Codility_

CodeCheck Report: trainingCQHBUW-WTC

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
NumberOfDiscIntersections Python	16 min	100%

Total score

Tasks Details

1. **NumberOfDiscIntersections**
Compute the number of intersections in a sequence of discs.

Medium

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

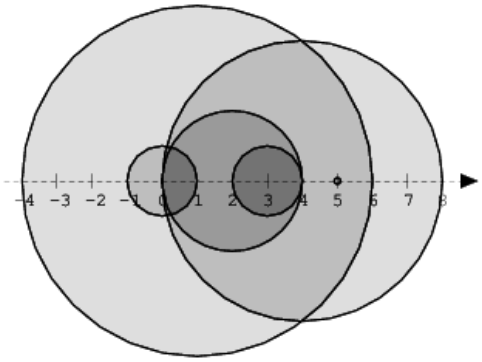
Programming language used:	Python
Total time used:	16 minutes
Effective time used:	16 minutes
Notes:	*not defined yet*

Task timeline

07:55:15 08:10:47

Code: 08:10:47 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debugging purpose
2  # print("this is a debug message")
3
4  def solution(A):
5      # Implement your solution here
```



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:

```
def solution(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].


```
6      # pass
7      N = len(A)
8      left_edges = [0] * N
9      right_edges = [0] * N
10
11     for i in range(N):
12         left_edges[i] = i - A[i]
13         right_edges[i] = i + A[i]
14
15     left_edges.sort()
16     right_edges.sort()
17
18     intersections = 0
19     circles = 0
20
21     j = 0
22     for i in range(N):
23         while j < N and right_edges[i] >= left_
24             intersections += circles
25             circles += 1
26             if intersections > 10_000_000:
27                 return -1
28             j += 1
29         circles -= 1
30
31     return intersections
32
```

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	✔ OK

[0]*100.000