



Demo ticket

Session

ID: demoKKXUQB-FYF
Time limit: 120 min.

Status: closed

Created on: 2014-03-25 01:34 UTC
Started on: 2014-03-25 01:34 UTC
Finished on: 2014-03-25 01:35 UTC

Tasks in test

1 | NumberOfDiscIntersections

Task score

100%

Test score

100%

100 out of 100 points

HARD

1. NumberOfDiscIntersections

Compute intersections between sequence of discs.

score: 100 of 100



Task description

Given an array A of N integers, we draw N discs in a 2D plane such that the I-th disc is centered on (0,I) and has a radius of A[I]. We say that the J-th disc and K-th disc intersect if $J \neq K$ and J-th and K-th discs have at least one common point.
Write a function:

```
def solution(A)
```

that, given an array A describing N discs as explained above, returns the number of pairs of intersecting discs. For example, given N=6 and:

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

intersecting discs appear in eleven pairs of elements:

- 0 and 1,
- 0 and 2,
- 0 and 4,
- 1 and 2,
- 1 and 3,
- 1 and 4,
- 1 and 5,
- 2 and 3,
- 2 and 4,
- 3 and 4,
- 4 and 5.

so the function should return 11.

The function should return -1 if the number of intersecting pairs exceeds 10,000,000.

Assume that:

- N is an integer within the range [0..100,000];
- each element of array A is an integer within the range [0..2147483647].

Complexity:

- expected worst-case time complexity is $O(N \log(N))$;
- expected worst-case space complexity is $O(N)$.

Solution

Programming language used: Python

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: correct functionality and scalability

Task timeline



01:34:22

01:35:09

Code: 01:35:09 UTC, py, final, score: 100.00

```
01. def solution(A):
02.     discs = [(i-a, i+a) for i, a in enumerate(A)]
03.     discs.sort()
04.
05.     count = 0
06.     for i, d in enumerate(discs):
07.         count += find_overlap(discs, d[1]) - i
08.         if count > 10000000:
09.             return -1
10.     return count
11.
12. def find_overlap(discs, x):
13.     low = 0
14.     high = len(discs) - 1
15.     while low <= high:
16.         mid = (low + high) // 2
17.         if discs[mid][0] <= x:
18.             low = mid + 1
19.         else:
20.             high = mid - 1
21.     return mid if discs[mid][0] <= x else mid-1
```

3/24/2014

Test results - Codility

beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Copyright 2009–2014 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

Analysis

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

test	time	result
example1 example test	0.050 s.	OK
simple1	0.050 s.	OK
simple2	0.050 s.	OK
simple3	0.050 s.	OK
extreme_small empty and [10]	0.050 s.	OK
small1	0.050 s.	OK
small2	0.050 s.	OK
small3	0.050 s.	OK
overflow arithmetic overflow tests	0.050 s.	OK
medium1	0.050 s.	OK
medium2	0.060 s.	OK
medium3	0.090 s.	OK
medium4	0.130 s.	OK
10M_intersections 10.000.000 intersections	0.120 s.	OK
big1	0.050 s.	OK
big2	0.050 s.	OK
big3 [0]*50000	0.050 s.	OK

Training center