Codility_

Candidate Report: training99E3BE-XS7

Test Name:

Summary   Review (0)   Timeline

Tasks summary

Task	Time spent	Score
FrogJmp Java 8	2 min	100%

Total score

100%

Tasks Details

Easy

1. FrogJmp
Count minimal number of jumps from position X to Y.

	Task Score	Correctness	Performance
	100%	100%	100%

Task description

A small frog wants to get to the other side of the road. The frog is currently located at position X and wants to get to a position greater than or equal to Y. The small frog always jumps a fixed distance, D.

Count the minimal number of jumps that the small frog must perform to reach its target.

Write a function:

```
class Solution { public int solution(int X, int Y, int D); }
```

that, given three integers X, Y and D, returns the minimal number of jumps from position X to a position equal to or greater than Y.

For example, given:

```
X = 10
Y = 85
D = 30
```

the function should return 3, because the frog will be positioned as follows:

- after the first jump, at position 10 + 30 = 40
- after the second jump, at position 10 + 30 + 30 = 70
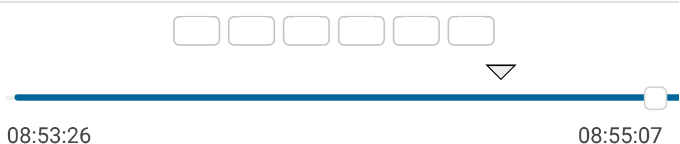- after the third jump, at position 10 + 30 + 30 + 30 = 100

Write an **efficient** algorithm for the following assumptions:

- X, Y and D are integers within the range [1..1,000,000,000];

Solution

Programming language used:	Java 8
Total time used:	2 minutes
Effective time used:	2 minutes
Notes:	*not defined yet*

Task timeline

08:53:26   08:55:07

Code: 08:55:07 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 X, int Y, int D) {
```

- X ≤ Y.

```
 9          if(X==Y)
10              return 0;
11          else if(D>(Y-X))
12              return 1;
13          else{
14              int diff = Y-X;
15              int count = diff/D;
16              if((diff%D)>0)
17                  return count+1;
18              else
19                  return count;
20          }
21      }
22  }
```

Analysis summary

The solution obtained perfect score.

Analysis ❓

Detected time complexity:

O(1)

expand all	Example tests	
▶ example	✓ OK	
example test		
expand all	Correctness tests	
▶ simple1	✓ OK	
simple test		
▶ simple2	✓ OK	
▶ extreme_position	✓ OK	
no jump needed		
▶ small_extreme_jump	✓ OK	
one big jump		
expand all	Performance tests	
▶ many_jump1	✓ OK	
many jumps, D = 2		
▶ many_jump2	✓ OK	
many jumps, D = 99		
▶ many_jump3	✓ OK	
many jumps, D = 1283		
▶ big_extreme_jump	✓ OK	
maximal number of jumps		
▶ small_jumps	✓ OK	
many small jumps		

The PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.