


Summary Timeline

Total score



	Task Score	Correctness	Performance
Easy	1. FrogJump Count minimal number of jumps from position X to Y.	100%	100%

Solution

Programming language used:	Python	
Total time used:	63 minutes	?
Effective time used:	63 minutes	?
Notes:	<i>not defined yet</i>	

Write a function:

```
def solution(X, Y, D)
```

Task timeline ?

X = 10
Y = 85
D = 30

Code: 05:31:46 UTC, py,
final, score: 100

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

```
1 # you can write to stdout for debugging purposes,
2 # print("this is a debug message")
3
4 # <釐清問題>
```

- 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 $[1..1,000,000,000]$;
- $X \leq Y$.

Copyright 2009–2021 by Codility Limited. All Rights Reserved. Unauthorized copying, publication or disclosure prohibited.

```

5  # 計算從起始座標(X=10)抵達某一座標(Y=85)至少需要多少
6  # e.g.
7  # X = 10 (起始座標)
8  # Y = 85 (須越過的target)
9  # D = 30 (每次跳躍的distance)
10 #
11 # 最終位置 = 10+(30*3)= 100 > 85(Y的位置),
12 # 表示X需要跳3次才能跳過Y, 故return3
13
14
15 # <定義知識&製作解決方案>
16 # 用三個變數做算術運算,並在X未移後的最終位置>Y的位置
17
18 # <驗證&改進>
19 # 可能的測資:
20 # - (1,1,1)
21 # - (10, 1000000000, 11) -> TIMEOUT ERROR (K
22 # - (1,1000000000,1)
23 # - (31, 44648891, 13) # print(31+13*3434528
24
25 # 需減少運算時間:
26 # -
27
28 # 其他限制:
29 # X, Y and D are integers within the range [
30 #  $X \leq Y$  (X只會右移動或不移動)
31
32 def solution(X, Y, D):
33     return countMinNumOfJumpsFast(X,Y,D)
34
35 def countMinNumOfJumps(start:int, target:int, dist
36     final_position = start
37     cnt_jumps = 0
38     while final_position<target:
39         final_position += distance
40         cnt_jumps += 1
41     return cnt_jumps
42
43 def countMinNumOfJumpsFast(start:int, target:int,
44     # 核心:
45     # 起始位置+ (位移*位移次數)= 最終位置 > Y的位置
46     #
47     # - 減少迴圈運算改善效能
48
49     tmp = target - start
50     jumps = int( tmp / distance)
51     if tmp%distance != 0: # 若整除則終
52         jumps += 1
53
54     return jumps
55
56 # When using:
57 # jumps = round(tmp/distance + 0.5)
58
59 # Get the wrong answer:
60 # Your test case: [1, 1000000000, 1]
61 # Returned value: 1000000000

```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **$O(1)$**

collapse all

Example tests

▼ example

✓ OK

example test

1.	0.036 s	OK
collapse all		Correctness tests
▼	simple1 simple test	✓ OK
1.	0.036 s	OK
2.	0.036 s	OK
▼	simple2	✓ OK
1.	0.036 s	OK
2.	0.036 s	OK
▼	extreme_position no jump needed	✓ OK
1.	0.036 s	OK
2.	0.036 s	OK
▼	small_extreme_jump one big jump	✓ OK
1.	0.036 s	OK
collapse all		Performance tests
▼	many_jump1 many jumps, D = 2	✓ OK
1.	0.036 s	OK
▼	many_jump2 many jumps, D = 99	✓ OK
1.	0.036 s	OK
▼	many_jump3 many jumps, D = 1283	✓ OK
1.	0.036 s	OK
▼	big_extreme_jump maximal number of jumps	✓ OK
1.	0.036 s	OK
▼	small_jumps many small jumps	✓ OK
1.	0.036 s	OK

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.