1453. Maximum Number of Darts Inside of a Circular Dartboard

Leetcode Link

Problem Description

In this problem, you have a square wall with a circular dartboard. You have been asked to throw darts at the wall blindfolded and each throw results are represented as an array of points in a 2D plane. The task is to find out the maximum number of points that fall within or lie on any circular dartboard of a given radius, `r`.

Let's have a look at an example:

Input: points = [[-2,0], [2,0], [0,2], [0,-2]], r = 2 Output: 4

Here all the dart throws fall on the dartboard of radius 2 unit, centered on the origin. Thus, the maximum number of points that can fit inside this circular dartboard is 4.

Approach

The given problem can be solved using a geometric approach. We generate all the pairs of the given points and find out the 2 possible centers of the maximum circle containing those two points. Later on, we check for each center's location if it holds those points or not and keep track of the maximum count of points that can be included.

To make this process easier, we can create a `Point` structure to represent a point in 2D coordinate space and use some geometry functions that can calculate distance, create a circle given two points.

Pseudo-code:

1. Convert given array of points into a list (vector) of Point objects.
2. For each pair of points, construct two circles where each point is on the edge of the circle.
3. For each of these circles, calculate the number of points within that circle.
4. Keep track of the maximum number of points encountered.

Now let's translate this approach into solutions in different languages:

Python Solution

```
```

Java Solution

```
```

Javascript Solution

```
```

C++ Solution

```
```

C# Solution

```
```

Ruby Solution

```
```

PHP Solution

```
```

Each language handles the geometric calculations slightly differently, but the principle remains the same. The code iterates over each pair of points, calculates possible circle centers, and then checks the number of points that sit inside the circle. The maximum of these counts is then returned as the result.