Codility_

CodeCheck Report: trainingDADG3Z-E54

Test Name:

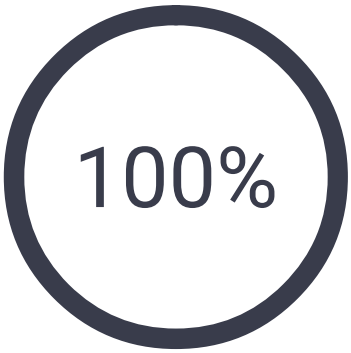
Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
NailingPlanks C	34 min	100%

Total score



Tasks Details

Medium

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

Solution

Programming language used:	C
Total time used:	34 minutes
Effective time used:	34 minutes
Notes:	*not defined yet*

Task timeline

07:50:24 08:23:55

Code: 08:23:55 UTC, c, final, score: **100** show code in pop-up

```
1  // you can write to stdout for debugging purposes,
2  // printf("this is a debug message\n");
3
```

```
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:

```
int solution(int A[], int B[], int N, int C[],
int M);
```

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


```
4  int solution(int A[], int B[], int N, int C[], int
5      int min_nails = 1;
6      int max_nails = M;
7      int mid;
8      int nails = -1;
9
10     // Possible nail position is 2 * M
11     int nailedCount = 2 * M + 1;
12     int nailed[2 * M + 1];
13
14     while (min_nails <= max_nails) {
15         for (int i = 0; i < nailedCount; ++i) {
16             nailed[i] = 0;
17         }
18
19         mid = (min_nails + max_nails) / 2;
20
21         for (int i = 0; i < mid; ++i) {
22             nailed[C[i]]++;
23         }
24
25         for (int i = 0; i < nailedCount; ++i) {
26             nailed[i + 1] += nailed[i];
27         }
28
29         int missing = 0;
30         for (int i = 0; i < N; ++i) {
31             if (nailed[A[i] - 1] == nailed[B[i]])
32                 // No nail exists for board i
33                 missing = 1;
34                 break;
35             }
36         }
37
38         if (missing) {
39             min_nails = mid + 1;
40         } else {
41             max_nails = mid - 1;
42             nails = mid;
43         }
44     }
45
46     return nails;
47 }
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example example test	✓ OK
expand all	Correctness tests	
▶	extreme_single single nail and single plank	✓ OK
▶	extreme_point nail is a point [1, 1]	✓ OK
▶	few_nails_in_the_same_place few nails are in the same place	✓ OK
▶		

random_small random sequence, length = ~100	✓ OK

expand all Performance tests

▶ random_medium random sequence, length = ~10,000	✓ OK
▶ random_large random sequence, length = ~30,000	✓ OK
▶ extreme_large_planks all large planks, length = ~30,000	✓ OK
▶ large_point all planks are points, length = ~30,000	✓ OK