Codility_

CodeCheck Report: trainingNXPMP2-WPH

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxNonoverlappingSegments Java 8	18 min	100%

Total score

100%

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.

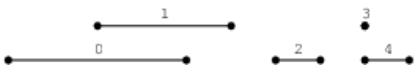
Two segments I and J, such that I ≠ J, are *overlapping* if they share at least one common point. In other words, A[I] ≤ A[J] ≤ B[I] or A[J] ≤ A[I] ≤ B[J].

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,

Solution

Programming language used:	Java 8
Total time used:	18 minutes
Effective time used:	18 minutes
Notes:	*not defined yet*

Task timeline

21:24:19 21:41:59

Code: 21:41:59 UTC, java, final, score: **100**

show code in pop-up

```
1  class Solution {
2          public int solution(int[] A, int[] B) {
3                  int N = A.length;
4
5                  int count = 0;
6
```

3} or {1, 2, 4}. There is no non-overlapping set with four segments.

Write a function:

```
class Solution { public int solution(int[] A,
int[] 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).


```
7                   int currentEnd = -1;
8                   for (int i = 0; i < N; i++) {
9                           if (A[i] > currentEnd) {
10                                  count++;
11                                  currentEnd = B[i
12                          }
13                  }
14                  return count;
15          }
16  }
```

Analysis summary

The solution obtained perfect score.

Analysis

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

collapse all **Example tests**

▼ example ✔ OK
example test

1. 0.004 s OK

collapse all **Correctness tests**

▼ extreme_empty_and_single ✔ OK
empty and single element

1. 0.008 s OK
2. 0.008 s OK

▼ small_functional ✔ OK
many overlapping

1. 0.004 s OK

▼ small_non_overlapping ✔ OK
all non-overlapping

1. 0.004 s OK
2. 0.008 s OK

▼ small_all_overlapping ✔ OK
small functional

1. 0.004 s OK
2. 0.004 s OK

▼ small_random_same_length ✔ OK
small random, length = ~40

1. 0.004 s OK

collapse all **Performance tests**

▼ medium_random_differ_length ✔ OK
medium random, length = ~300

1. 0.004 s OK

▼ large_points ✔ OK
all points, length = ~30,000

1. 0.160 s OK
2. 0.260 s OK

▼ large_random_many_overlapping large random, length = ~30,000	✔ OK
1. 0.220 s OK	
▼ large_random_few_overlapping large random, length = ~30,000	✔ OK
1. 0.216 s OK	
▼ extreme_large large size of intervals, length = ~30,000	✔ OK
1. 0.176 s OK	
2. 0.208 s OK	
3. 0.144 s OK	