Open Feedback Dialog

codility

Training center

Check out Codility training tasks

Demo ticket

Session

ID: demoGTU9N3-873
Time limit: 120 min.

Status: closed

Created on: 2014-03-23 07:47 UTC
Started on: 2014-03-23 07:47 UTC
Finished on: 2014-03-23 07:48 UTC

Tasks in test

1 | {} ~~ArrayInversionCount~~

Task score

100%

Test score

100%

100 out of 100 points

MEDIUM

1. ArrayInversionCount

Compute number of inversion in an array.

score: 100 of 100

Task description

A zero-indexed array A consisting of N integers is given. An *inversion* is a pair of indexes (P, Q) such that P < Q and A[Q] < A[P].
Write a function:

```
def solution(A)
```

that computes the number of inversions in A, or returns −1 if it exceeds 1,000,000,000.
Assume that:

- N is an integer within the range [0..100,000];
- each element of array A is an integer within the range [−2,147,483,648..2,147,483,647].

For example, in the following array:

```
A[0] = -1 A[1] = 6 A[2] = 3
A[3] =  4 A[4] = 7 A[5] = 4
```

there are four inversions:

```
(1,2)  (1,3)  (1,5)  (4,5)
```

so the function should return 4.
Complexity:

- expected worst-case time complexity is O(N*log(N));
- expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

Solution

Programming language used: Python

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: correct functionality and scalability

Task timeline

07:47:34 07:48:25

Code: 07:48:24 UTC, py, final, score: **100.00**

```
01. def solution(A):
02.     return sort_count(A)[0]
03.
04. def sort_count(l):
05.     N = len(l)
06.     if N < 2:
07.         return 0, l
08.     mid = N / 2
09.     c1, l1 = sort_count(l[:mid])
10.     c2, l2 = sort_count(l[mid:])
11.     c3, l3 = merge_count(l1, l2)
12.     return c1 + c2 + c3, l3
13.
14. def merge_count(l1, l2):
15.     count = i = j = k = 0
16.     N = len(l1)
17.     M = len(l2)
18.     l3 = [0] * (N + M)
19.     while i < N and j < M:
20.         if l1[i] <= l2[j]:
21.             l3[k] = l1[i]
22.             i += 1
```

```
23.         else:
24.             l3[k] = l2[j]
25.             j += 1
26.             count += N - i
27.         k += 1
28.     while i < N:
29.         l3[k] = l1[i]
30.         i += 1
31.         k += 1
32.     while j < M:
33.         l3[k] = l2[j]
34.         j += 1
35.         k += 1
36.     return  count, l3
```

Analysis

Detected time complexity:

O(N*log(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_0_inv [0], [], [1,2,3], [1,1,1]	0.050 s.	OK
medium1 n=100	0.050 s.	OK
medium2 n=200	0.050 s.	OK
medium3 n=1000	0.050 s.	OK
big1 n=10000	0.140 s.	OK
big2 n=20000	0.250 s.	OK
big3 n=30000	0.360 s.	OK

Training center