Codility_

CodeCheck Report: training3GNSN9-9W9

Test Name:

Check out Codility training tasks

Summary | Timeline

Tasks summary

Task	Time spent	Score
NumberOfDiscIntersections ⚠ Python	2 min	100%

Total score

100%

Tasks Details

Medium

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

Task Score	Correctness	Performance
100%	100%	100%

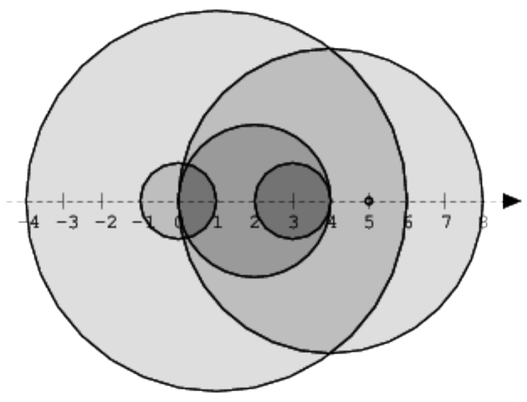
Task description

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
```



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].

Solution

Programming language used:	Python
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	*not defined yet*

Task timeline

22:42:23 — 22:44:07

Code: 22:44:07 UTC, py, final, score: 100 — show code in pop-up

```python
# you can write to stdout for debugging purposes, e.g.
# print("this is a debug message")

def solution(A):
    # write your code in Python 3.6
    N = len(A)
    if N < 2:
        return 0
    if N == 2:
        if 1 <= A[0]+A[1]:
            return 1
        else:
            return 0

    disksides = []
    for idx, el in enumerate(A):
        disksides += [(idx-el, 'left'), (idx+el, 'right')]
    disksides.sort(key=lambda x: (x[0], x[1]))

    diskpairs = 0
    disksundercheck = 0
    for dummy, side in disksides:
        if side=='left':
            diskpairs += disksundercheck
            if diskpairs > 10000000:
                return -1
            disksundercheck += 1
        elif side=='right':
            disksundercheck -= 1
        else:
            pass

    return diskpairs
```

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