Congratulations

You have completed a Codility training test.

Sign up for our newsletter!

Like us on Facebook!

Training ticket

Session

ID: trainingDPTVC9-ZQM
Time limit: 120 min.

Status: closed

Created on: 2016-06-11 20:28 UTC
Started on: 2016-06-11 20:28 UTC
Finished on: 2016-06-11 20:29 UTC

Tasks in test	Correctness	Performance	Task score
1 Fish Submitted in: Java	100%	100%	100%

Test score

100%

100 out of 100 points

EASY

1. Fish

N voracious fish are moving along a river. Calculate how many fish are alive.

score: 100 of 100

Task description

You are given two non-empty zero-indexed arrays A and B consisting of N integers. Arrays A and B represent N voracious fish in a river, ordered downstream along the flow of the river.

The fish are numbered from 0 to N − 1. If P and Q are two fish and P < Q, then fish P is initially upstream of fish Q. Initially, each fish has a unique position.

Fish number P is represented by A[P] and B[P]. Array A contains the sizes of the fish. All its elements are unique. Array B contains the directions of the fish. It contains only 0s and/or 1s, where:

- 0 represents a fish flowing upstream,
- 1 represents a fish flowing downstream.

If two fish move in opposite directions and there are no other (living) fish between them, they will eventually meet each other. Then only one fish can stay alive − the larger fish eats the smaller one. More precisely, we say that two fish P and Q meet each other when P < Q, B[P] = 1 and B[Q] = 0, and there are no living fish between them. After they meet:

- If A[P] > A[Q] then P eats Q, and P will still be flowing downstream,
- If A[Q] > A[P] then Q eats P, and Q will still be flowing upstream.

We assume that all the fish are flowing at the same speed. That is, fish moving in the same direction never meet. The goal is to calculate the number of fish that will stay alive.

For example, consider arrays A and B such that:

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

Solution

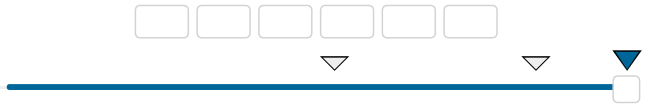
Programming language used: Java

Total time used: 1 minutes

Effective time used: 1 minutes

Notes: *not defined yet*

Task timeline



20:28:59 20:29:33

Code: 20:29:33 UTC, java, final, score: **100**

show code in pop-up

```
1  // you can also use imports, for example:
2  // import java.util.*;
3
4  // you can write to stdout for debugging purposes, e.g.
5  // System.out.println("this is a debug message");
6
7  class Solution {
8    public int solution(int[] A, int[] B) {
9      int f = -1; // previous i value
10     int sum=A.length;
11     int jump=0;
12     int i = 0;
13     int j = -1;
14
15     for (; i < B.length && j < B.length;) {
```

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

Initially all the fish are alive and all except fish number 1 are moving upstream. Fish number 1 meets fish number 2 and eats it, then it meets fish number 3 and eats it too. Finally, it meets fish number 4 and is eaten by it. The remaining two fish, number 0 and 4, never meet and therefore stay alive.

Write a function:

```
class Solution { public int solution(int[] A, int[] B); }
```

that, given two non-empty zero-indexed arrays A and B consisting of N integers, returns the number of fish that will stay alive.

For example, given the arrays shown above, the function should return 2, as explained above.

Assume that:

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

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.


```
16
17        if (B[i] == 1) {
18          if (f > -1) jump++;
19          f = i;
20        }
21        if (B[i] == -1 || f < 0) {
22          i++;
23          continue;
24        }
25
26        if (i>=j) j = i+1;
27
28        for (; j < B.length;) {
29          if (B[j] == -1) {
30            j++;
31            continue;
32          }
33          if (B[j] == 1) {
34            i++;
35            break;
36          }
37
38          sum--;
39
40          if(A[i] > A[j]) {
41            B[j] = -1;
42            j++;
43          } else {
44            B[i] = -1;
45            f = -1;
46            // search first "1" back if jump > 0
47            if (jump>0) {
48              for (;i >= 0;i--) {
49                if (B[i] == 1) {
50                  jump--;
51                  break;
52                }
53              }
54            } else {
55              i = j + 1;
56            }
57            break;
58          }
59        }
60      }
61      return sum;
62    }
63  }
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity:
O(N)

expand all	Example tests	
▶	example example test	✔ OK
expand all	Correctness tests	
▶	extreme_small 1 or 2 fishes	✔ OK
▶	simple1 simple test	✔ OK
▶	simple2 simple test	✔ OK
▶	small_random small random test, N = ~100	✔ OK
expand all	Performance tests	
▶	medium_random small medium test, N = ~5,000	✔ OK
▶	large_random large random test, N = ~100,000	✔ OK
▶	extreme_range1 all except one fish flowing in the same	✔ OK

	direction	
▶	extreme_range2 all fish flowing in the same direction	✔ OK

Training center