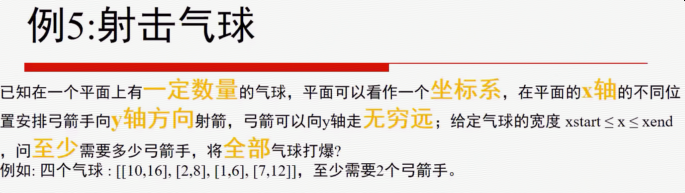
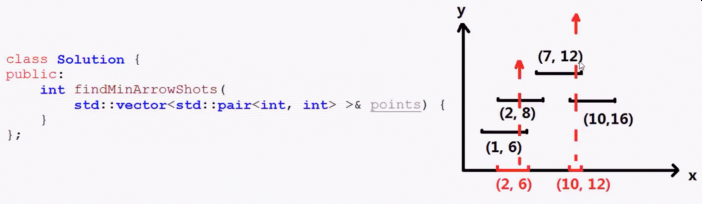
Leetcode\_452\_MinimumNumberofArrowstoBurstBalloons\_射击气球\_Medium

# Leetcode\_452\_MinimumNumberofArrowstoBurstBalloons\_射击气球\_Medium

## 问题介绍





\* 难度：Medium

\* <https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/description/>

\*

\* 题目介绍：

\* 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:

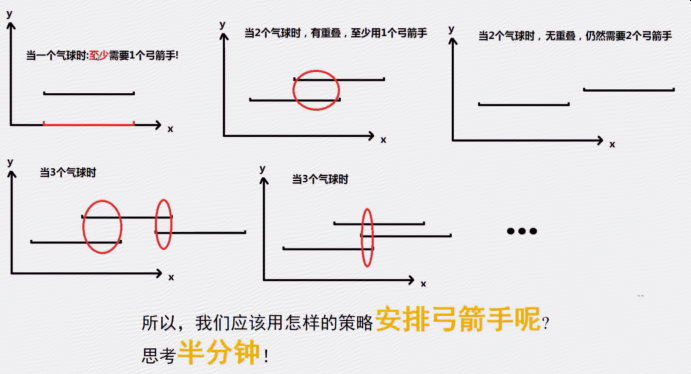
\* Input:[[10,16], [2,8], [1,6], [7,12]]

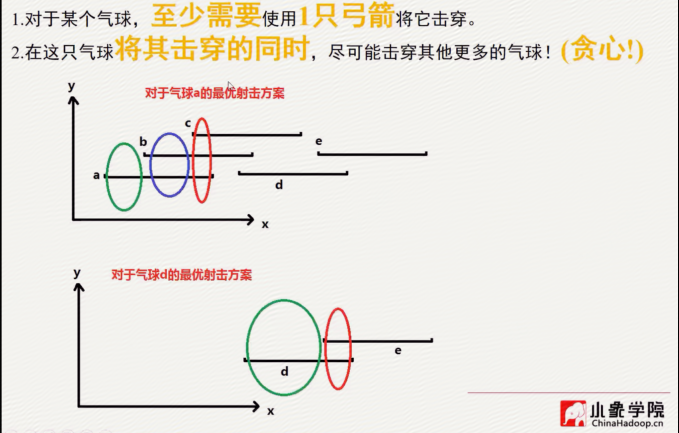
\* Output: 2

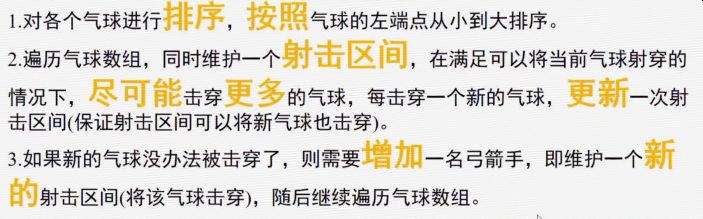
\* Explanation:One way is to shoot one arrow for example at x = 6

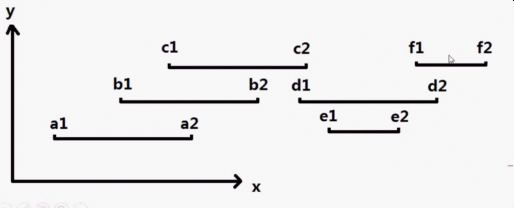
\* (bursting the balloons [2,8] and [1,6]) and another arrow at x = 11 (bursting the other two balloons).

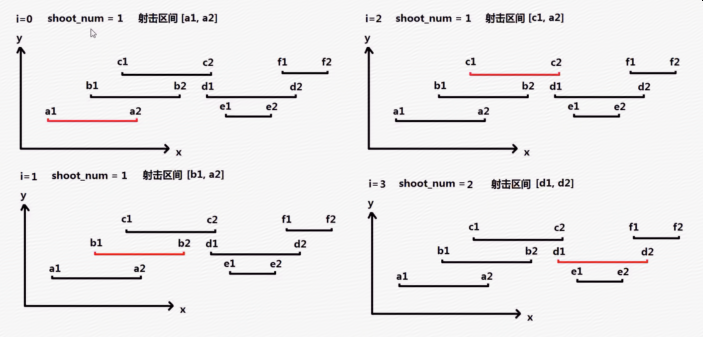
## 思路分析

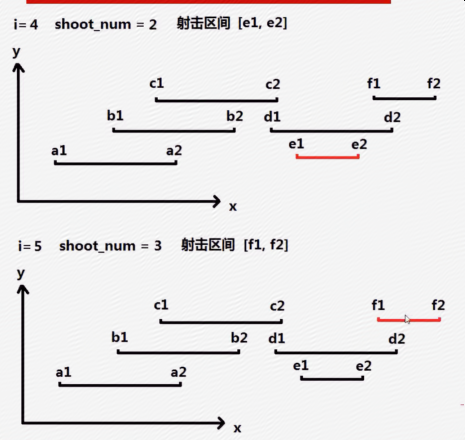












\* 思路分析：利用贪心思想，尽可能地让一名弓箭手射击尽可能多的气球。

\* 只需要初始化第一个气球的右边界right，然后判断后续的气球的左边界是否在right的左侧，

\* 若在，则可以射击该气球，并更新right；（更新right：取原来right和当前气球的右边界的最小值）；

\* 若不在，则需要重新安排一位弓箭手，射击后续气球。

\* 一开始是设置：[left，right];但是left是没有必要设置的，只需要比较right就可以了。

\* 注意：紧紧挨着也算是可以射击中。如{1,2},{2,3}只需要一次射击。

## Java代码

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

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

//对这些点按照起点大小排序

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

**@Override**

**public int compare(int[] o1, int[] o2) {**

**return o1[0]-o2[0];//只根据左坐标排序**

**}**

**});**

int arrowCount = 1,right = points[0][1];//初始化

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

if(points[i][0] <= right){//尽可能一箭穿更多气球

right = right > points[i][1]?points[i][1]:right;//更新右边界

}else{

arrowCount++;//开启新的弓箭手

right = points[i][1];//限制右边界

}

}

return arrowCount;

}

