


CodeCheck Report: trainingMAH26X-CFJ

[Check out Codility training tasks](#)

Test Name:

Summary Timeline

Tasks summary

Task		Time spent	Score
NailingPlanks Python		25 min	100%

Total score



Tasks Details

1. Medium	NailingPlanks			
	Count the minimum number of nails that allow a series of planks to be nailed.	Task Score	Correctness	Performance
		100%	100%	100%

Task description



You are given two non-empty arrays A and B consisting of N integers. These arrays represent N planks. More precisely, A[K] is the start and B[K] the end of the K-th plank.

Next, you are given a non-empty array C consisting of M integers. This array represents M nails. More precisely, C[I] is the position where you can hammer in the I-th nail.

We say that a plank (A[K], B[K]) is nailed if there exists a nail C[I] such that $A[K] \leq C[I] \leq B[K]$.

The goal is to find the minimum number of nails that must be used until all the planks are nailed. In other words, you should find a value J such that all planks will be nailed

Solution

Programming language used:	Python	
Total time used:	25 minutes	
Effective time used:	25 minutes	
Notes:	not defined yet	

Task timeline



after using only the first J nails. More precisely, for every plank $(A[K], B[K])$ such that $0 \leq K < N$, there should exist a nail $C[I]$ such that $I < J$ and $A[K] \leq C[I] \leq B[K]$.

For example, given arrays A, B such that:

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

four planks are represented: $[1, 4]$, $[4, 5]$, $[5, 9]$ and $[8, 10]$.

Given array C such that:

```
C[0] = 4
C[1] = 6
C[2] = 7
C[3] = 10
C[4] = 2
```

if we use the following nails:

- 0, then planks $[1, 4]$ and $[4, 5]$ will both be nailed.
- 0, 1, then planks $[1, 4]$, $[4, 5]$ and $[5, 9]$ will be nailed.
- 0, 1, 2, then planks $[1, 4]$, $[4, 5]$ and $[5, 9]$ will be nailed.
- 0, 1, 2, 3, then all the planks will be nailed.

Thus, four is the minimum number of nails that, used sequentially, allow all the planks to be nailed.

Write a function:

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

that, given two non-empty arrays A and B consisting of N integers and a non-empty array C consisting of M integers, returns the minimum number of nails that, used sequentially, allow all the planks to be nailed.

If it is not possible to nail all the planks, the function should return -1 .

For example, given arrays A, B, C such that:

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

```
C[0] = 4
C[1] = 6
C[2] = 7
C[3] = 10
C[4] = 2
```

the function should return 4, as explained above.

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

- N and M are integers within the range $[1..30,000]$;

08:13:09

08:37:39

Code: 08:37:39 UTC, py, [show code in pop-up](#)
final, score: 100

```
1 # you can write to stdout for debugging
2 # print("this is a debug message")
3
4 def solution(A, B, C):
5     # Implement your solution here
6     # pass
7
8     planks = sorted(list(zip(A, B)))
9
10    def check(mid):
11        nails = sorted(C[:mid])
12        j = 0
13
14        for nail in nails:
15            while j < len(planks) and pl
16                j += 1
17            if j == len(planks):
18                return True
19        return False
20
21    lower, upper = 1, len(C) + 1
22    result = -1
23    while lower < upper:
24        mid = (lower + upper) // 2
25        if check(mid):
26            upper = mid
27            result = mid
28        else:
29            lower = mid + 1
30
31    return result
32
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests
▶ example	✓ OK
example test	
expand all	Correctness tests
▶ extreme_single	✓ OK
single nail and single plank	

- each element of arrays A, B and C is an integer within the range $[1..2*M]$;
- $A[K] \leq B[K]$.

Copyright 2009–2023 by Codility Limited. All Rights Reserved.
Unauthorized copying, publication or disclosure prohibited.

▶ extreme_point	✓ OK
nail is a point [1, 1]	
▶ few_nails_in_the_same_place	✓ OK
ce few nails are in the same place	
▶ random_small	✓ OK
random sequence, length = ~100	
expand all Performance tests	
▶ random_medium	✓ OK
random sequence, length = ~10,000	
▶ random_large	✓ OK
random sequence, length = ~30,000	
▶ extreme_large_planks	✓ OK
all large planks, length = ~30,000	
▶ large_point	✓ OK
all planks are points, length = ~30,000	