closed

Demo ticket

Session
ID: demoB7PRGN-C9U
Time limit: 120 min.

Status: closed
Started on: 2014-01-12 15:26 UTC

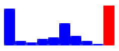
Score:

100

of 100

★★ 1. Fish

score: 100 of 100



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

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, fish number P is represented by A[P] and B[P], and if P < Q then fish P is initially upstream of fish Q. Initially, each fish has a unique position.
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
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, numbers 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.

Solution

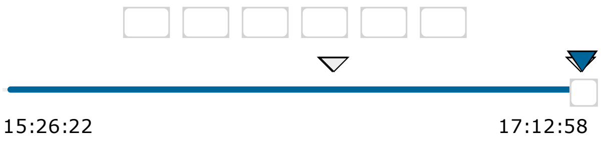
Programming language used: C#

Total time used: 107 minutes

Effective time used: 47 minutes

Notes: correct functionality and scalability

Task timeline



15:26:22 17:12:58

Code: 17:12:58 UTC, cs, final, score: 100.00

```
using System;
using System.Collections.Generic;
class Solution {
        public int solution(int[] A, int[] B)
        {
            // write your code in C# with .NET 2.0
            if(A == null || A.Length < 1 || A.Length > 100000) throw new ArgumentOutOfRangeException();

            var result = 0;
            var downStreamFishes = new Stack<int>();
            for (var count = 0; count < A.Length; count++) {
                if(B[count] == 0) {
                    while(downStreamFishes.Count > 0 && downStreamFishes.Peek() < A[count]) {
                        downStreamFishes.Pop();
                    }
                    if(downStreamFishes.Count == 0) result++;
                } else {
                    downStreamFishes.Push(A[count])
                }
            }

            result += downStreamFishes.Count;

            return result;
        }
}
```

Elements of input arrays can be modified.

Analysis

test	time	result
example example test	0.080 s.	**OK**
extreme_small 1 or 2 fishes	0.080 s.	**OK**
simple1 simple test	0.080 s.	**OK**
simple2 simple test	0.080 s.	**OK**
small_random small random test, N = ~100	0.080 s.	**OK**
medium_random small medium test, N = ~5,000	0.080 s.	**OK**
large_random large random test, N = ~100,000	0.140 s.	**OK**
extreme_range1 all except one fish flowing in the same direction	0.120 s.	**OK**
extreme_range2 all fish flowing in the same direction	0.150 s.	**OK**

Get acco