Problem 3661: Maximum Walls Destroyed by Robots

Problem Information

Difficulty: Hard
Acceptance Rate: 23.59%
Paid Only: No
Tags: Array, Binary Search, Dynamic Programming, Sorting

Problem Description

There is an endless straight line populated with some robots and walls. You are given integer arrays `robots`, `distance`, and `walls`:

* `robots[i]` is the position of the `ith` robot. * `distance[i]` is the **maximum** distance the `ith` robot's bullet can travel. * `walls[j]` is the position of the `jth` wall.

Every robot has **one** bullet that can either fire to the left or the right **at most**`distance[i]` meters.

A bullet destroys every wall in its path that lies within its range. Robots are fixed obstacles: if a bullet hits another robot before reaching a wall, it **immediately stops** at that robot and cannot continue.

Return the **maximum** number of **unique** walls that can be destroyed by the robots.

Notes:

* A wall and a robot may share the same position; the wall can be destroyed by the robot at that position. * Robots are not destroyed by bullets.

Example 1:

Input: robots = [4], distance = [3], walls = [1,10]

Output: 1

Explanation:

* `robots[0] = 4` fires **left** with `distance[0] = 3`, covering `[1, 4]` and destroys `walls[0] = 1`.
* Thus, the answer is 1.

Example 2:

Input: robots = [10,2], distance = [5,1], walls = [5,2,7]

Output: 3

Explanation:

* `robots[0] = 10` fires **left** with `distance[0] = 5`, covering `[5, 10]` and destroys `walls[0] = 5` and `walls[2] = 7`. * `robots[1] = 2` fires **left** with `distance[1] = 1`, covering `[1, 2]` and destroys `walls[1] = 2`. * Thus, the answer is 3.

Example 3:

Input: robots = [1,2], distance = [100,1], walls = [10]

Output: 0

Explanation:

In this example, only `robots[0]` can reach the wall, but its shot to the **right** is blocked by `robots[1]`; thus the answer is 0.

Constraints:

* `1 <= robots.length == distance.length <= 105` * `1 <= walls.length <= 105` * `1 <= robots[i], walls[j] <= 109` * `1 <= distance[i] <= 105` * All values in `robots` are **unique** * All values in `walls` are **unique**

Code Snippets

C++:

```
class Solution {
public:
int maxWalls(vector<int>& robots, vector<int>& distance, vector<int>& walls)
{

}
};
```

Java:

```
class Solution {
public int maxWalls(int[] robots, int[] distance, int[] walls) {

}
}
```

Python3:

```
class Solution:
def maxWalls(self, robots: List[int], distance: List[int], walls: List[int])
-> int:
```