Codility_

CodeCheck Report: training3VHGH8-QQ2

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxNonoverlappingSegments Python	5 min	100%

Total score

Tasks Details

1. MaxNonoverlappingSegments

Easy

Find a maximal set of non-overlapping segments.

Task Score	Correctness	Performance
100%	100%	100%

Task description

Located on a line are N segments, numbered from 0 to N − 1, whose positions are given in arrays A and B. For each I (0 ≤ I < N) the position of segment I is from A[I] to B[I] (inclusive). The segments are sorted by their ends, which means that B[K] ≤ B[K + 1] for K such that 0 ≤ K < N − 1.

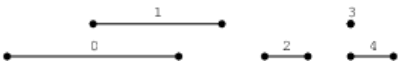
Two segments I and J, such that I ≠ J, are *overlapping* if they share at least one common point. In other words, A[I] ≤ A[J] ≤ B[I] or A[J] ≤ A[I] ≤ B[J].

We say that the set of segments is *non-overlapping* if it contains no two overlapping segments. The goal is to find the size of a non-overlapping set containing the maximal number of segments.

For example, consider arrays A, B such that:

```
A[0] = 1    B[0] = 5
A[1] = 3    B[1] = 6
A[2] = 7    B[2] = 8
A[3] = 9    B[3] = 9
A[4] = 9    B[4] = 10
```

The segments are shown in the figure below.

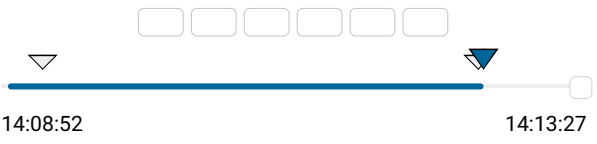


The size of a non-overlapping set containing a maximal number of segments is 3. For example, possible sets are {0, 2, 3}, {0, 2, 4}, {1, 2, 3} or {1, 2, 4}. There is no non-overlapping set with four segments.

Solution

Programming language used:	Python
Total time used:	5 minutes
Effective time used:	5 minutes
Notes:	*not defined yet*

Task timeline



14:08:52 14:13:27

Code: 14:13:27 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debugging purposes,
2  # print("this is a debug message")
3
4  def solution(A, B):
5      # Implement your solution here
6      # pass
7      N = len(A)
```

Write a function:

```
def solution(A, B)
```

that, given two arrays A and B consisting of N integers, returns the size of a non-overlapping set containing a maximal number of segments.

For example, given arrays A, B shown above, the function should return 3, as explained above.

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

- N is an integer within the range [0..30,000];
- each element of arrays A and B is an integer within the range [0..1,000,000,000];
- A[I] ≤ B[I], for each I (0 ≤ I < N);
- B[K] ≤ B[K + 1], for each K (0 ≤ K < N − 1).


```
8      if N == 0:
9          return 0
10
11     count = 1
12     prev_end = B[0]
13
14     for i in range(1, N):
15         if A[i] > prev_end:
16             count += 1
17             prev_end = B[i]
18
19     return count
20
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example example test	✔ OK
expand all	**Correctness tests**	
▶	extreme_empty_and_single empty and single element	✔ OK
▶	small_functional many overlapping	✔ OK
▶	small_non_overlapping all non-overlapping	✔ OK
▶	small_all_overlapping small functional	✔ OK
▶	small_random_same_length small random, length = ~40	✔ OK
expand all	**Performance tests**	
▶	medium_random_differ_length medium random, length = ~300	✔ OK
▶	large_points all points, length = ~30,000	✔ OK
▶	large_random_many_overlapping large random, length = ~30,000	✔ OK
▶	large_random_few_overlapping large random, length = ~30,000	✔ OK
▶	extreme_large large size of intervals, length = ~30,000	✔ OK