Open Feedback Dialog

codility

Training center

Check out Codility training tasks

Demo ticket

Session

ID: demoRBDFW9-6AY
Time limit: 120 min.

Status: closed

Created on: 2014-03-17 05:11 UTC
Started on: 2014-03-17 05:11 UTC
Finished on: 2014-03-17 05:12 UTC

Tasks in test

Task score

Test score

100%

100 out of 100 points

MEDIUM

1. Ladder

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

score: 100 of 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,
- 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 zero-indexed arrays A and B of L integers, returns an array consisting of L integers specifying the consecutive

Solution

Programming language used: Python

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: correct functionality and scalability

Task timeline

05:11:59 05:12:43

Code: 05:12:43 UTC, py, final, score: **100.00**

```
01. def solution(A, B):
02.     L = len(A)
03.     mask = (1 << 30) - 1
04.     ways = [0] * (L + 1)
05.     ways[1] = 1
06.     if L > 1: ways[2] = 2
07.     for i in xrange(3, L + 1):
08.         ways[i] = (ways[i-1] + ways[i-2]) & mask
09.
10.     ret = [0] * L
11.     for i in xrange(L):
12.         ret[i] = ways[A[i]] & ((1 << B[i]) - 1)
13.
14.     return ret
```

Analysis

Detected time complexity:

O(L)

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.
Assume that:

- L is an integer within the range [1..30,000];
- each element of array A is an integer within the range [1..L];
- each element of array B is an integer within the range [1..30].

Complexity:

- expected worst-case time complexity is O(L);
- expected worst-case space complexity is O(L), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

test	time	result
example example test	0.050 s.	**OK**
extreme extreme small values	0.050 s.	**OK**
small_functional small functional	0.050 s.	**OK**
small small tests	0.050 s.	**OK**
small_random small random, length = ~100	0.050 s.	**OK**
medium_random medium random, length = ~1,000	0.050 s.	**OK**
large_range large range, length = ~30,000	0.220 s.	**OK**
large_random large random, length = ~30,000	0.220 s.	**OK**
extreme_large all max size of the ladder	0.220 s.	**OK**

Training center