Open Feedback Dialog

codility

Training center

Check out Codility training tasks

Demo ticket

Session

ID: demoVW6VZF-HFR
Time limit: 120 min.

Status: closed

Created on: 2014-03-23 04:07 UTC
Started on: 2014-03-23 04:07 UTC
Finished on: 2014-03-23 04:20 UTC

Tasks in test

1 | {} ~~FallingDisks~~

Task score

100%

Test score

100%

100 out of 100 points

MEDIUM

1. FallingDisks

Determine the number of disks that fit into the well.

score: 100 of 100

Task description

There is an old dry well. Its sides are made of concrete rings. Each such ring is one meter high, but the rings can have different (internal) diameters. Nevertheless, all the rings are centered on one another. The well is N meters deep; that is, there are N concrete rings inside it. You are about to drop M concrete disks into the well. Each disk is one meter thick, and different disks can have different diameters. Once each disk is dropped, it falls down until:

- it hits the bottom of the well;
- it hits a ring whose internal diameter is smaller then the disk's diameter; or
- it hits a previously dropped disk.

(Note that if the internal diameter of a ring and the diameter of a disk are equal, then the disk can fall through the ring.)
The disks you are about to drop are ready and you know their

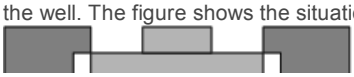
diameters, as well as the diameters of all the rings in the well. The question arises: how many of the disks will fit into the well?
Write a function:

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

that, given two zero-indexed arrays of integers − A, containing the internal diameters of the N rings (in top-down order), and B, containing the diameters of the M disks (in the order they are to be dropped) − returns the number of disks that will fit into the well.
For example, given the following two arrays:

```
A[0] = 5    B[0] = 2
A[1] = 6    B[1] = 3
A[2] = 4    B[2] = 5
A[3] = 3    B[3] = 2
A[4] = 6    B[4] = 4
A[5] = 2
A[6] = 3
```

the function should return 4, as all but the last of the disks will fit into the well. The figure shows the situation after dropping four disks.



Solution

Programming language used: Python

Total time used: 14 minutes

Effective time used: 14 minutes

Notes: correct functionality and scalability

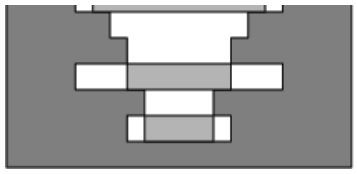
Task timeline

04:07:37 04:20:40

Code: 04:20:40 UTC, py, final, score: **100.00**

```
01. def solution(A, B):
02.     N = len(A)
03.     M = len(B)
04.     A = A[::-1]
05.
06.     for i in xrange(N-2, -1, -1):
07.         if A[i] > A[i+1]:
08.             A[i] = A[i+1]
09.
10.     i = 0
11.     j = 0
12.     while i < N and j < M:
13.         if A[i] >= B[j]:
14.             j += 1
15.         i += 1
16.
17.     return j
```

Analysis



Assume that:

- N is an integer within the range [1..200,000];
- M is an integer within the range [1..200,000];
- each element of array A is an integer within the range [1..1,000,000,000];
- each element of array B is an integer within the range [1..1,000,000,000].

Complexity:

- expected worst-case time complexity is O(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.

Detected time complexity:
O(N)

test	time	result
example example from the problem statement	0.050 s.	**OK**
too_big disk too big to reach its potential place	0.050 s.	**OK**
no_space no space for one disk	0.050 s.	**OK**
simple one disk every 2 positions	0.050 s.	**OK**
boundary_cases1 boundary cases tests, one disk, one level	0.050 s.	**OK**
boundary_cases2 boundary cases tests, one disk	0.050 s.	**OK**
boundary_cases3 boundary cases, disk too large	0.050 s.	**OK**
boundary_cases4 boundary cases, disk too large	0.050 s.	**OK**
boundary_cases5 boundary cases tests, M > N	0.050 s.	**OK**
medium_full pipe of length 1000 full of disks	0.050 s.	**OK**
every_second_level one disk every 2 positions, N=5000	0.060 s.	**OK**
large_full pipe of length 10000 full of disks	0.090 s.	**OK**
medium_random random data, N=50,000, M=20,000	0.190 s.	**OK**
max_full pipe of max size full of disks	0.810 s.	**OK**
max_random maximum size random data	0.830 s.	**OK**

Training center