Codility_

CodeCheck Report: trainingW2Q27V-ZW8

Test Name:

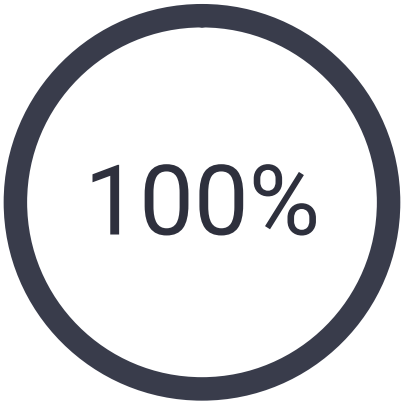
Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
Ladder Python	1 min	100%

Total score



Tasks Details

1. **Ladder**

Medium

Count the number of different ways of climbing to the top of a ladder.

Task Score	Correctness	Performance
100%	100%	100%

Task description

You have to climb up a ladder. The ladder has exactly N rungs, numbered from 1 to N. With each step, you can ascend by one or two rungs. More precisely:

- with your first step you can stand on rung 1 or 2,

Solution

Programming language used: Python

Total time used: 1 minutes

- if you are on rung K, you can move to rungs K + 1 or K + 2,
- finally you have to stand on rung N.

Your task is to count the number of different ways of climbing to the top of the ladder.

For example, given N = 4, you have five different ways of climbing, ascending by:

- 1, 1, 1 and 1 rung,
- 1, 1 and 2 rungs,
- 1, 2 and 1 rung,
- 2, 1 and 1 rungs, and
- 2 and 2 rungs.

Given N = 5, you have eight different ways of climbing, ascending by:

- 1, 1, 1, 1 and 1 rung,
- 1, 1, 1 and 2 rungs,
- 1, 1, 2 and 1 rung,
- 1, 2, 1 and 1 rung,
- 1, 2 and 2 rungs,
- 2, 1, 1 and 1 rungs,
- 2, 1 and 2 rungs, and
- 2, 2 and 1 rung.

The number of different ways can be very large, so it is sufficient to return the result modulo 2^P, for a given integer P.

Write a function:

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

that, given two non-empty arrays A and B of L integers, returns an array consisting of L integers specifying the consecutive answers; position I should contain the number of different ways of climbing the ladder with A[I] rungs modulo $2^{B[I]}$.

For example, given L = 5 and:

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

the function should return the sequence [5, 1, 8, 0, 1], as explained above.

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

- L is an integer within the range [1..50,000];
- each element of array A is an integer

Effective time used: 1 minutes

Notes: *not defined yet*

Task timeline

15:54:13 15:55:13

Code: 15:55:13 UTC, py, final, score: **100** show code in pop-up

```
1  # you can write to stdout for debugg
2  # print("this is a debug message")
3
4  def solution(A, B):
5      # Implement your solution here
6      # pass
7      L = len(A)
8      max_A = max(A)
9      max_B = max(B)
10
11     # Calculate the Fibonacci sequen
12     fib = [0] * (max_A + 2)
13     fib[1] = 1
14     for i in range(2, max_A + 2):
15         fib[i] = (fib[i - 1] + fib[i
16
17     # Calculate the number of ways f
18     result = [0] * L
19     for i in range(L):
20         result[i] = fib[A[i] + 1] %
21
22     return result
23
```

Analysis summary

The solution obtained perfect score.

Analysis

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

expand all	Example tests	
▶	example	✔
	example test	OK
expand all	Correctness tests	

within the range [1..L];
- each element of array B is an integer within the range [1..30].

▶ extreme extreme small values	✔ OK
▶ small_functional small functional	✔ OK
▶ small small tests	✔ OK
▶ small_random small random, length = ~100	✔ OK
expand all	**Performance tests**
▶ medium_random medium random, length = ~1,000	✔ OK
▶ large_range large range, length = ~30,000	✔ OK
▶ large_random large random, length = ~30,000	✔ OK
▶ extreme_large all max size of the ladder	✔ OK