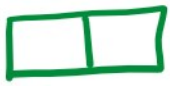
05-09 Recursion Problem - 3

05 September 2022 20:09

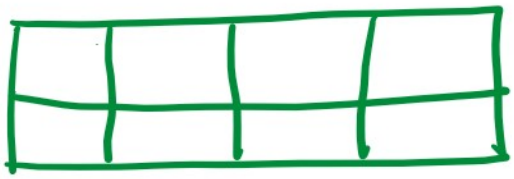
Tiling Problem:-

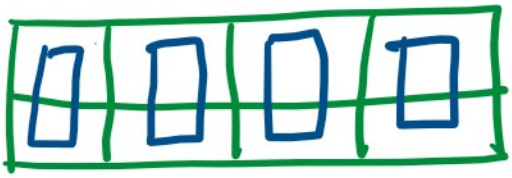
Given with 2*n board and tiles of size "2*1"
Count no. of ways to the tile so that you can fill the board

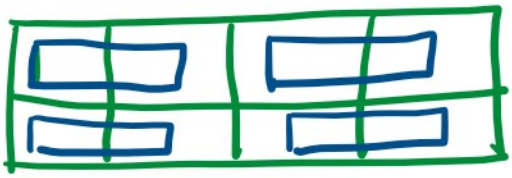
2 ways
- Horizontal  ✓
- Verticle  ✓

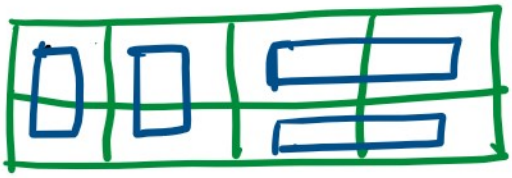
n=4

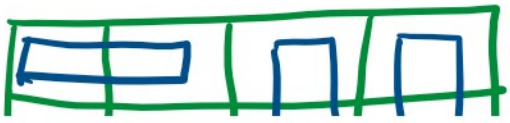
2xn ⇒ board size
2x4 ⇒ board size.

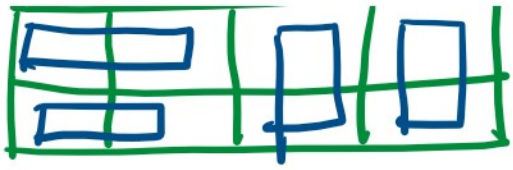


① 
All tile vertically.

② 
All tile horizontally.

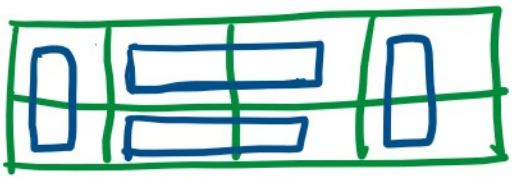
③ 
2V+2H

④ 
2H+2V



2 H + 2 V

⑤

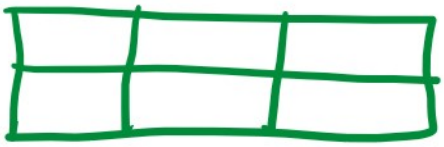


1 V + 2 H + 1 V

Total no. of ways = ⑤

---

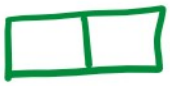
n = 3 $\quad 2 \times n \Rightarrow 2 \times 3$



Pen-Paper

??? $\quad$ 2 x 1

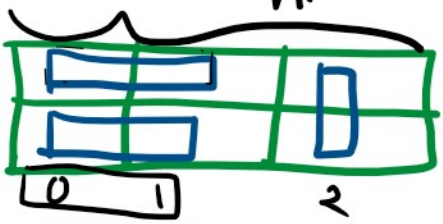
2 ways

- Horizontal  ✓
- Vertical  ✓

① 

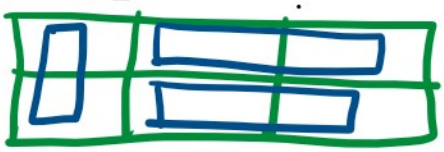
⇒ All vertical

n.

② 

⇒ 2 H + 1 V

0 1 2

③ 

⇒ 1 V + 2 H

Total ⇒ 3

n = 1 $\quad$ 2 x 1  ⇒ ①

n=2 2x2

=> 2

Base case

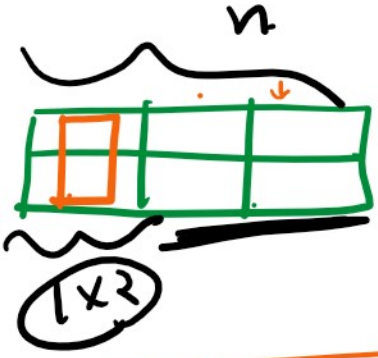
n==1 || n==2 // return n

n<=2
↳ return n; | n>0

Recursive Call

n=3

Countways(n)
↳ Countway(n-2) // 1
Countway(n-1) // 2
∴ 2



=> n-2

10 => 8

Q Given an int no. n
↓
return => how many minimum steps to convert
n -> 1

n=7 -> 1

1. Subtract n by 1
2. if n divided by 2, divide no. by 2.
3. if n divided by 3, divide no. by 3.

---

n=7 7-1 = 6/3 = 2/2 = 1
(1) (2) (3)

O/P => 3 steps

O/P => 3 steps

n = 10 → 10/2 = 5 - 1 = 4/2 = 2/2 = 1

[4 steps]

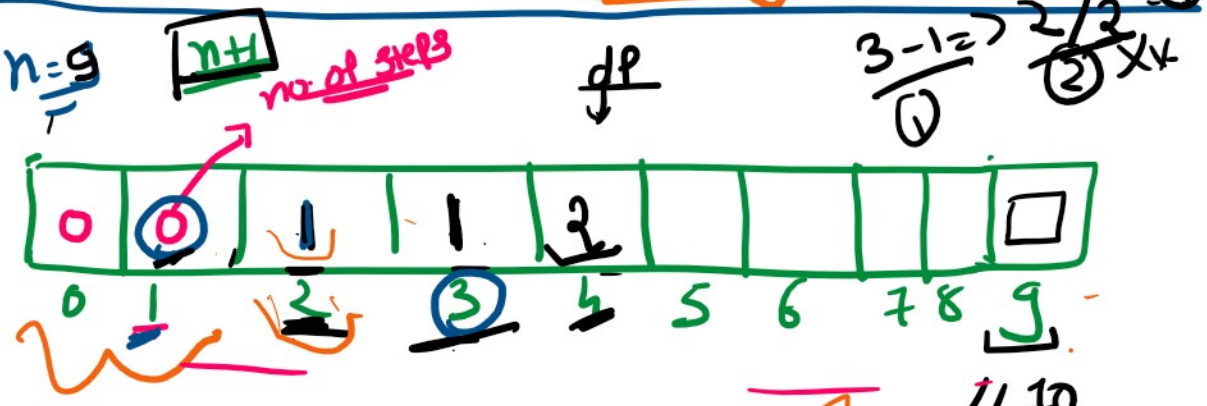
10 - 1 = 9/3 = 3/3 = 1

[3 steps]

3 => 3/3 ✓ → 1

3 - 1 = 2 → 2/3 ✗ → 2 → 1

n = 9



① dp[i] = dp[i-1] + 1

dp[2] = dp[1] + 1

dp[i] = 0 + 1

(2, dp(2)+1) → (2, 1+1) → (2, 2)

② min (dp[i], dp[i/2] + 1)

min (1, dp(2/2) + 1)

min (1, dp(1) + 1) => (1, 0+1) = (1, 1)

min (1, 1) => ①

② min (dp[i] , dp [i/3] +1)

③ min(dp[i], dp[i/3]+1)

min(2, dp(3/3)+1)

min(2, dp(1)+1)

min(2, 0+1) =>

min(2,1) => ①

Q ATM 33 868 33

0 2000 - 0
500 - 1
200 - 1
100 - 1
50 - 1
20 - 0
10 - 1
5 - 1
1 - 3

868
-500
368
-200
168
-100
68
-50
18
-10
8
-5
3

868

500-1
200-1
100-1

amount = 868

notes: [2000, 500, 200, 100, 50, 20, 10, 5, 1]

noteCounter: [0, 1, 1, 0, 0, 0, 0, 0, 0]

i=2

for => 0 to 9

368 >= 200

```
if(amount >= notes[i])
{
```

{

noteCounter[i] = floor(amount / notes[i])

amount = amount - notesCounter[i] * notes[i]

}

364/200

364 - 200 x 1 => 164