

Write a recursive function that takes as input a
non-negative number (num) and returns the number of
occurrences of the digit 1. count0nes(123114) = 3

```
def count0nes(num):
```

Write a recursive function that takes as input a
non-negative number (num) and returns the number of
occurrences of the digit 1. count0nes(123114) = 3

```
def count0nes(num):  
    if num == 0:  
        return 0  
    else:
```

Write a recursive function that takes as input a
non-negative number (num) and returns the number of
occurrences of the digit 1. count0nes(123114) = 3

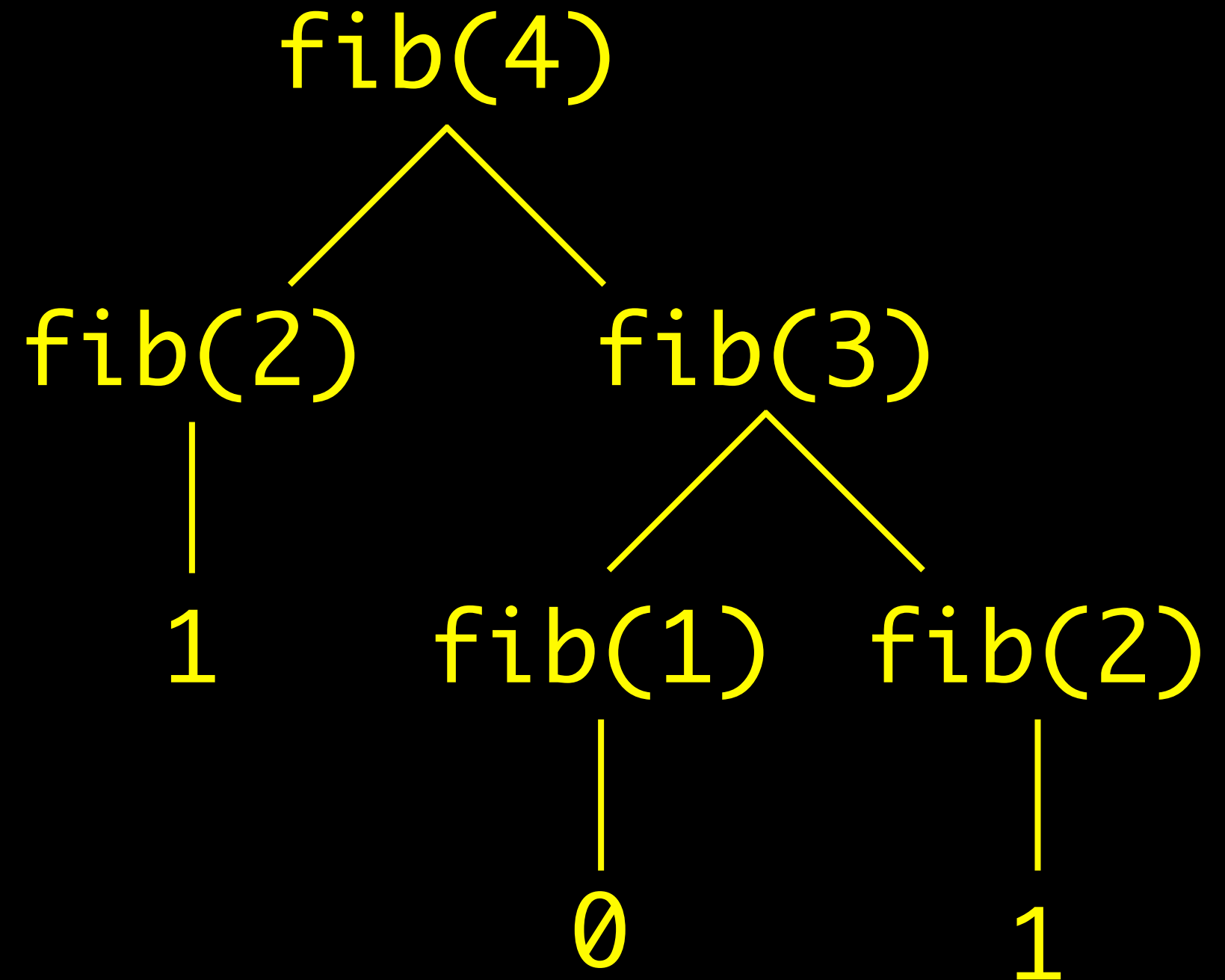
```
def count0nes(num):  
    if num == 0:  
        return 0  
    else:  
        if num % 10 == 1:  
            return 1 + count0nes(num//10)
```

Write a recursive function that takes as input a
non-negative number (num) and returns the number of
occurrences of the digit 1. count0nes(123114) = 3

```
def count0nes(num):  
    if num == 0:  
        return 0  
    else:  
        if num % 10 == 1:  
            return 1 + count0nes(num//10)  
        else:  
            return count0nes(num//10)
```

Tree recursion: Fibonacci sequence

```
def fib(n):  
    if n == 1:  
        return 0  
    elif n == 2:  
        return 1  
    else:  
        return fib(n-2) + fib(n-1)
```



Tree recursion: count partitions

The number of partitions of a positive integer n , using parts up to size m , is the number of ways in which n can be expressed as the sum of positive integer parts up to m in non-decreasing order.

cp(4, 2)

1 + 1 + 1 + 1

1 + 1 + 2

2 + 2

Tree recursion: count partitions

cp(6,4)

1 + 1 + 1 + 1 + 1 + 1

1 + 1 + 1 + 1 + 2

1 + 1 + 2 + 2

2 + 2 + 2

1 + 1 + 1 + 3

1 + 2 + 3

3 + 3

1 + 1 + 4

2 + 4

Tree recursion: count partitions

cp(6,4)

1 + 1 + 1 + 1 + 1 + 1 # don't use 4

1 + 1 + 1 + 1 + 2

1 + 1 + 2 + 2

2 + 2 + 2

1 + 1 + 1 + 3

1 + 2 + 3

3 + 3

1 + 1 + 4

2 + 4 # use 4

Tree recursion: count partitions

cp(6,4)

1 + 1 + 1 + 1 + 1 + 1 # don't use 4: cp(6,3)

1 + 1 + 1 + 1 + 2

1 + 1 + 2 + 2

2 + 2 + 2

1 + 1 + 1 + 3

1 + 2 + 3

3 + 3

1 + 1 + 4

2 + 4

use 4: cp(6-4,4)

Tree recursion: count partitions

cp(6,4)

1 + 1 + 1 + 1 + 1 + 1 # don't use 3: cp(6,2)

1 + 1 + 1 + 1 + 2

1 + 1 + 2 + 2

2 + 2 + 2

1 + 1 + 1 + 3 # use 3: cp(6-3,3)

1 + 2 + 3

3 + 3

Tree recursion: partitions

```
def cp(n, m):  
    if n == 0:  
        return 1  
    elif n < 0 or m == 0:  
        return 0  
    else:  
        return + cp(n, m-1) + cp(n-m, m)
```

mutual recursion: Luhn sum (check sum)

7 9 9 2 7 3 9 8 7 1 3 # acct number

mutual recursion: Luhn sum (check sum)

7 9 9 2 7 3 9 8 7 1 3 # acct number

18 4 6 16 2 # double every other

mutual recursion: Luhn sum (check sum)

7 9 9 2 7 3 9 8 7 1 3 # acct number

18 4 6 16 2 # double every other

9 4 6 7 2 # sum digits > 10

mutual recursion: Luhn sum (check sum)

7 9 9 2 7 3 9 8 7 1 3 # acct number

18 4 6 16 2 # double every other

9 4 6 7 2 # sum digits > 10

7 +9 +9 +4 +7 +6 +9 +7 +7 +2 +3 = 70 # sum

mutual recursion: Luhn sum (check sum)

7 9 9 2 7 3 9 8 7 1 3 # acct number

18 4 6 16 2 # double every other

9 4 6 7 2 # sum digits > 10

7 +9 +9 +4 +7 +6 +9 +7 +7 +2 +3 = 70 # sum

70 % 10 == 0 # valid Luhn sum is multiple of 10

mutual recursion: Luhn sum (check sum)

7 9 9 2 7 3 9 8 7 1 3

18 4 6 16 2

9 4 6 7 2

7 +9 +9 +4 +7 +6 +9 +7 +7 +2 +3 = 70

luhn_sum(79927398713)

luhn_sum2(7992739871) + 3

mutual recursion: Luhn sum (check sum)

7 9 9 2 7 3 9 8 7 1 3

18 4 6 16 2

9 4 6 7 2

7 +9 +9 +4 +7 +6 +9 +7 +7 +2 +3 = 70

luhn_sum(79927398713)

luhn_sum2(7992739871) + 3

luhn_sum(799273987) + sum_dig(2*1)

mutual recursion: Luhn sum (check sum)

7 9 9 2 7 3 9 8 7 1 3

18 4 6 16 2

9 4 6 7 2

7 +9 +9 +4 +7 +6 +9 +7 +7 +2 +3 = 70

luhn_sum(79927398713)

luhn_sum2(7992739871) + 3

luhn_sum(799273987) + sum_dig(2*1)

luhn_sum2(79927398) + 7

mutual recursion: Luhn sum (check sum)

7 9 9 2 7 3 9 8 7 1 3

18 4 6 16 2

9 4 6 7 2

7 +9 +9 +4 +7 +6 +9 +7 +7 +2 +3 = 70

luhn_sum(79927398713)

luhn_sum2(7992739871) + 3

luhn_sum(799273987) + sum_dig(2*1)

luhn_sum2(79927398) + 7

luhn_sum(7992739) + sum_dig(2*8)

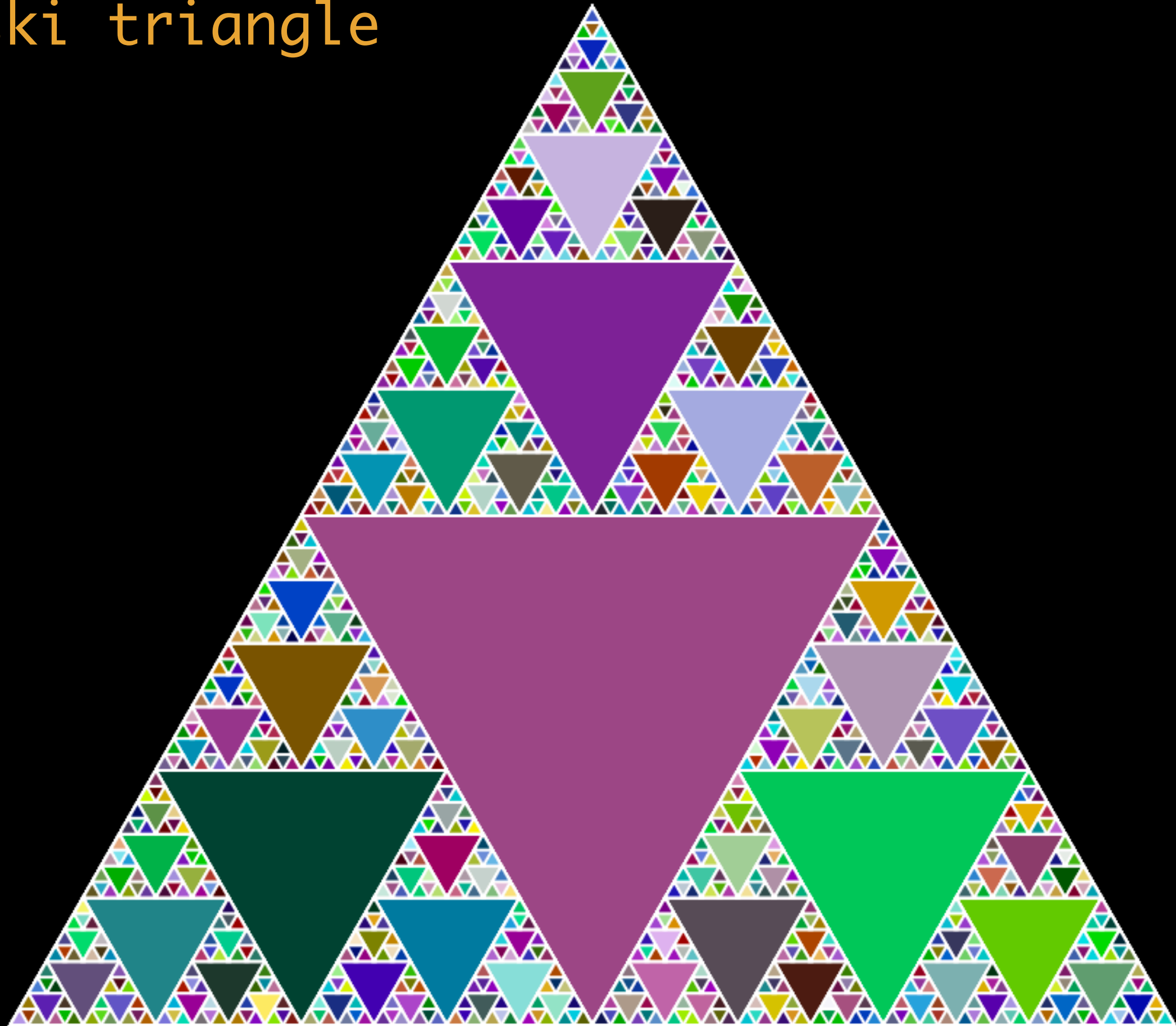
```
def split(n):  
    # Split a positive integer into all but its last digit and  
    # its last digit  
    # split(123) -> (123 // 10 = 12, 123 % 10 = 3)  
    return n // 10, n % 10
```

```
def sum_digits(n):  
    # Return the sum of the digits of positive integer n  
    if n < 10:  
        return n  
    else:  
        a, b = split(n)  
        return sum_digits(a) + b
```

```
def luhn_sum(n):  
    if n < 10:  
        return n  
    else:  
        a, b = split(n)  
        return luhn_sum2(a) + b
```

```
def luhn_sum2(n):  
    a, b = split(n)  
    d = sum_digits(2 * b)  
    if n < 10:  
        return d  
    else:  
        return luhn_sum(a) + d
```

Sierpiński triangle

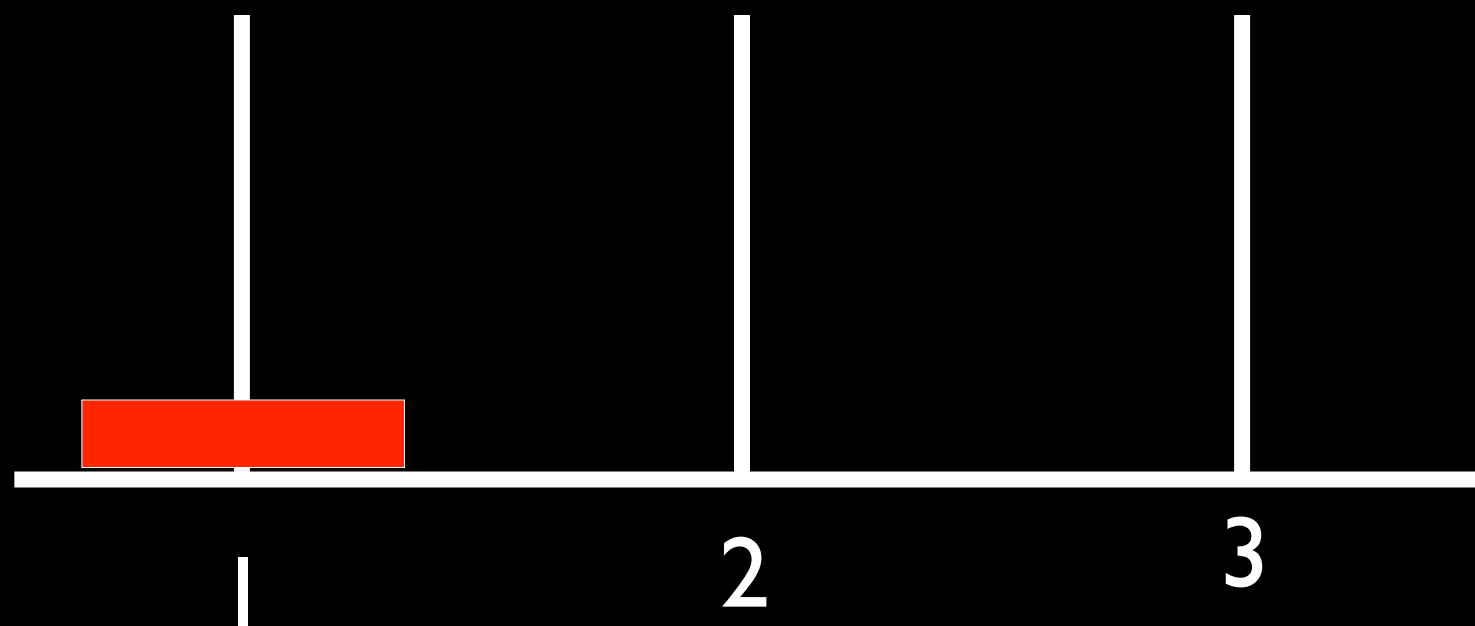


Towers of Hanoi

<http://haubergs.com/hanoi>

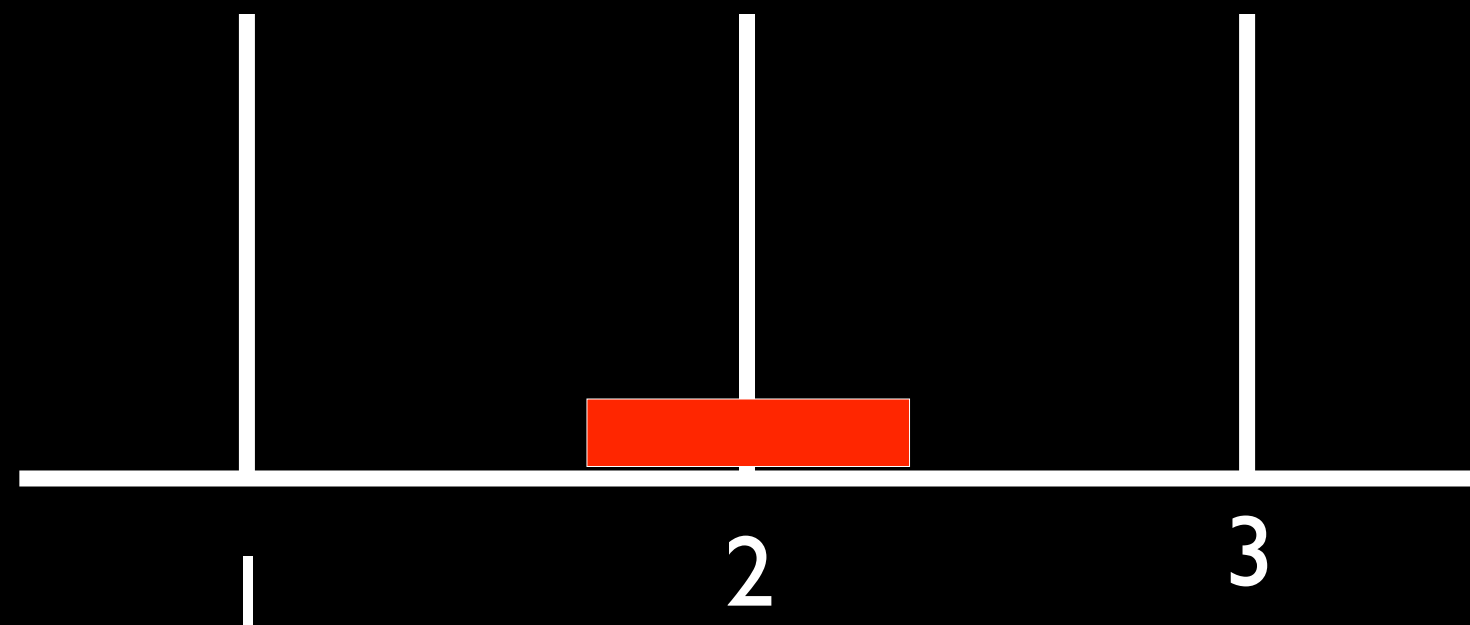
Towers of Hanoi

$n = 1$: move disk from post 1 to post 2



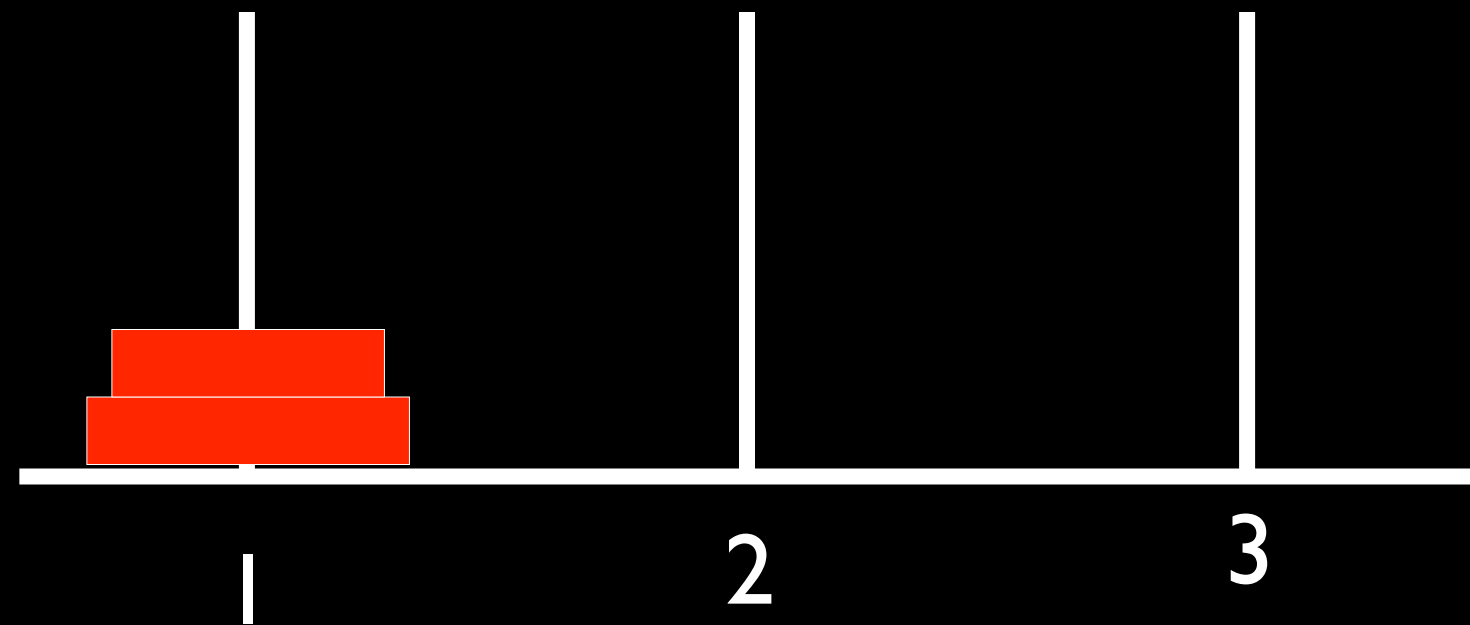
Towers of Hanoi

$n = 1$: move disk from post 1 to post 2



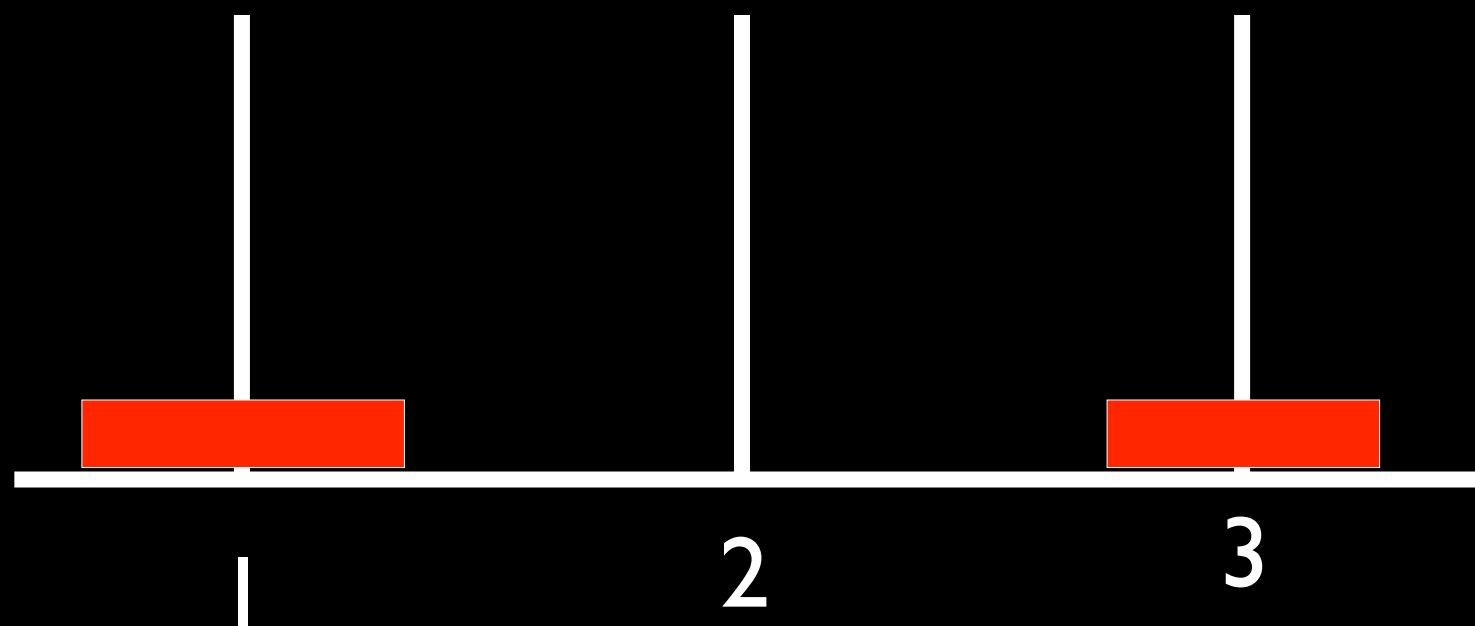
Towers of Hanoi

$n = 2$: move disks from post 1 to post 2



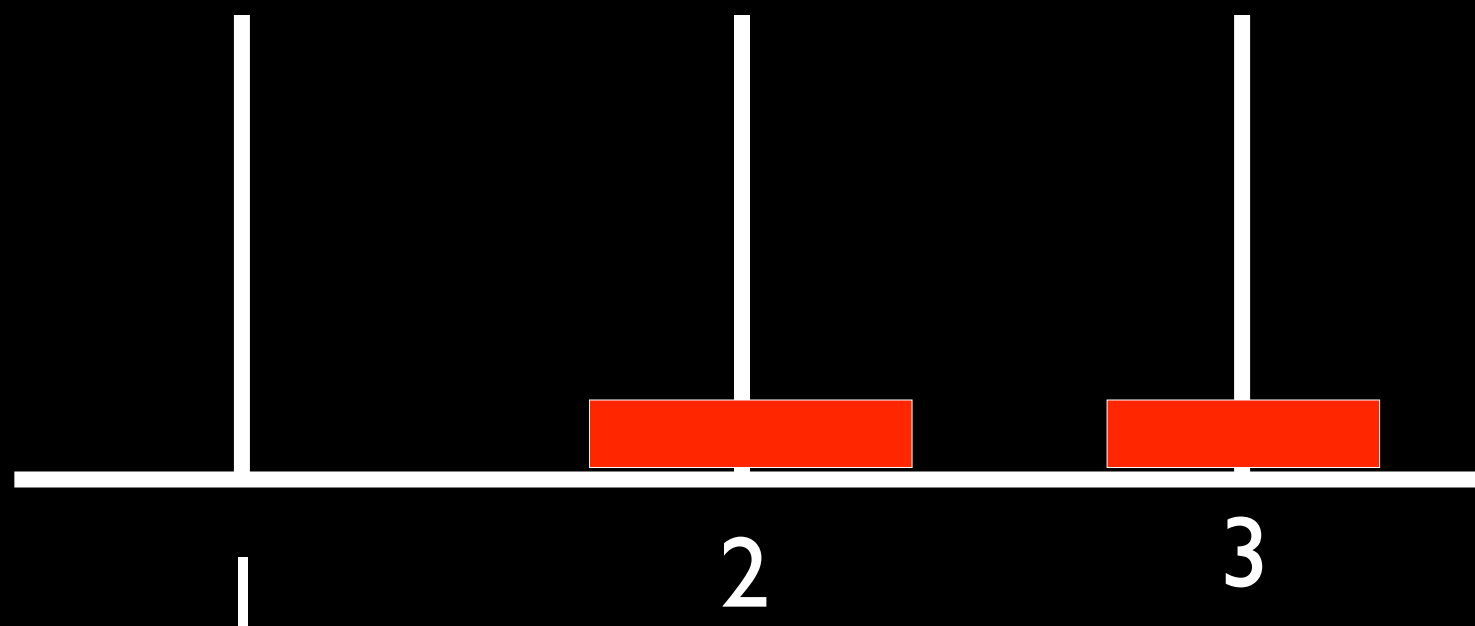
Towers of Hanoi

$n = 2$: move disks from post 1 to post 2



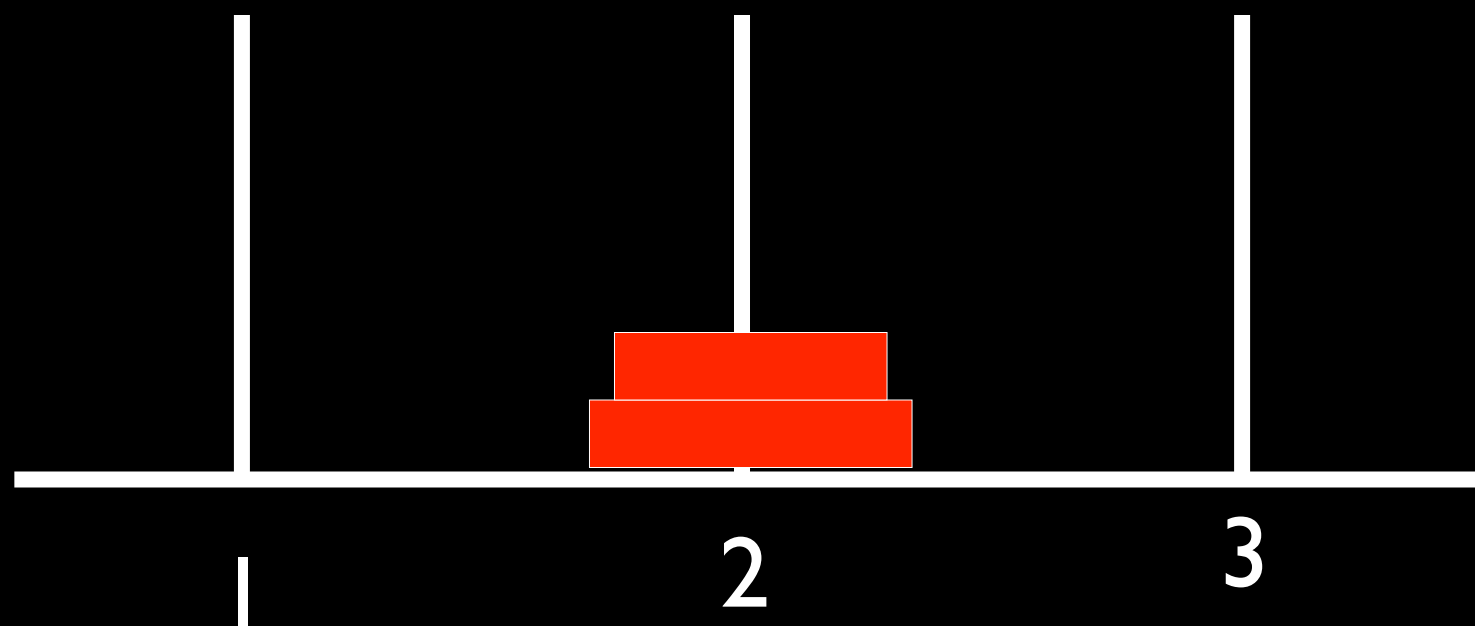
Towers of Hanoi

$n = 2$: move disks from post 1 to post 2



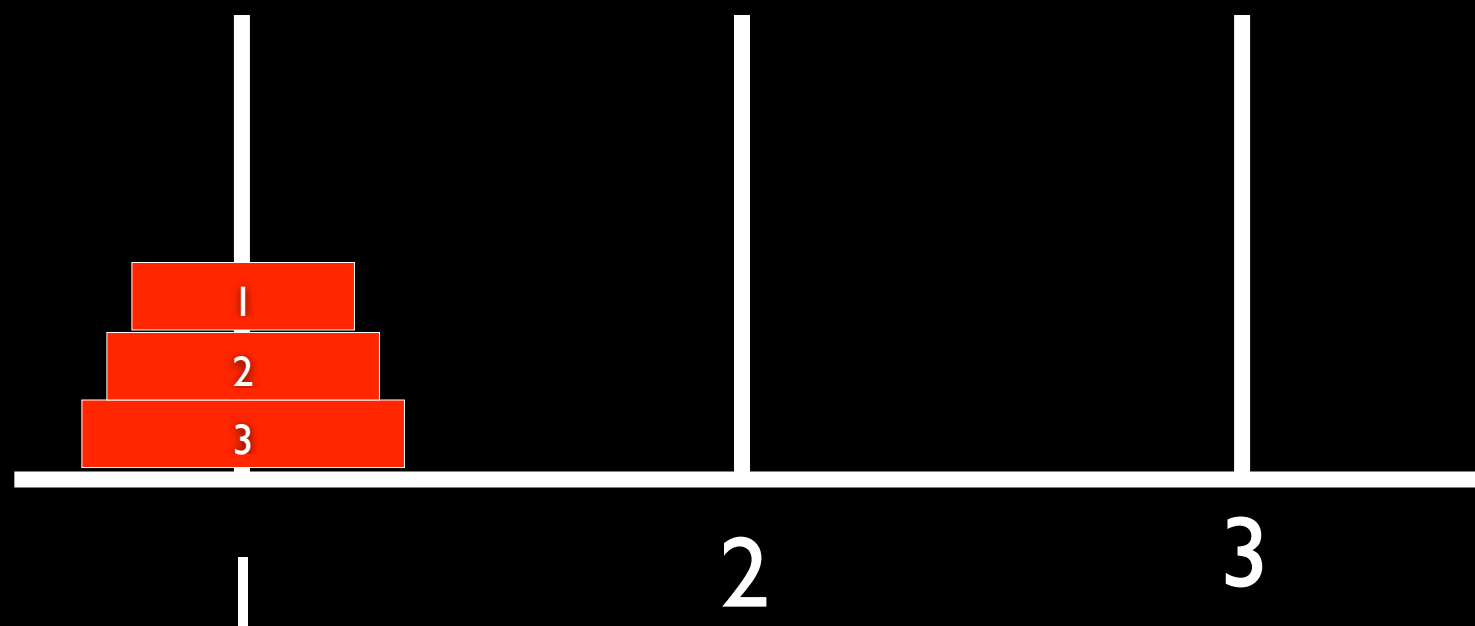
Towers of Hanoi

$n = 2$: move disks from post 1 to post 2



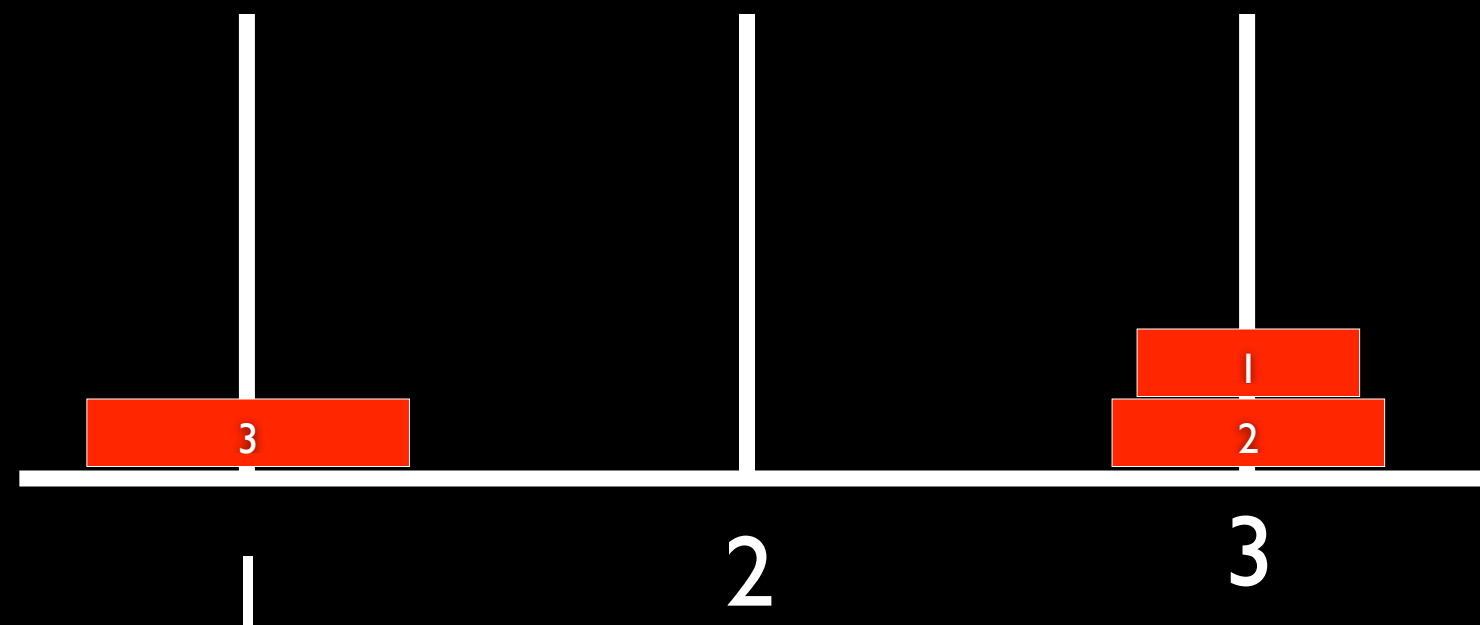
Towers of Hanoi

$n = 3$: move disks from post 1 to post 2



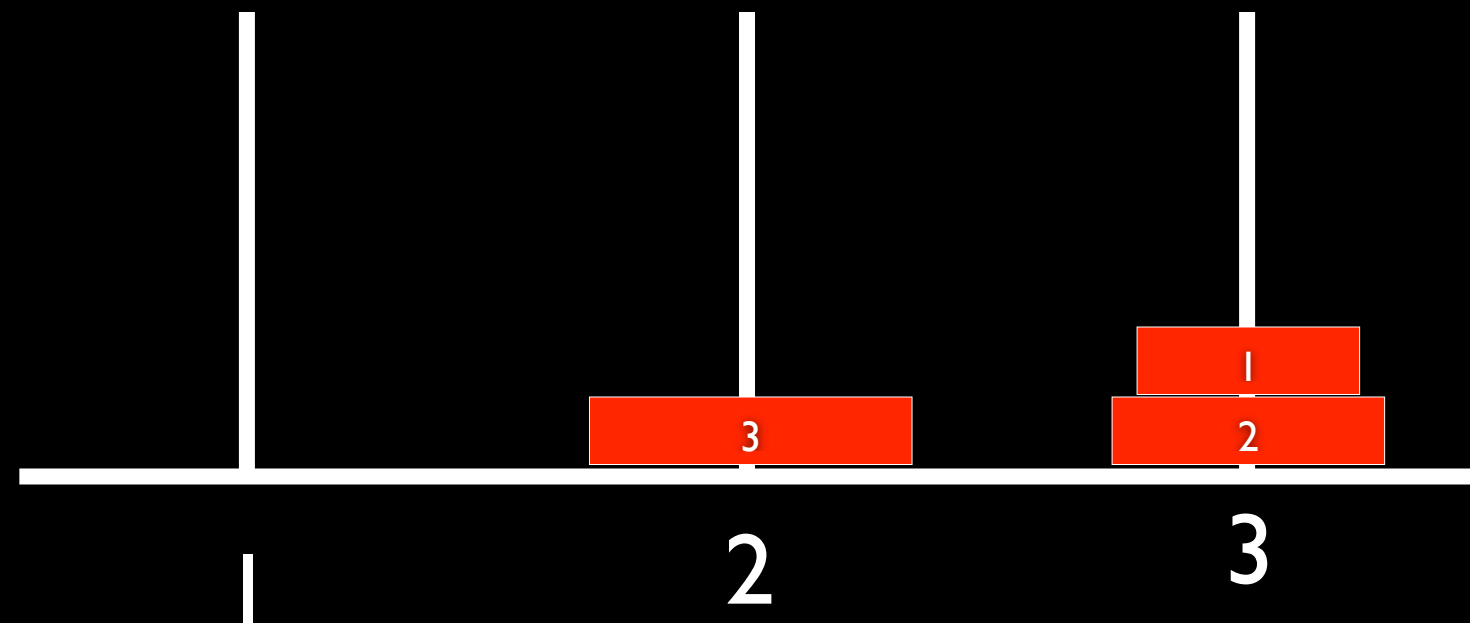
Towers of Hanoi

n = 3: move disks 1 & 2 from post 1 to 3



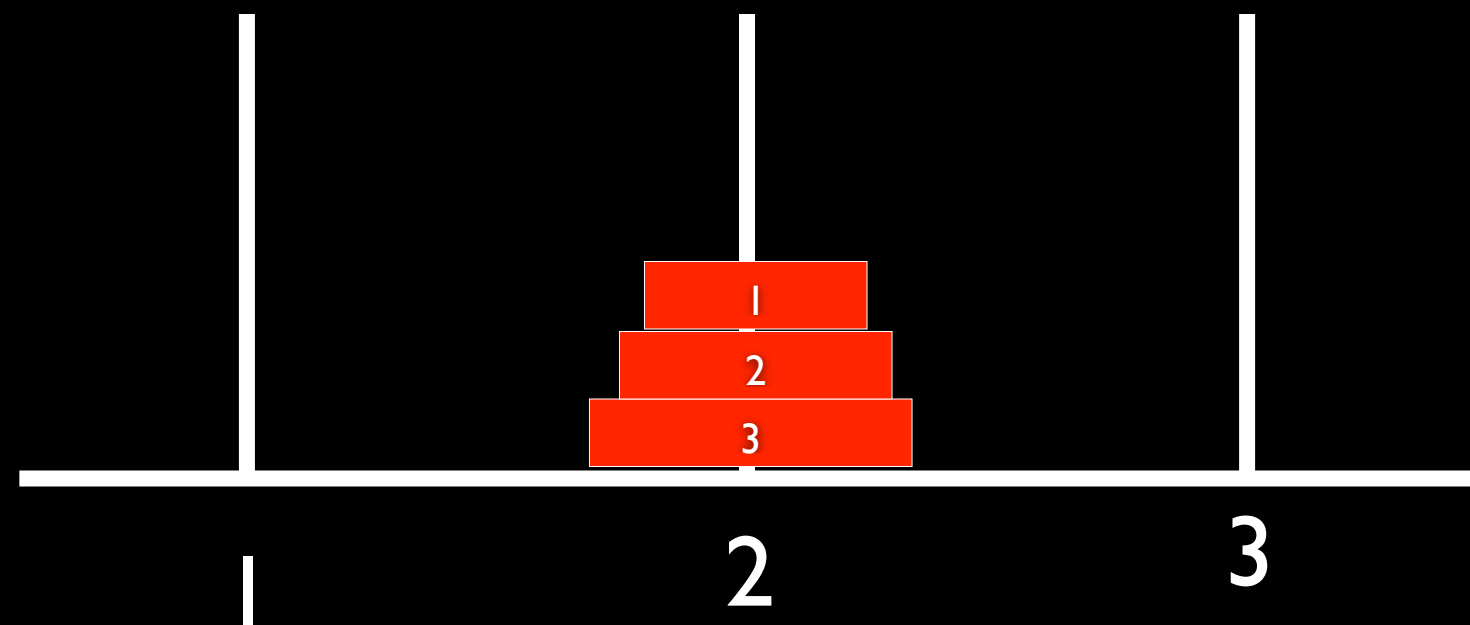
Towers of Hanoi

n = 3: move disks 3 from post 1 to 2

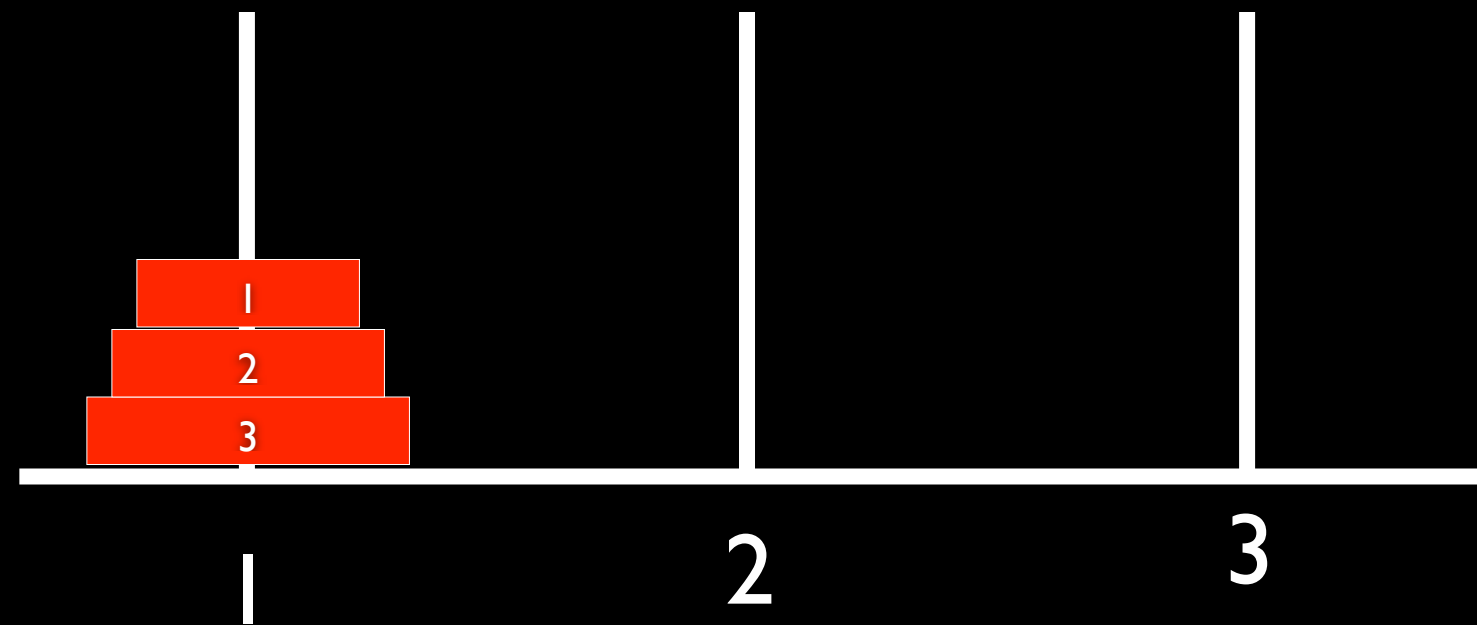


Towers of Hanoi

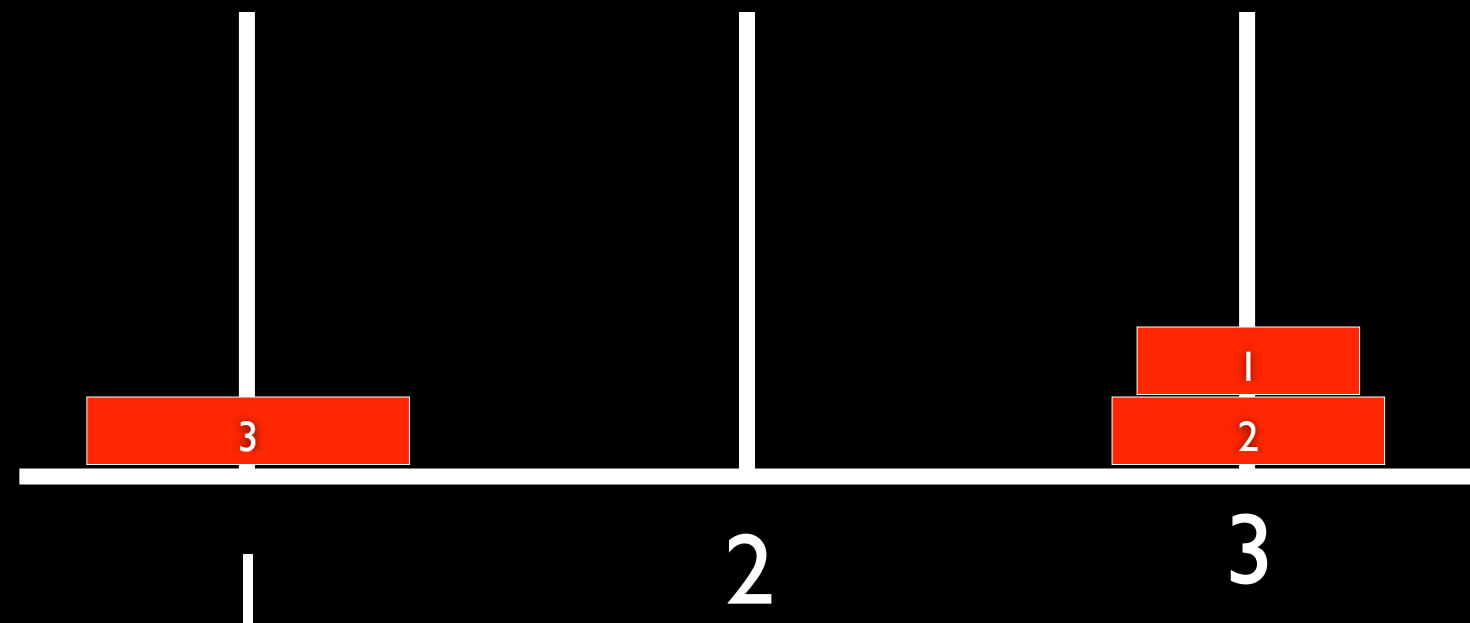
n = 3: move disks 1 & 2 from post 3 to 2



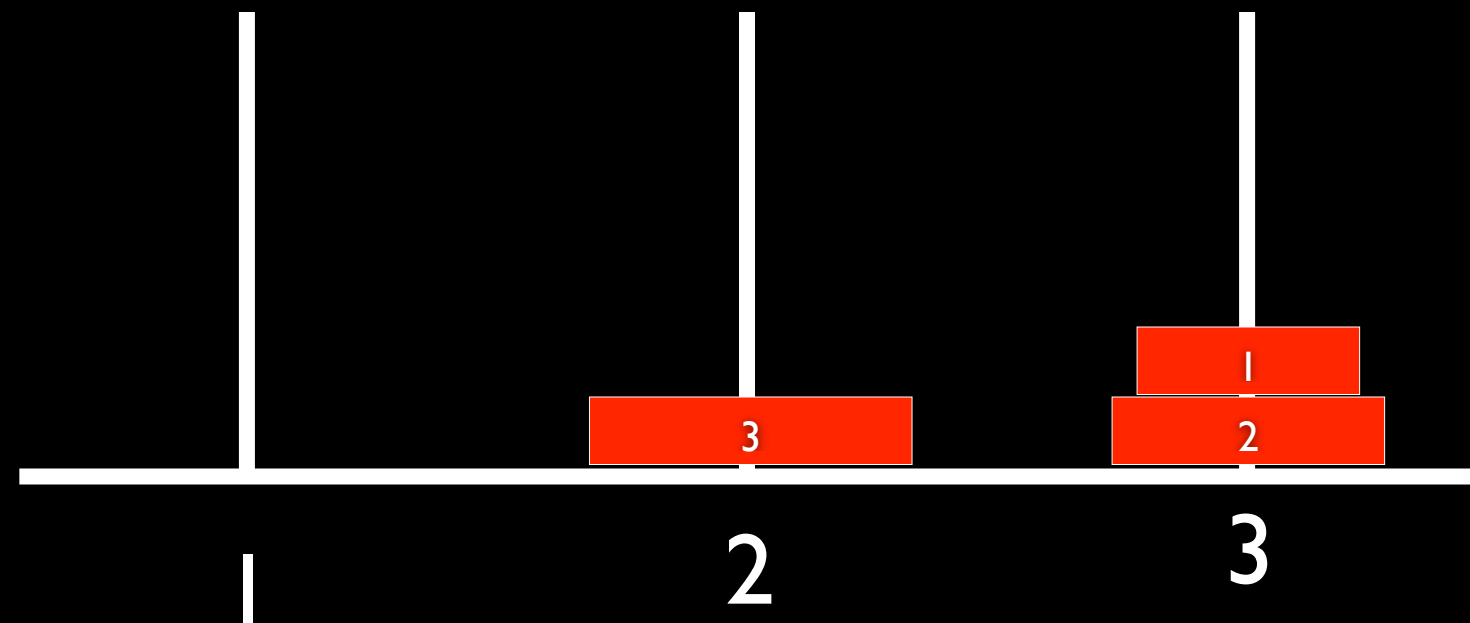
hanoi(3,1,2) # move 3 disks from post 1 to 2



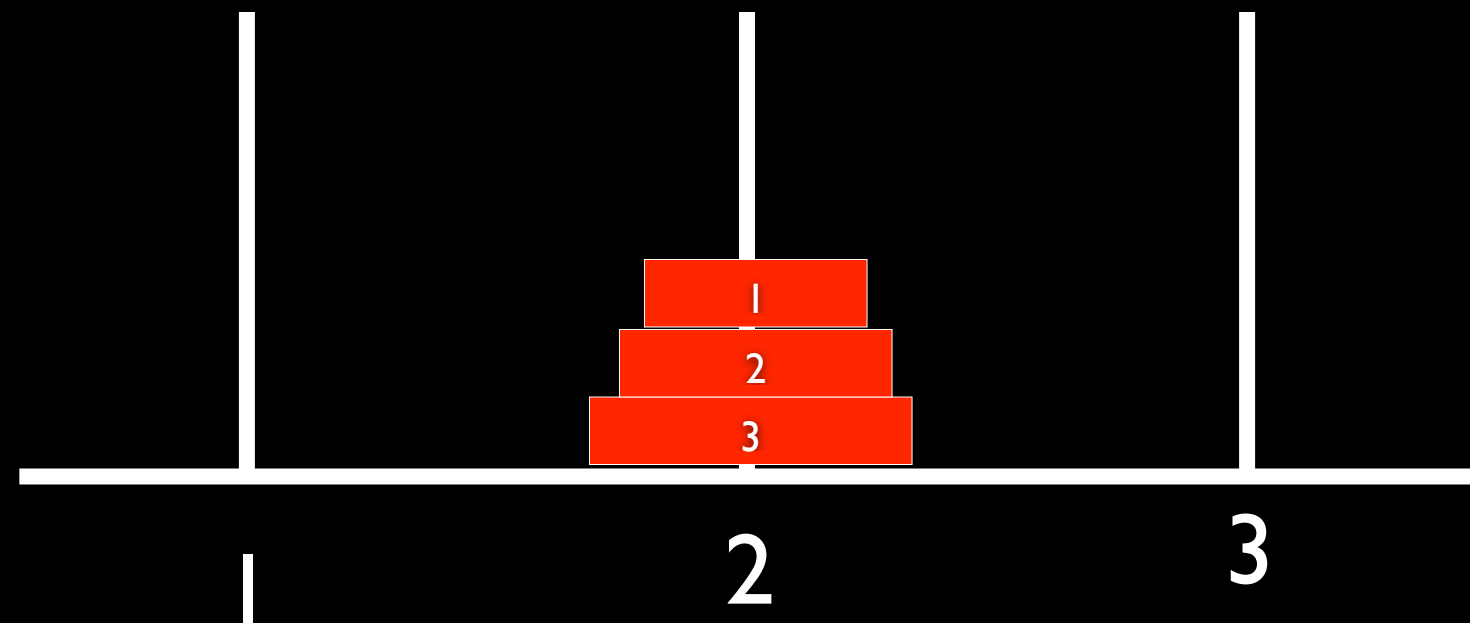
hanoi(3,1,2) # move 3 disks from post 1 to 2
hanoi(2,1,3) # move 2 disks from post 1 to 3



```
hanoi(3,1,2) # move 3 disks from post 1 to 2  
  hanoi(2,1,3) # move 2 disks from post 1 to 3  
    move(3,1,2) # move disk 3 from post 1 to 2
```



```
hanoi(3,1,2) # move 3 disks from post 1 to 2  
  hanoi(2,1,3) # move 2 disks from post 1 to 3  
    move(3,1,2) # move disk 3 from post 1 to 2  
      hanoi(2,3,2) # move 2 disks from post 3 to 2
```

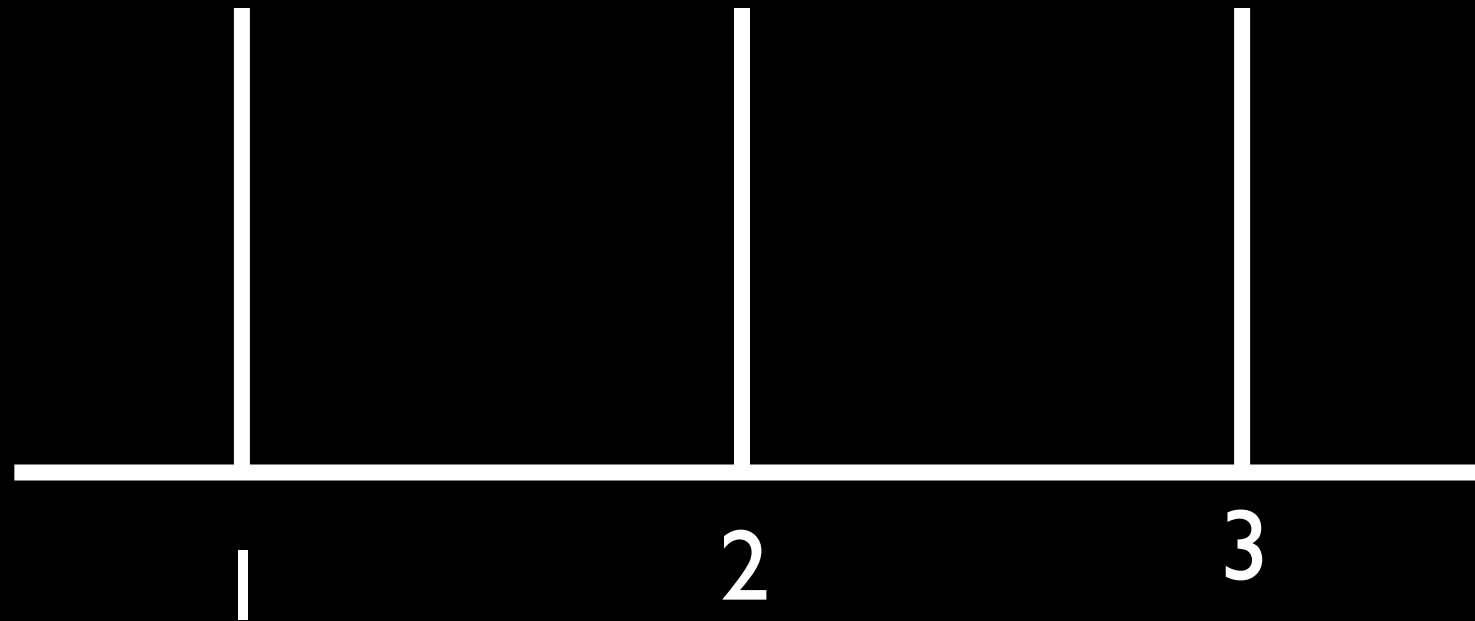


```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:
```



```
def move_disk(disk_number, from_peg, to_peg):  
    print("Move disk " + str(disk_number) + " from peg " \  
          + str(from_peg) + " to peg " + str(to_peg) + ".")  
  
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:
```

`spare_peg = 6 - start_peg - end_peg`



```
def move_disk(disk_number, from_peg, to_peg):  
    print("Move disk " + str(disk_number) + " from peg " \  
          + str(from_peg) + " to peg " + str(to_peg) + ".")  
  
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:
```

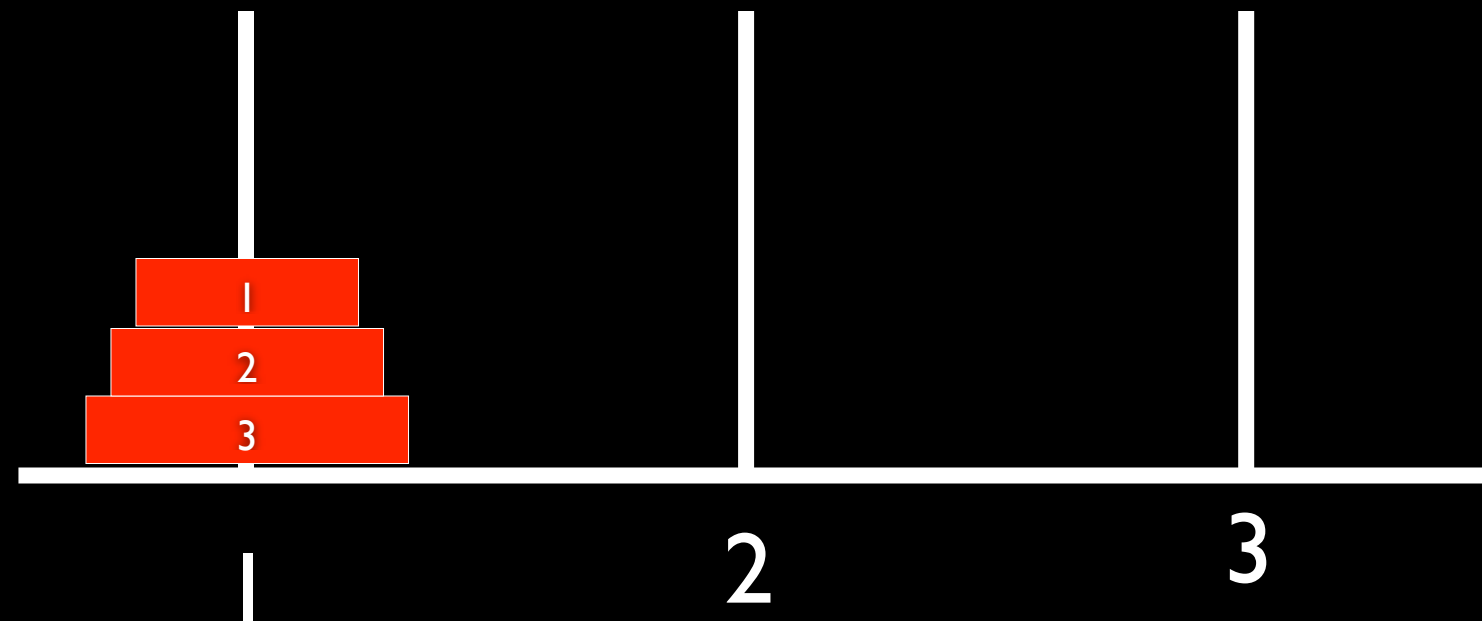
```
def move_disk(disk_number, from_peg, to_peg):  
    print("Move disk " + str(disk_number) + " from peg " \  
          + str(from_peg) + " to peg " + str(to_peg) + ".")  
  
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)
```

```
def move_disk(disk_number, from_peg, to_peg):  
    print("Move disk " + str(disk_number) + " from peg " \  
          + str(from_peg) + " to peg " + str(to_peg) + ".")  
  
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)
```

```
def move_disk(disk_number, from_peg, to_peg):  
    print("Move disk " + str(disk_number) + " from peg " \  
          + str(from_peg) + " to peg " + str(to_peg) + ".")  
  
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

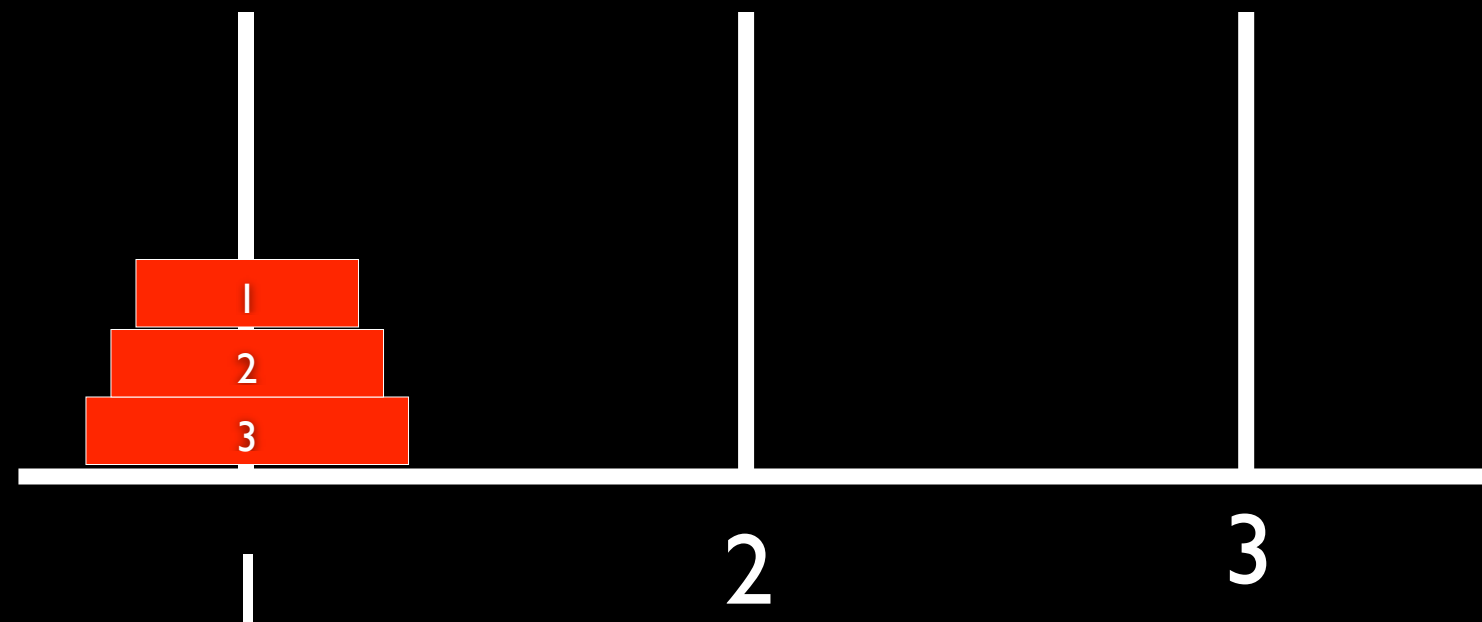
```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

hanoi(3,1,2)



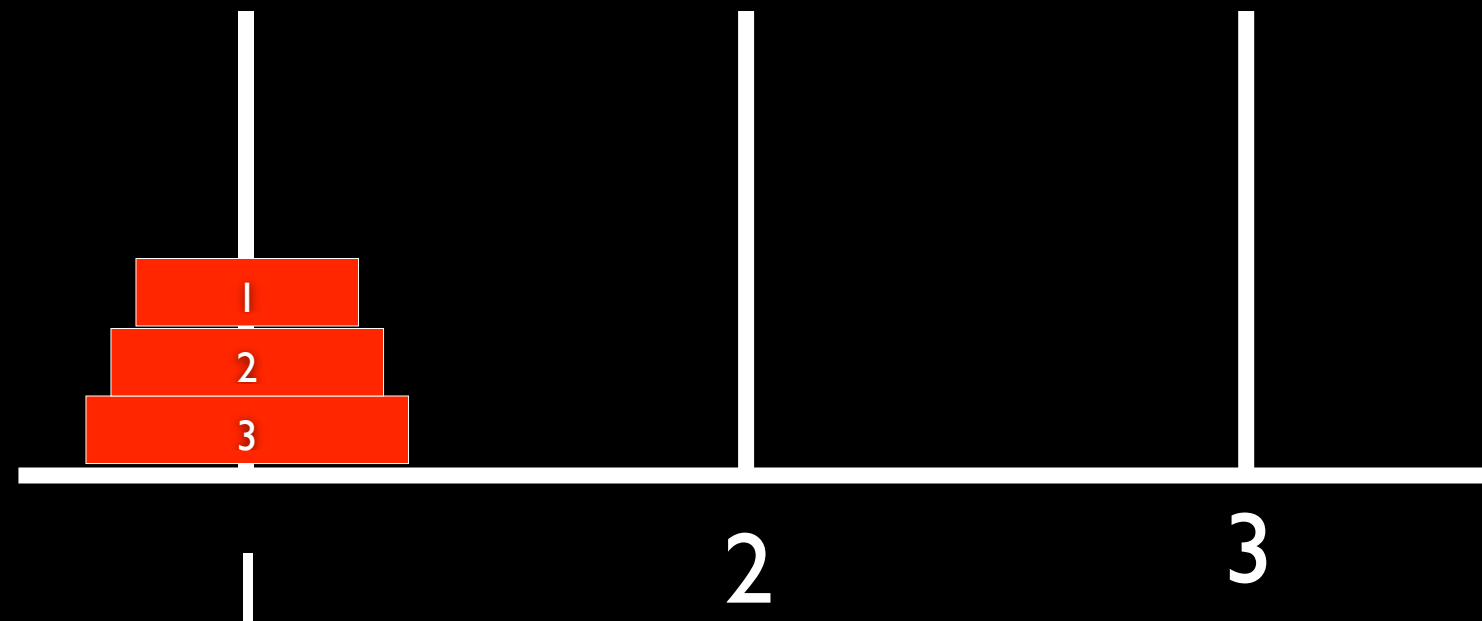
```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

```
hanoi(3,1,2)  
    hanoi(2,1,3)  
        move_disk(3,1,2)  
            hanoi(2,3,2)
```



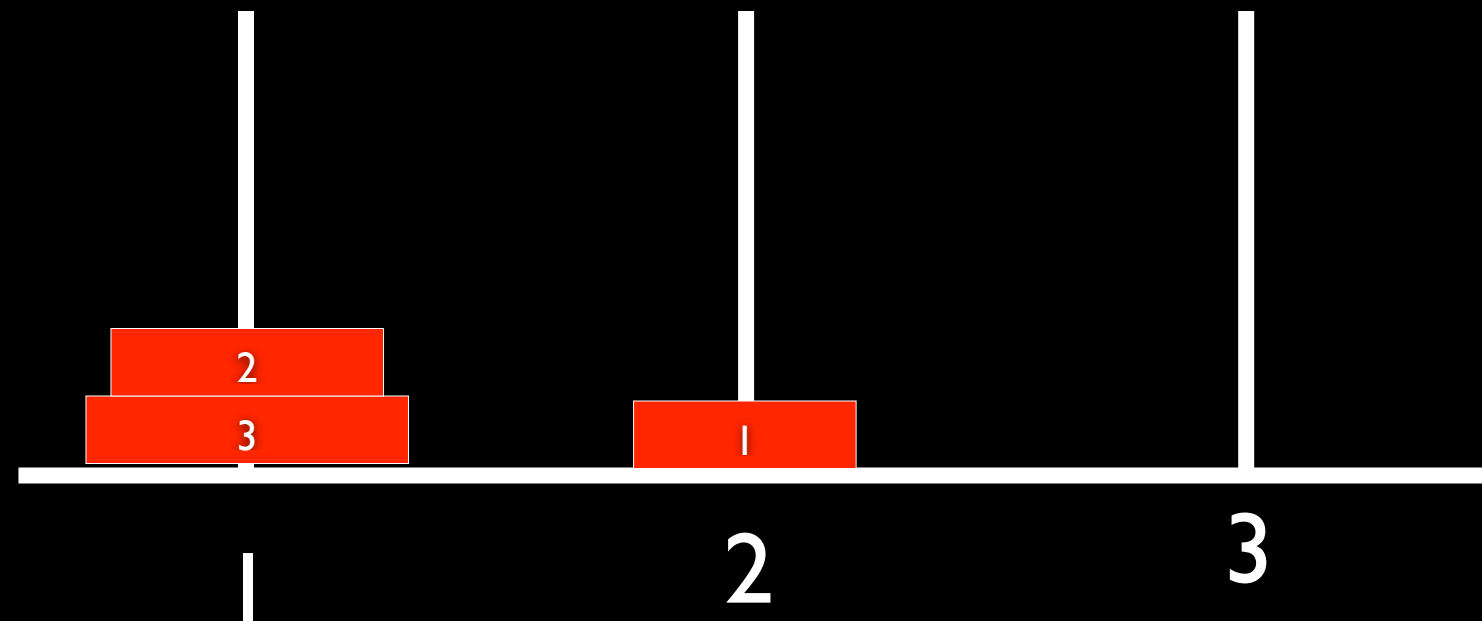

```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

```
hanoi(3,1,2)  
  hanoi(2,1,3)  
    hanoi(1,1,2)  
    move_disk(2,1,3)  
    hanoi(1,2,3)  
  move_disk(3,1,2)  
  hanoi(2,3,2)
```



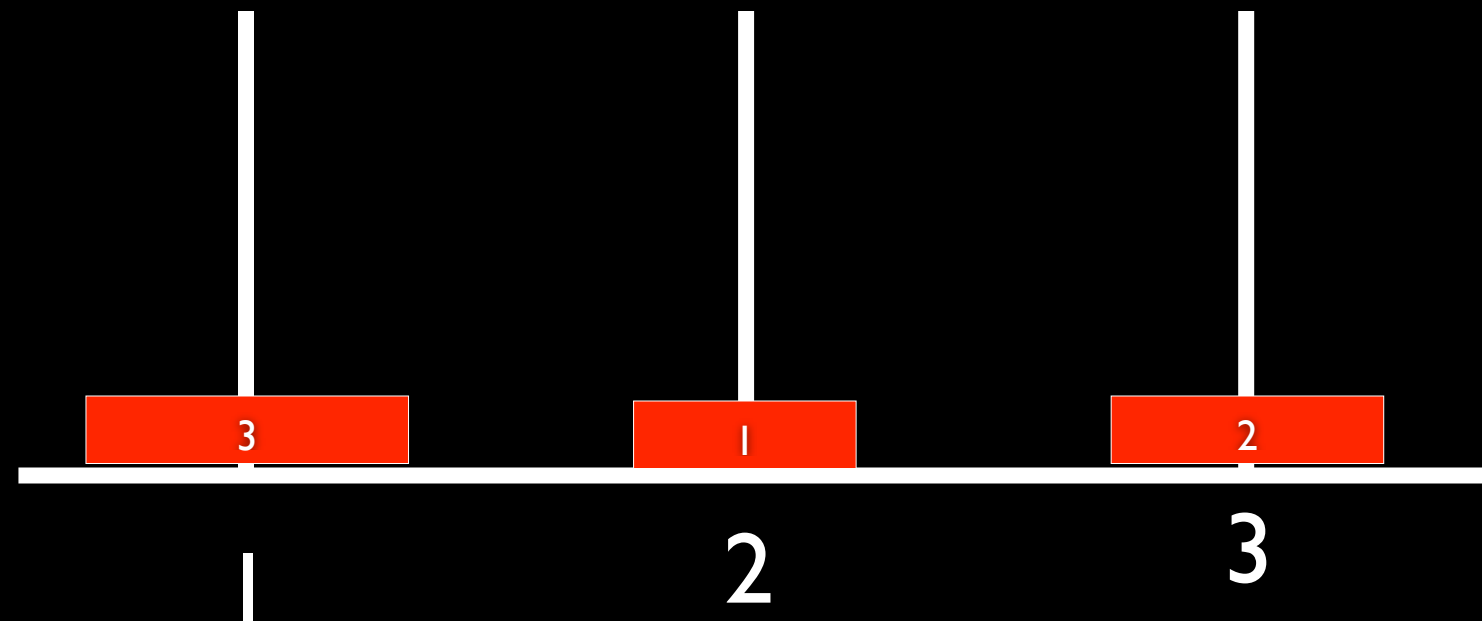
```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

```
hanoi(3,1,2)  
    hanoi(2,1,3)  
        hanoi(1,1,2)  
        move_disk(2,1,3)  
        hanoi(1,2,3)  
    move_disk(3,1,2)  
    hanoi(2,3,2)
```



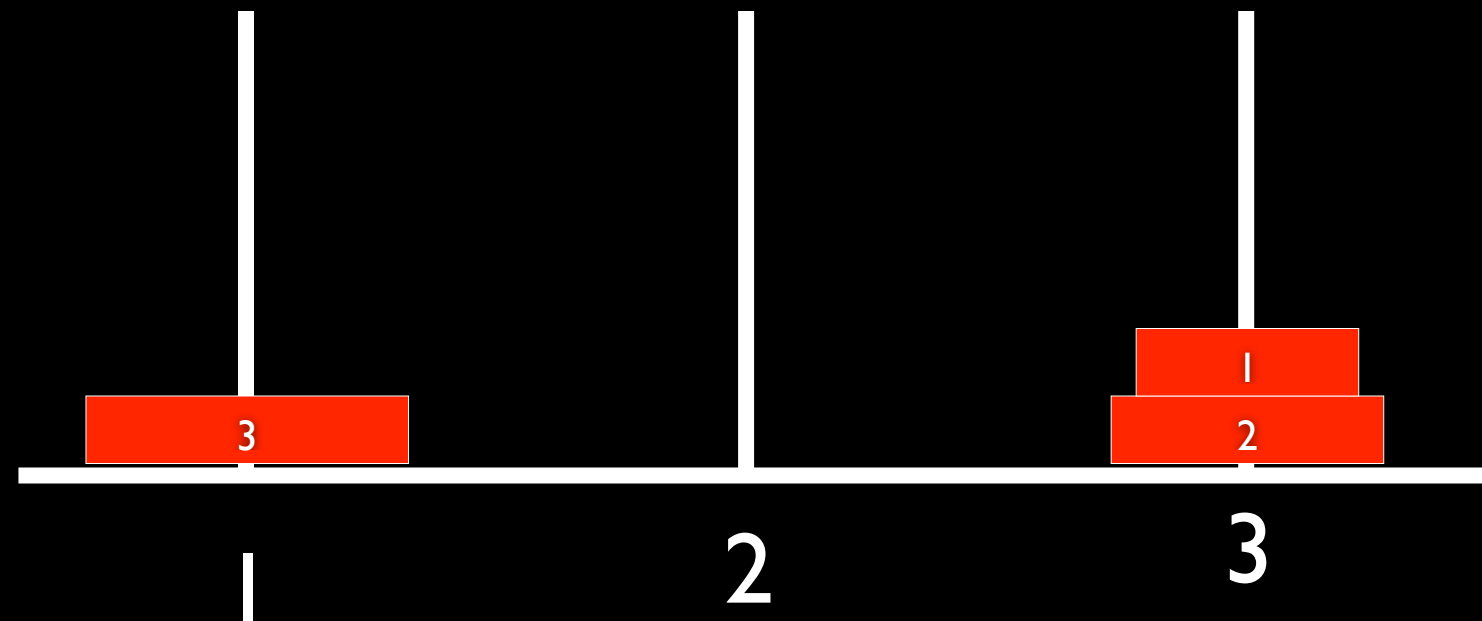
```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

```
hanoi(3,1,2)  
    hanoi(2,1,3)  
        hanoi(1,1,2)  
        move_disk(2,1,3)  
        hanoi(1,2,3)  
    move_disk(3,1,2)  
    hanoi(2,3,2)
```



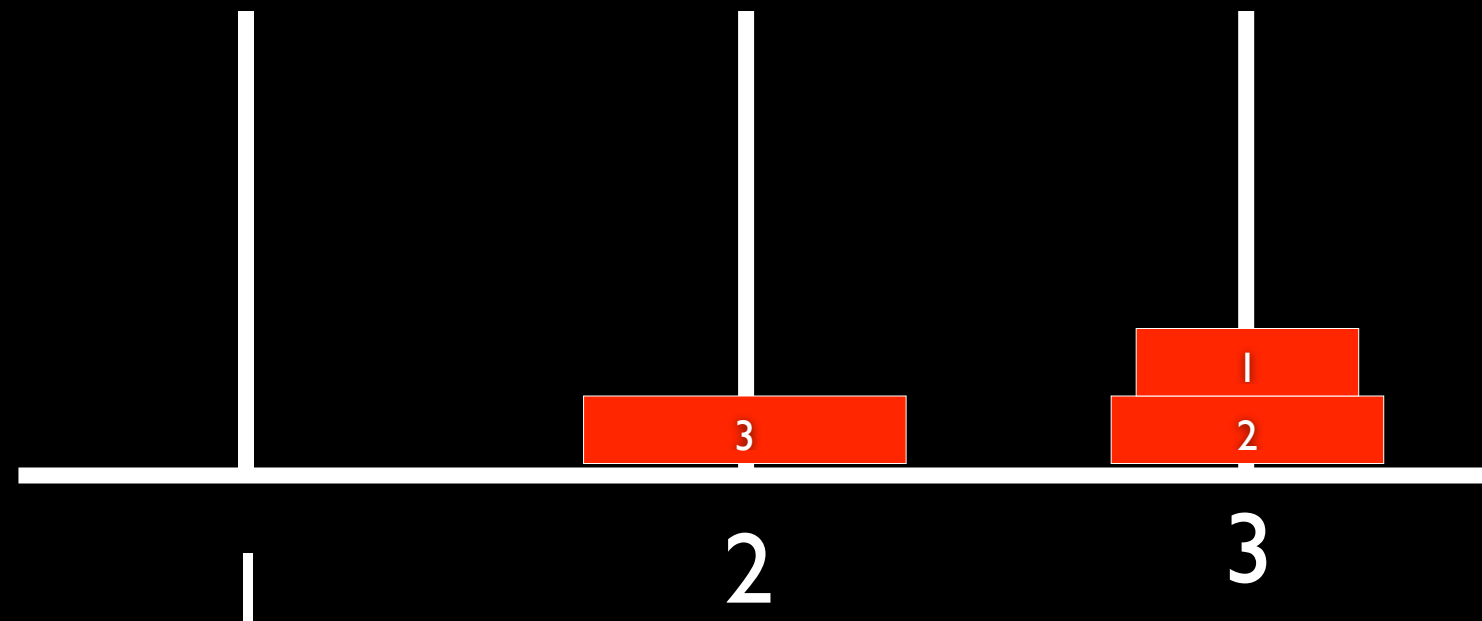
```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

```
hanoi(3,1,2)  
    hanoi(2,1,3)  
        hanoi(1,1,2)  
        move_disk(2,1,3)  
        hanoi(1,2,3)  
    move_disk(3,1,2)  
    hanoi(2,3,2)
```



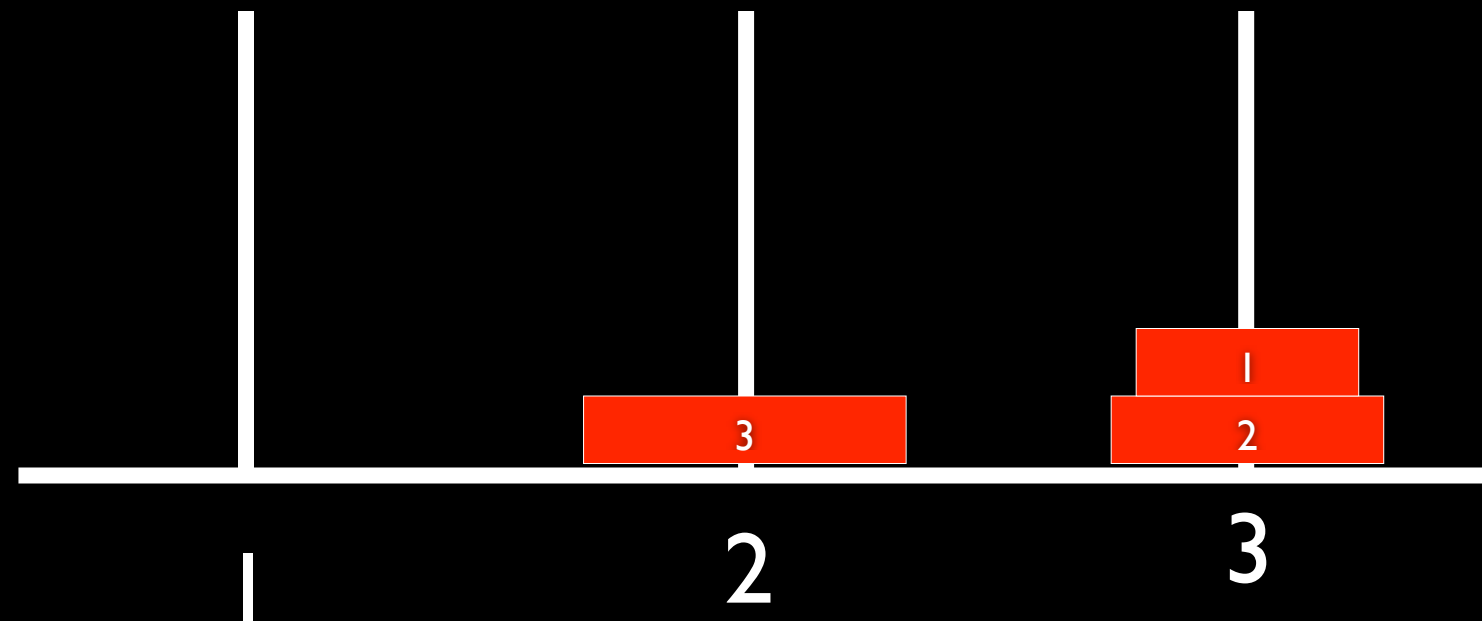
```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

```
hanoi(3,1,2)  
    hanoi(2,1,3)  
        hanoi(1,1,2)  
        move_disk(2,1,3)  
        hanoi(1,2,3)  
    move_disk(3,1,2)  
    hanoi(2,3,2)
```



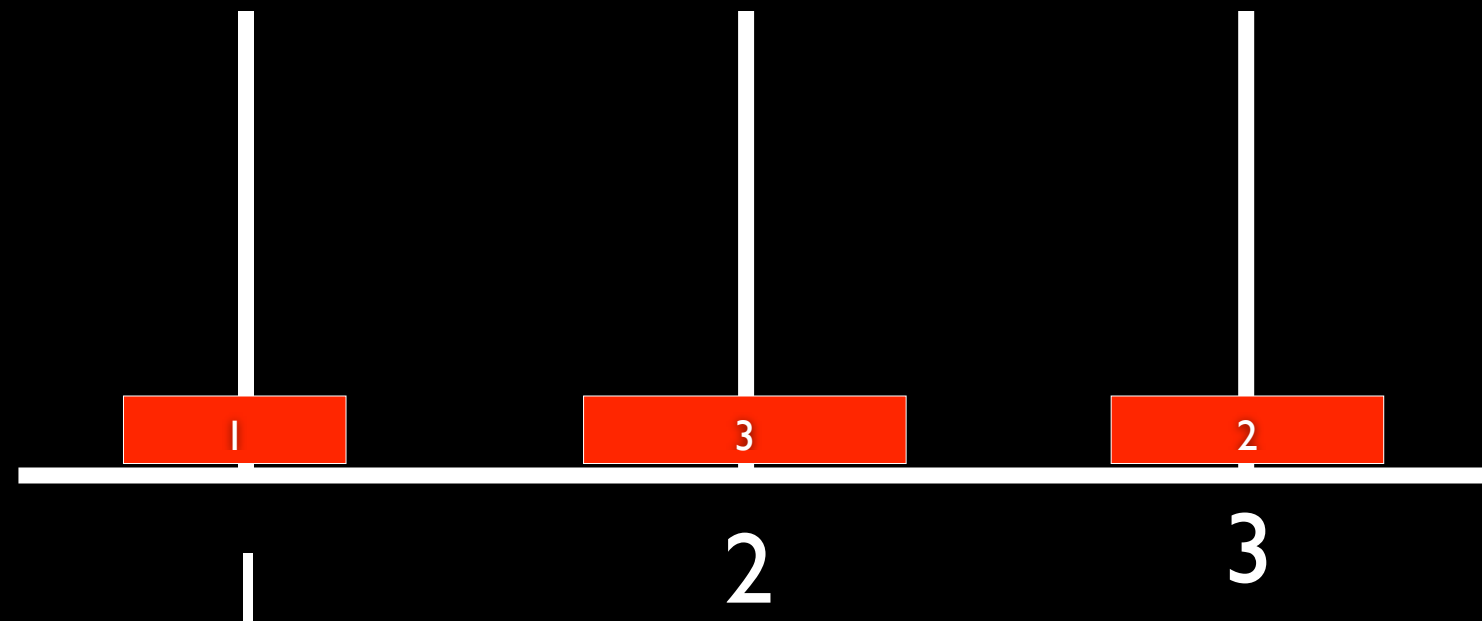
```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

```
hanoi(3,1,2)  
    hanoi(2,1,3)  
        hanoi(1,1,2)  
        move_disk(2,1,3)  
        hanoi(1,2,3)  
    move_disk(3,1,2)  
    hanoi(2,3,2)  
        hanoi(1,3,1)  
        move_disk(2,3,2)  
        hanoi(1,1,2)
```



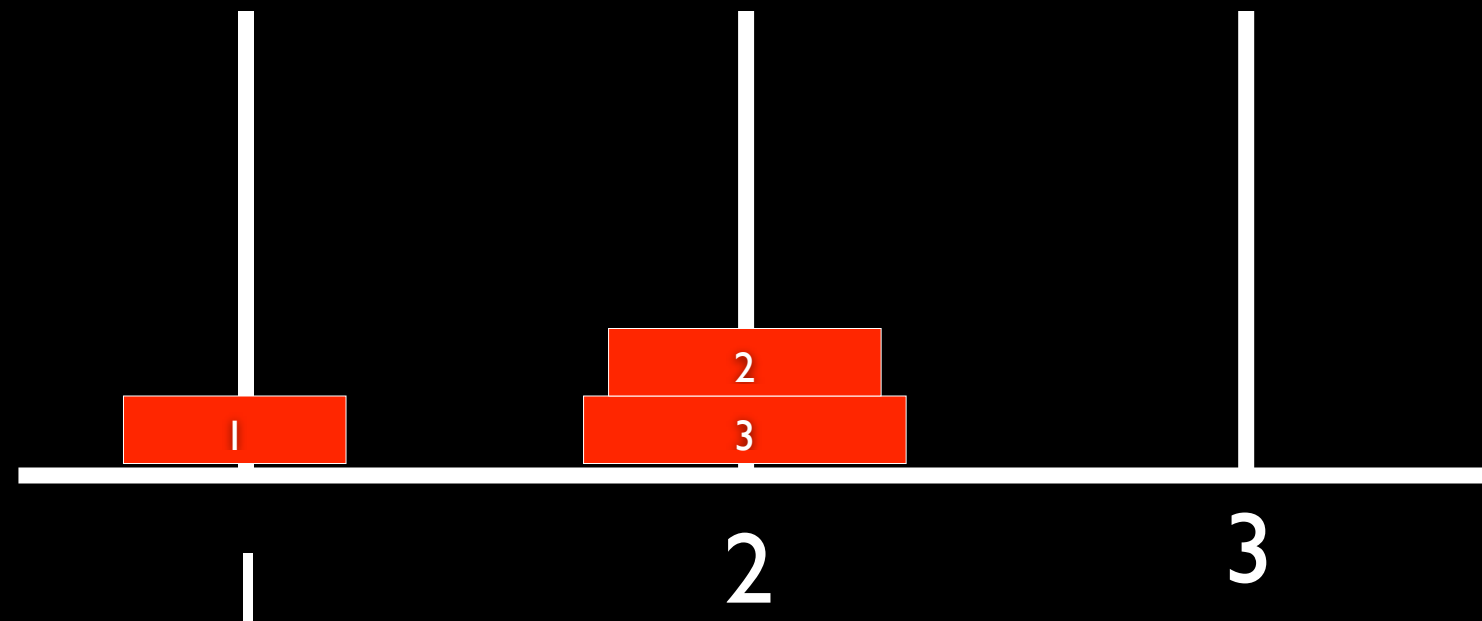
```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

```
hanoi(3,1,2)  
    hanoi(2,1,3)  
        hanoi(1,1,2)  
        move_disk(2,1,3)  
        hanoi(1,2,3)  
    move_disk(3,1,2)  
    hanoi(2,3,2)  
        hanoi(1,3,1)  
        move_disk(2,3,2)  
        hanoi(1,1,2)
```



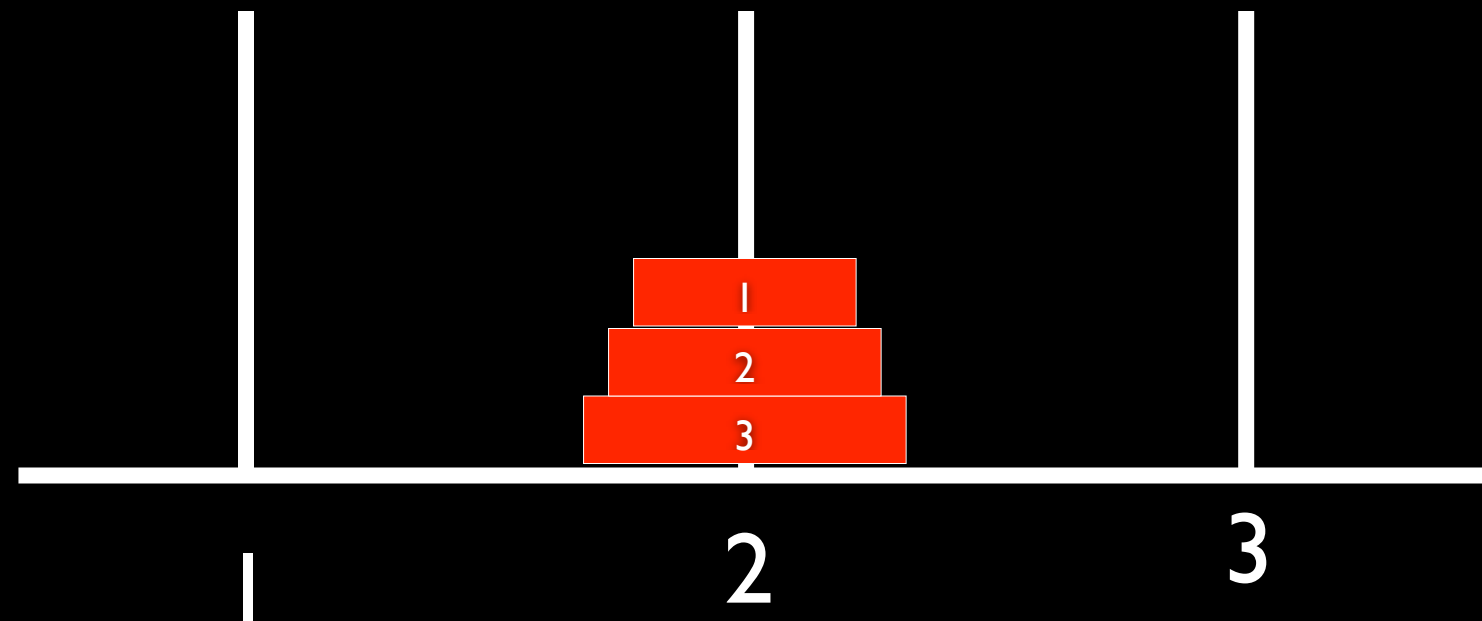
```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

```
hanoi(3,1,2)  
    hanoi(2,1,3)  
        hanoi(1,1,2)  
        move_disk(2,1,3)  
        hanoi(1,2,3)  
    move_disk(3,1,2)  
    hanoi(2,3,2)  
        hanoi(1,3,1)  
        move_disk(2,3,2)  
        hanoi(1,1,2)
```



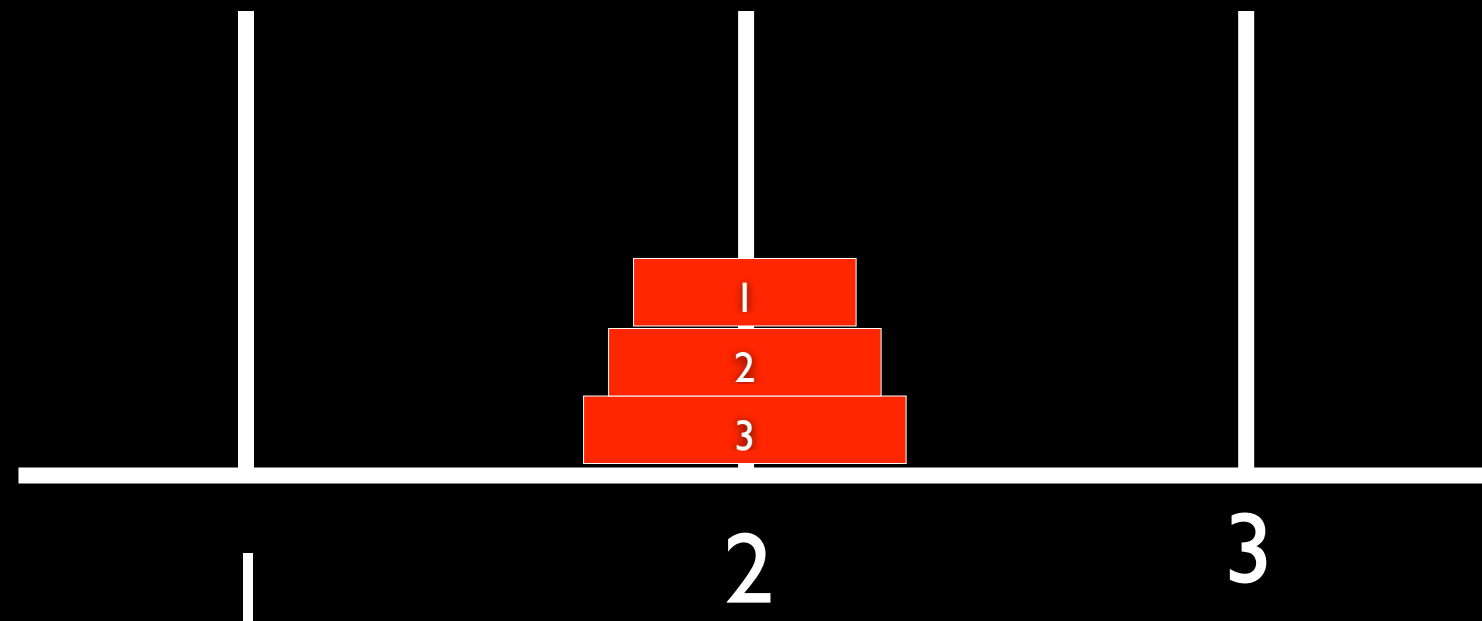

```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

```
hanoi(3,1,2)  
    hanoi(2,1,3)  
        hanoi(1,1,2)  
        move_disk(2,1,3)  
        hanoi(1,2,3)  
    move_disk(3,1,2)  
    hanoi(2,3,2)  
        hanoi(1,3,1)  
        move_disk(2,3,2)  
        hanoi(1,1,2)
```



```
def solve_hanoi(n, start_peg, end_peg):  
    if n == 1:  
        move_disk(n, start_peg, end_peg)  
    else:  
        spare_peg = 6 - start_peg - end_peg  
        solve_hanoi(n - 1, start_peg, spare_peg)  
        move_disk(n, start_peg, end_peg)  
        solve_hanoi(n - 1, spare_peg, end_peg)
```

```
hanoi(3,1,2)  
    hanoi(2,1,3)  
        hanoi(1,1,2)  
        move_disk(2,1,3)  
        hanoi(1,2,3)  
    move_disk(3,1,2)  
    hanoi(2,3,2)  
        hanoi(1,3,1)  
        move_disk(2,3,2)  
        hanoi(1,1,2)
```



discs moves

1 1

2 3

3

4

5

6

7

8

9

10

11

12

...

64

discs	moves
1	1
2	3
3	7
4	
5	
6	
7	
8	
9	
10	
11	
12	
...	
64	

discs moves

1	1
2	3
3	7
4	15
5	
6	
7	
8	
9	
10	
11	
12	
...	
64	

discs moves

1	1
2	3
3	7
4	15
5	31
6	63
7	127
8	255
9	511
10	1,023
11	2,047
12	4,095
...	
64	18,446,744,073,709,551,615