codility

Training center

Check out Codility training tasks

Congratulations

You have completed a Codility demo.

Tweet this!
I scored 100% in #java on @Codility!
https://codility.com/demo/take-sample-test/frog_jmp/

Sign up for our newsletter!

Like us on Facebook!

Demo ticket

Session

ID: demoBZBP66-567
Time limit: 120 min.

Status: closed

Created on: 2015-03-20 04:44 UTC
Started on: 2015-03-20 04:44 UTC
Finished on: 2015-03-20 04:45 UTC

Tasks in test	Correctness	Performance	Task score
1 | FrogJmp	100%	100%	100%

Test score

100%

100 out of 100 points

EASY

1. FrogJmp

Count minimal number of jumps from position X to Y.

score: 100 of 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

Assume that:

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

Complexity:

- expected worst-case time complexity is O(1);

Solution

Programming language used: Java

Total time used: 2 minutes

Effective time used: 2 minutes

Notes: *not defined yet*

Task timeline

04:44:42 04:45:51

Code: 04:45:51 UTC, java, final, score: **100.00**
show code in pop-up

```
1  // you can also use imports, for example:
2  // import java.util.*;
3
4  // you can use System.out.println for debugging purpose
5  // System.out.println("this is a debug message");
6
7  class Solution {
8      public int solution(int X, int Y, int D) {
9          // write your code in Java SE 8
```

- expected worst-case space complexity is O(1).


```
10
11          int curr = (Y-X)/D;
12          if((Y-X)%D > 0)
13              return curr +1;
14          else
15              return curr;
16
17      }
18  }
```

Analysis

Detected time complexity:

O(1)

test	time	result
Example tests		
example example test	0.928 s	**OK**
Correctness tests		
simple1 simple test	0.956 s	**OK**
simple2	1.060 s	**OK**
extreme_position no jump needed	1.056 s	**OK**
small_extreme_jump one big jump	1.056 s	**OK**
Performance tests		
many_jump1 many jumps, D = 2	1.064 s	**OK**
many_jump2 many jumps, D = 99	1.064 s	**OK**
many_jump3 many jumps, D = 1283	1.052 s	**OK**
big_extreme_jump maximal number of jumps	1.056 s	**OK**
small_jumps many small jumps	1.056 s	**OK**

Training center