裝任何油，問從哪個station開始走能走完一圈？

思路：如果totalGAS>=totalCost一定存在一個begin station可以走完一圈。

從station 0 開始走。

如果某個station 不夠油走下去，說明從A 到B中需要額外的油，A到B中間的不可能是begin station，因為沒有之前的油，更加走不完A到B。把begin++；

直到掃描到數組最後，如果totalgas>=totalcost, begin 就是開始的station，否則沒有哪個station可以走完一圈。

class Solution

{

public:

int canCompleteCircuit(vector<int>& gas, vector<int>& cost)

{

int totalGas = 0;

int totalCost = 0;

int tank = 0;

int begin = 0;

for (int i = 0; i < gas.size(); i++)

{

totalGas += gas[i];

totalCost += cost[i];

tank += (gas[i] - cost[i]);

if (tank < 0)

{

begin = i + 1;

tank = 0;

}

}

return (totalGas >= totalCost ? begin : -1);

}

};

class Solution {

public:

int canCompleteCircuit(vector<int>& gas, vector<int>& cost) {

int size=gas.size();

int sum=0;

int res=0;

int total=0;

for(int i=0; i<size; ++i){

sum+=gas[i]-cost[i];

if(sum<0){

total+=sum;

sum=0;

res=i+1;

}

}

total+=sum;

return total<0?-1:res;

}};