题目难度：MEDUM

在二维空间中有许多球形的气球。

对于每个气球，提供的输入是水平方向上，气球直径的开始和结束坐标。

由于它是水平的，所以纵坐标并不重要，因此只要知道开始和结束的横坐标就足够了。

开始坐标总是小于结束坐标。

一支弓箭可以沿着 x 轴从不同点完全垂直地射出。

在坐标 x 处射出一支箭，若有一个气球的直径的开始和结束坐标为 xstart，xend， 且满足 xstart ≤ x ≤ xend，则该气球会被引爆。

可以射出的弓箭的数量没有限制。

弓箭一旦被射出之后，可以无限地前进。

我们想找到使得所有气球全部被引爆，所需的弓箭的最小数量。

给你一个数组 points ，其中 points [i] = [xstart,xend] ，返回引爆所有气球所必须射出的最小弓箭数。

示例 1：

输入：points = [[10,16],[2,8],[1,6],[7,12]]

输出：2

解释：对于该样例，x = 6 可以射爆 [2,8],[1,6] 两个气球，以及 x = 11 射爆另外两个气球

示例 2：

输入：points = [[1,2],[3,4],[5,6],[7,8]]

输出：4

示例 3：

输入：points = [[1,2],[2,3],[3,4],[4,5]]

输出：2

示例 4：

输入：points = [[1,2]]

输出：1

示例 5：

输入：points = [[2,3],[2,3]]

输出：1

提示：

0 <= points.length <= 104

points[i].length == 2

-231 <= xstart < xend <= 231 - 1

思路：贪心

先判空（WA点：没判空）

然后判断长度为1直接返回1

对数组从小到大排序，然后贪心逆序遍历最后一项的左边界是否在前一项的范围内

如果在就继续循环

否则结果加一，然后更新当前项的左边界为判断值

最后返回结果即可

时间：前33%

空间：前50%

上代码：

class Solution {

public:

int res = 1;

int findMinArrowShots(vector<vector<int>>& points) {

if(points.size() == 0) return 0;

if(points.size() == 1) return res;

sort(points.begin(), points.end());

int cur = points[points.size() - 1][0];

for(int i = points.size() - 2; i >= 0; i--) {

if(cur <= points[i][1]) continue;

else {

res++;

cur = points[i][0];

}

}

return res;

}

};