Open Feedback Dialog

codility

Training center

Check out Codility training tasks

Demo ticket

Session

ID: demoKZDQKK-G9P
Time limit: 120 min.

Status: closed

Created on: 2014-03-17 04:05 UTC
Started on: 2014-03-17 04:05 UTC
Finished on: 2014-03-17 04:18 UTC

Tasks in test

Task score

Test score

100%

100 out of 100 points

MEDIUM

1. FibFrog

Count the minimum number of jumps required for a frog to get to the other side of a river.

score: 100 of 100

Task description

The Fibonacci sequence is defined using the following recursive formula:

```
F(0) = 0
F(1) = 1
F(M) = F(M - 1) + F(M - 2) if M >= 2
```

A small frog wants to get to the other side of a river. The frog is initially located at one bank of the river (position −1) and wants to get to the other bank (position N). The frog can jump over any distance F(K), where F(K) is the K-th Fibonacci number. Luckily, there are many leaves on the river, and the frog can jump between the leaves, but only in the direction of the bank at position N.
The leaves on the river are represented in a zero-indexed array A consisting of N integers. Consecutive elements of array A represent consecutive positions from 0 to N − 1 on the river. Array A contains

only 0s and/or 1s:

- 0 represents a position without a leaf;
- 1 represents a position containing a leaf.

The goal is to count the minimum number of jumps in which the frog can get to the other side of the river (from position −1 to position N). The frog can jump between positions −1 and N (the banks of the river) and every position containing a leaf.
For example, consider array A such that:

```
A[0] = 0
A[1] = 0
A[2] = 0
A[3] = 1
A[4] = 1
A[5] = 0
A[6] = 1
A[7] = 0
A[8] = 0
A[9] = 0
A[10] = 0
```

The frog can make three jumps of length F(5) = 5, F(3) = 2 and F(5) =

Solution

Programming language used: Python

Total time used: 14 minutes

Effective time used: 14 minutes

Notes: correct functionality and scalability

Task timeline

04:05:17 04:18:46

Code: 04:18:46 UTC, py, final, score: **100.00**

```
01. def solution(A):
02.     # write your code in Python 2.6
03.     N = len(A)
04.     if N < 3: return 1
05.
06.     fab = [0] * N
07.     fab[0] = 1
08.     fab[1] = 2
09.     for i in xrange(2, N):
10.         fab[i] = fab[i-1] + fab[i-2]
11.         if fab[i] > N: break
12.
13.     INF = sys.maxint
14.     steps = [INF] * (N + 1)
15.     for i in xrange(N + 1):
16.         if i < N and A[i] == 0:
17.             continue
18.         for j in fab:
19.             last = i - j
20.             if last == -1:
21.                 steps[i] = 1
22.                 break
```

5.
Write a function:

```
def solution(A)
```

that, given a zero-indexed array A consisting of N integers, returns the minimum number of jumps by which the frog can get to the other side of the river. If the frog cannot reach the other side of the river, the function should return −1.
For example, given:

```
A[0] = 0
A[1] = 0
A[2] = 0
A[3] = 1
A[4] = 1
A[5] = 0
A[6] = 1
A[7] = 0
A[8] = 0
A[9] = 0
A[10] = 0
```

the function should return 3, as explained above.
Assume that:

- N is an integer within the range [0..100,000];
- each element of array A is an integer that can have one of the following values: 0, 1.

Complexity:

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

Elements of input arrays can be modified.


```
23.             elif last >= 0 and A[last] == 1:
24.                 steps[i] = min(steps[i], steps[last] +
                        1)
25.             elif last < -1:
26.                 break
27.
28.     return steps[N] if steps[N] < INF else -1
```

Analysis

Detected time complexity:

O(N * log(N))

test	time	result
example example test	0.050 s.	OK
extreme_small_ones empty array and all ones	0.050 s.	OK
extreme_small_zeros all zeros	0.050 s.	OK
simple_functional simple functional tests	0.050 s.	OK
small_random small random test, length = ~100	0.050 s.	OK
small_cyclic small cyclic test, length = ~500	0.050 s.	OK
small_fibonacci small Fibonacci word test, length = 610	0.050 s.	OK
medium_random medium random test, length = ~5,000	0.080 s.	OK
medium_thue_morse medium Thue-Morse sequence, lenght = 2^13	0.100 s.	OK
large_big_result large test with big result, length = ~100,000	0.190 s.	OK
large_cyclic large cyclic test, length = ~100,000	0.270 s.	OK
large_random large random test, length = ~100,000	0.760 s.	OK
extreme_large_ones_zeros all zeros / ones, length = ~100,000	1.700 s.	OK

Training center