[LeetCode](https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/submissions/1206841549/?envType=daily-question&envId=2024-03-18)

<https://github.com/AdityaKonda6/-50DaysOfCoding>

<https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/submissions/1206841549/?envType=daily-question&envId=2024-03-18>

<https://www.linkedin.com/in/aditya-adi-konda/>

Day 28 of [#50dayscodingchallenge](https://www.linkedin.com/feed/hashtag/?keywords=50dayscodingchallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633):  
[#leetcode](https://www.linkedin.com/feed/hashtag/?keywords=leetcode&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcodechallenge](https://www.linkedin.com/feed/hashtag/?keywords=leetcodechallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcodestreak](https://www.linkedin.com/feed/hashtag/?keywords=leetcodestreak&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcode2024](https://www.linkedin.com/feed/hashtag/?keywords=leetcode2024&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcode50day](https://www.linkedin.com/feed/hashtag/?keywords=leetcode50day&highlightedUpdateUrns=urn:li:activity:7166316239483461633)  
   
Just kicked off my coding journey with a fascinating problem - "Successfully solved LeetCode Problem “452. Minimum Number of Arrows to Burst Balloons.”  
   
✨ Task: There are some spherical balloons taped onto a flat wall that represents the XY-plane. The balloons are represented as a 2D integer array points where points[i] = [xstart, xend] denotes a balloon whose horizontal diameter stretches between xstart and xend. You do not know the exact y-coordinates of the balloons.

Arrows can be shot up directly vertically (in the positive y-direction) from different points along the x-axis. A balloon with xstart and xend is burst by an arrow shot at x if xstart <= x <= xend. There is no limit to the number of arrows that can be shot. A shot arrow keeps traveling up infinitely, bursting any balloons in its path.

Given the array points, return the minimum number of arrows that must be shot to burst all balloons.

Examples:

Example 1:

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

Output: 2

Explanation: The balloons can be burst by 2 arrows:

- Shoot an arrow at x = 6, bursting the balloons [2,8] and [1,6].

- Shoot an arrow at x = 11, bursting the balloons [10,16] and [7,12].

Example 2:

Input: points = [[1,2],[3,4],[5,6],[7,8]]

Output: 4

Explanation: One arrow needs to be shot for each balloon for a total of 4 arrows.

Example 3:

Input: points = [[1,2],[2,3],[3,4],[4,5]]

Output: 2

Explanation: The balloons can be burst by 2 arrows:

- Shoot an arrow at x = 2, bursting the balloons [1,2] and [2,3].

- Shoot an arrow at x = 4, bursting the balloons [3,4] and [4,5].

Let's Connect:

If you find this problem intriguing or have insights to share, let's connect! I'm passionate about problem-solving, algorithmic thinking, and collaborative learning. Feel free to comment or reach out for engaging discussions and knowledge exchange.Unravel the mystery using your coding skills!

[#CodingChallenge](https://www.linkedin.com/feed/hashtag/?keywords=codingchallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#Algorithm](https://www.linkedin.com/feed/hashtag/?keywords=algorithm&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#LinkedInPost](https://www.linkedin.com/feed/hashtag/?keywords=linkedinpost&highlightedUpdateUrns=urn:li:activity:7166316239483461633) #Algorithm #Optimization #DataStructures #CodingChallenge  
  
Excited about the progress and challenges ahead!  
   
Make Sure You Follow My GitHub For Solutions: <https://github.com/AdityaKonda6/-50DaysOfCoding>  
  
  
Happy coding!

**Solution:-**

class Solution {

 public:

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

    ranges::sort(points,

                 [](const auto& a, const auto& b) { return a[1] < b[1]; });

    int ans = 1;

    int arrowX = points[0][1];

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

      if (points[i][0] > arrowX) {

        arrowX = points[i][1];

        ++ans;

      }

    return ans;

  }

};

