Back

[Python3] Clear, short and easy DP solution, No Need for BFS or math

richardCorona ★ 65 Last Edit: September 23, 2019 8:49 PM 7.1K VIEWS

64

The key observation is that we do not need to distinguish x and y, and we don't care whether x and y are positive or negative at all.

I believe it's easier than BFS or math solutions

```
from functools import lru_cache
class Solution:
    def minKnightMoves(self, x: int, y: int) -> int:
        @lru_cache(None)
        def DP(x,y):
            if x + y == 0:
                return 0
            elif x + y == 2:
                return 2
            return min(DP(abs(x-1),abs(y-2)),DP(abs(x-2),abs(y-1)))+1
        return DP(abs(x),abs(y))
```

Thanks @kuznecpl for pointing out that we need to take care of the (1,1) and (2,0) cases properly.

dynamic programming trick

Comments: 30 Best Most Votes Newest to Oldest Oldest to Newest

Type comment here... (Markdown is supported)

Post

zwSsx ★ 32 January 9, 2020 5:33 AM

Thanks! Share the Java version

```
class Solution {
public int minKnightMoves(int x, int y) {
    x = Math.abs(x);
    y = Math.abs(y);
    Map<String, Integer> memo = new HashMap<>();
    return helper(x, y, memo);
}

private int helper(int x, int y, Map<String, Integer> memo) {
```

Read More

9 Show 2 replies Reply

jacot ★ 32 March 15, 2020 7:27 AM

```
    def dfs(x,y):
        if (x,y) not in memo:
            x,y = abs(x),abs(y)
            if x == y == 0:
                return 0
            if x + y == 2:
                return 2
            ans = min(dfs(x-1,y-2),dfs(x-2,y-1))+1
            memo[(x,y)] = ans
        return memo[(x,y)]
```

Read More

6 Show 1 reply Reply

ngu ★ 18 September 30, 2019 2:28 AM

Thanks for sharing, have a doubt though, why does simply do recursion on x-1 and x-2 is correct? why don't we need to handle x+1 and x+2?

6 Show 3 replies Reply

Chen Xiang ★ 63 October 16, 2019 6:33 AM

Great idea, Java version

```
class Solution {
    public int minKnightMoves(int x, int y) {
        int MOD = Math.abs(y) + 2;
        return dfs(Math.abs(x), Math.abs(y), new HashMap<>(), MOD);
    }

    public int dfs(int x, int y, Map<Integer, Integer> map, int MOD) {
        int index = x * MOD + y;
        if (map.containsKey(index)) {
```

Read More

8 Show 3 replies Reply

zsz1990ustc ★ 56 February 28, 2020 10:48 AM

I am still confusing why you only consider DP(abs(x-1),abs(y-2)),DP(abs(x-2),abs(y-1)), but not DP(abs(x+1),abs(y+2)),DP(abs(x+2),abs(y+1)) or DP(abs(x-