Codility_

CodeCheck Report: trainingD2TTCE-EGA

Test Name:

Check out Codility training tasks

Summary | Timeline

Tasks summary

Task	Time spent	Score
NumberOfDiscIntersections C#	12 min	100%

Total score

Tasks Details

Medium

1. NumberOfDiscIntersections

Compute the number of intersections in a sequence of discs.

Task Score	Correctness	Performance
100%	100%	100%

Task description

We draw N discs on a plane. The discs are numbered from 0 to N − 1. An array A of N non-negative integers, specifying the radiuses of the discs, is given. The J-th disc is drawn with its center at (J, 0) and radius A[J].

We say that the J-th disc and K-th disc intersect if J ≠ K and the J-th and K-th discs have at least one common point (assuming that the discs contain their borders).

The figure below shows discs drawn for N = 6 and A as follows:

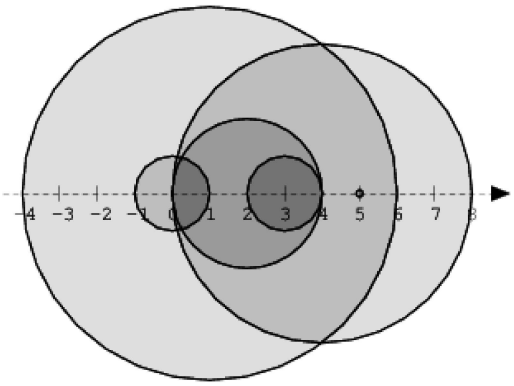
```
A[0] = 1
A[1] = 5
A[2] = 2
A[3] = 1
A[4] = 4
A[5] = 0
```

Solution

Programming language used:	C#
Total time used:	12 minutes
Effective time used:	12 minutes
Notes:	*not defined yet*

Task timeline

19:31:22 19:43:06



There are eleven (unordered) pairs of discs that intersect, namely:

- discs 1 and 4 intersect, and both intersect with all the other discs;
- disc 2 also intersects with discs 0 and 3.

Write a function:

```
class Solution { public int solution(int[] A); }
```

that, given an array A describing N discs as explained above, returns the number of (unordered) pairs of intersecting discs. The function should return −1 if the number of intersecting pairs exceeds 10,000,000.

Given array A shown above, the function should return 11, as explained above.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [0..100,000];
- each element of array A is an integer within the range [0..2,147,483,647].

Code: 19:43:06 UTC, cs, final, score: **100** show code in pop-up

```
using System;

/**
 * 6.4 - Number of Discs Intersercts
 * Paulo Santos
 * 07.Dec.2022
 */
class Solution {
    public int solution(int[] A) {

    int result = 0;
    var dps = new int[A.Length];
    var dpe = new int[A.Length];

    /*
     * Calculate start and finish
     * For the disks
     */
    for (int i = 0, x = A.Length - 1; i < A.Length;
    {
        var s = (i > A[i])     ? i - A[i] : 0;
        var e = (x - i > A[i]) ? i + A[i] : x;
        dps[s]++; // Add overlap at the start
        dpe[e]++; // Add overlap ar the end
    }

    var t = 0;
    for (var i = 0; i < A.Length; i++)
    {
        /*
         * Does a circle starts
         * at this position?
         */
        if (dps[i] > 0)
        {
            /*
             * Calculate the number
             * of overlaps
             */
            result += t * dps[i];
            result += dps[i] * (dps[i] - 1) / 2;

            /*
             * If the result is grater than 10,000,(
             * Return -1.
             */
            if (result > 10000000)
                return -1;

            /*
             * Add the overlap to a
             * temporary variable
             */
            t += dps[i];
        }
        t -= dpe[i];
    }

    return result;

    }
}
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(N*log(N)) or O(N)**

expand all	Example tests	
▶	example1 example test	✓ OK
expand all	**Correctness tests**	
▶	simple1	✓ OK
▶	simple2	✓ OK
▶	simple3	✓ OK
▶	extreme_small empty and [10]	✓ OK
▶	small1	✓ OK
▶	small2	✓ OK
▶	small3	✓ OK
▶	overflow arithmetic overflow tests	✓ OK
expand all	**Performance tests**	
▶	medium1	✓ OK
▶	medium2	✓ OK
▶	medium3	✓ OK
▶	medium4	✓ OK
▶	10M_intersections 10.000.000 intersections	✓ OK
▶	big1	✓ OK
▶	big2	✓ OK
▶	big3 [0]*100.000	✓ OK