https://github.com/prateek27/java-jan-22

Recursion Basics

Recursion

Recursion

Technique breakdown into smaller problems of the same type

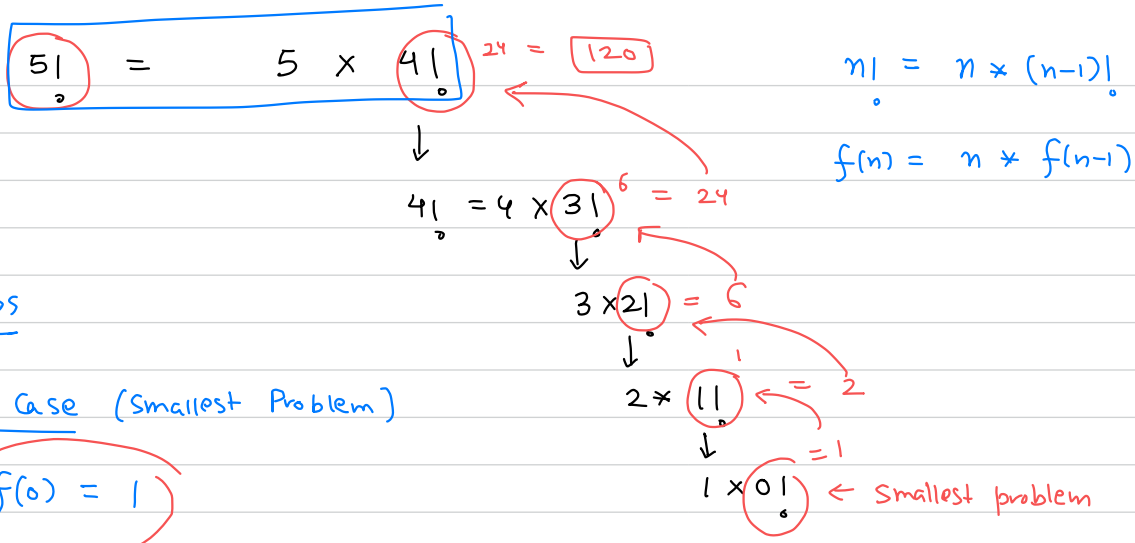
$5!_{0} = 1 \times 2 \times 3 \times 4 \times 5$

Loop

Larger: $5!_{0}$ = 5 * Smaller: $4!_{0}$

$\Rightarrow f(n) = n * f(n-1)$

Factorial



$n! = n * (n-1)!$

$f(n) = n * f(n-1)$

Two Things

1. Base Case (Smallest Problem)

⇒ $f(0) = 1$

2. Recursive Fn/Case

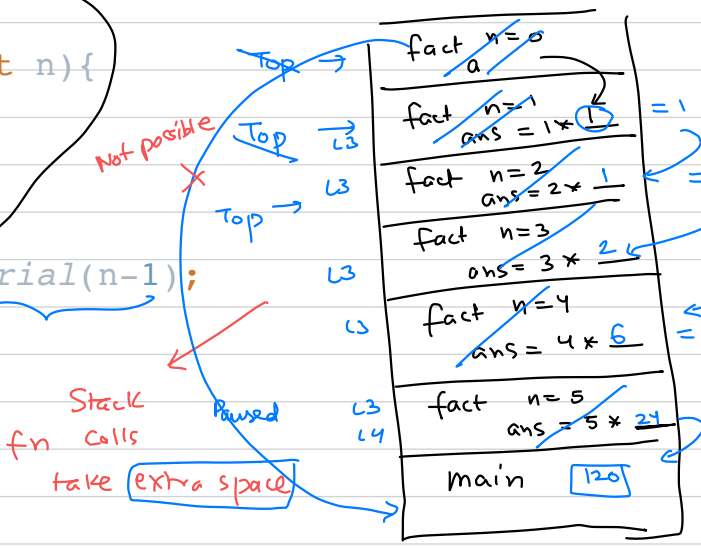
⇒ $f(n) = n * f(n-1)$

1 or more Function Call to same function

```
static int factorial(int n){
    //Base Case
L1  if(n==0){
L2      return 1;
    }
    //Rec Case
L3  int ans = n * factorial(n-1);
L4  return ans;
}

main() {

    fact(5) → 120

}
```

Local

New Fn call → new variables are created



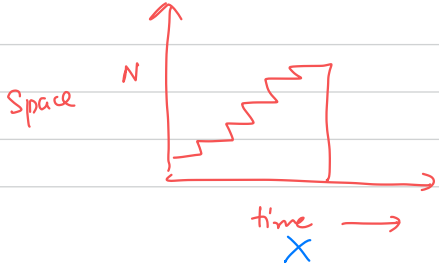
n copies of n, ans

2n buckets

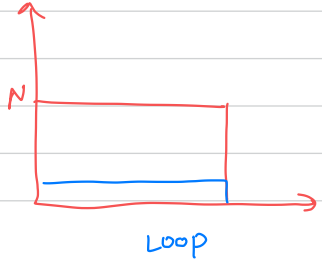
$\propto n$

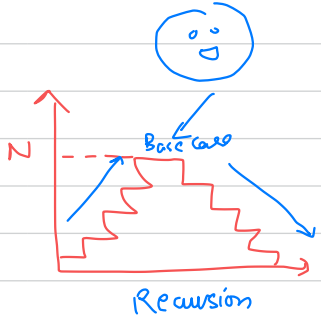
$= O(N)$

Space



Space





```
ans = 1
for( i=1 ; i<=n ; i++ )
  { ans = ans * i }
```

2 buckets

i = ans =

→ Constant

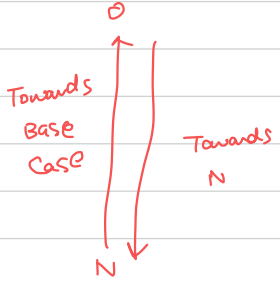
O(1) ← Notation

Problems

- Recursion
- Loop (fact)

↓

[Writing loops can become complex.]



Recursion

Rec. Code

```
void goToHome (int x) {
    // Base case
    if (x==0)
        return;

    • Take A Step left
    • goToHome (x-1);
}

goToHome (N);
```

Loop

```
x = N
while (x > 0){
    Take a step left
    x = x - 1;
}
```

Principle of Mathematical Induction (PMI)

(1) Figure out the smallest case

$f(0) = ?$ $f(1) = ?$

(2) Assume that prob can be solve for given k

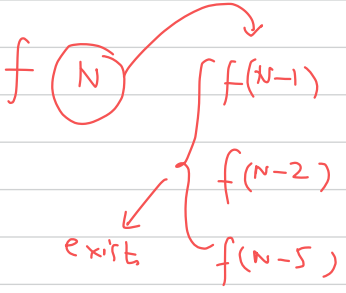
Hypothesis

Solutions to sub problems exist.

$f(k)$ exist s.t $k < N$

(3) Using $f(k)$ write / find out

$f(N)$



Fact

→ $f(0) = 1$

→ $f(N) = N * f(N-1)$

↑ Assume.

(2) Multiply (a) and (n) without using '*' // Recursion

$a = 5$

$n = 6$ 0?

Smaller → n-1

$a + [a + a + \cdots a]$

n times

$f(a, n)$

Base case $f(a, 0) = 0$ ✓

$f(a, 1) = a$ ✓

Rec case $f(a, n) = a + f(a, n-1)$

Add a n times

$1 + 1 + 1 + 1 + 1 = 5$

$[1 + 1 + 1 + 1 + 1] + 1 = 6$

$a * 0 = 0$

$5 * 6 = 5 + \underbrace{5 + 5 + 5 + 5 + 5}_{n \text{ times}}$

$\underbrace{\quad\quad}_{n-1 \text{ times}}$

$f(a,n) = 5 + 5 * 5$

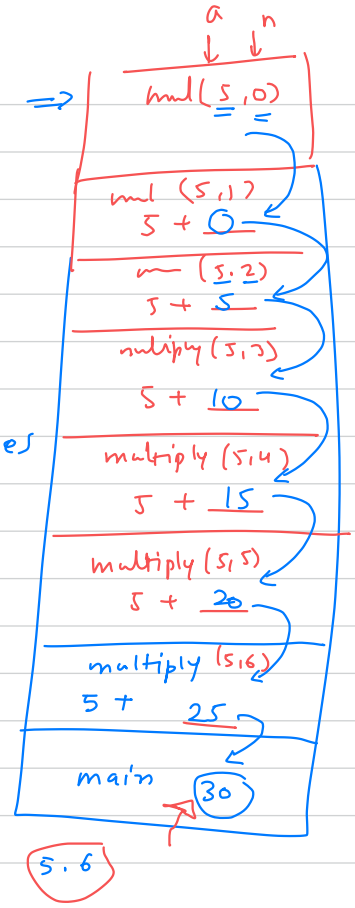
$= a + f(a, n-1)$

multiply a n-1 times

$a * n = a + a * (n-1)$

$f(a,n) = a + f(a, n-1)$

$a + mul(a, n-1)$

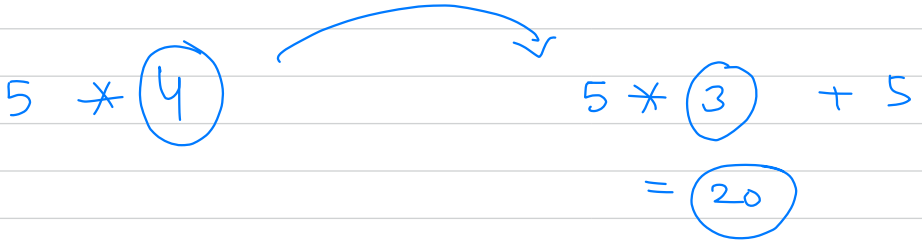



```
static int multiply(int a,int n){
    //base case
    if(n==0){
        return 0;
    }
    return a + multiply(a,n-1);
}
```

(1)

```
int ans=0
for(i=1; i<=n; i++)
    [ ans = ans+a ]
```

(+) (x)



Simple

Increasing order from 1 to N
Dec order from N to 1.

N = 5 → 1, 2, 3, 4, 5
N = 5 → 5, 4, 3, 2, 1

Dec

n ↓ dec(n-1)
5, 4, 3, 2, 1

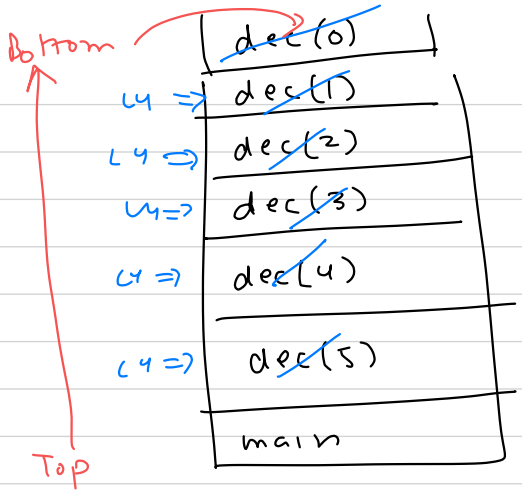
Dec[n] → if (n == 0) return (Base)
print (n)
Dec (n-1) (Rec. case)

dec(n) = n, dec(n-1)
dec(n-1) = n-1, n-2, ... 3, 2, 1

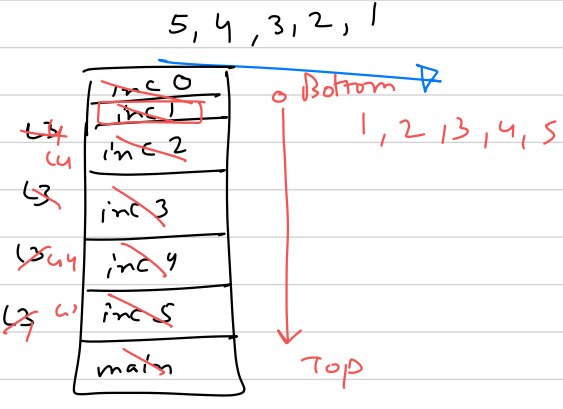
Inc(n) if n == 0 return (Base Case)
inc(n-1)
print (n) (Rec case)

inc(n-1) n ↓
1, 2, 3, 4, 5, ... 8, 9, 10

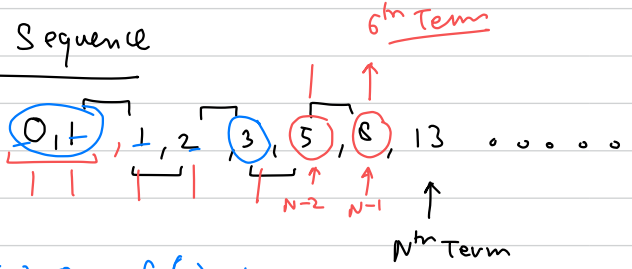
```java
static void dec(int n){
    //base case
    if(n==0){
        return;
    }
    //rec case
    System.out.print(n + ",");
    dec(n-1);
}
```




```java
static void inc(int n){
    //base case
    if(n==0){
        return;
    }
    //rec case
    inc(n-1);
    System.out.print(n + ",");
}
```

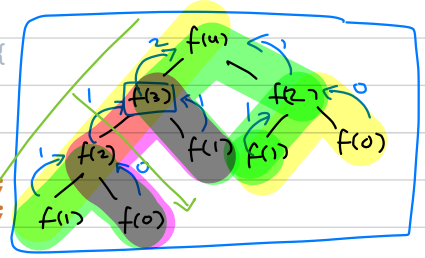


Fibonacci Sequence

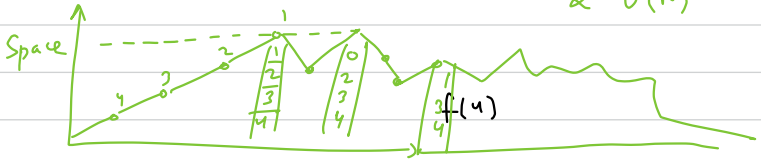


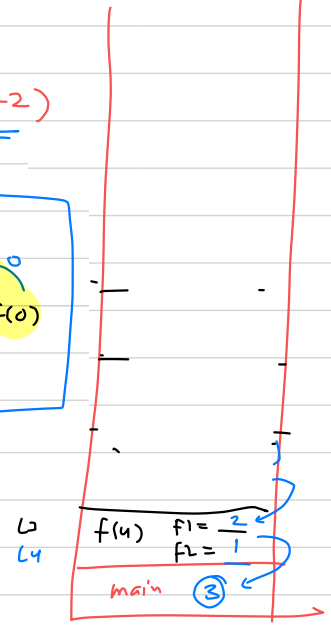
- $f(0)=0$, $f(1)=1$
- $f(N) = f(N-1) + f(N-2)$

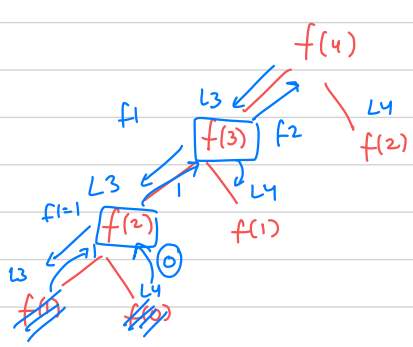
```
static int fibonacci(int n){
  L1  if(n==0 || n==1){
  L2      return n;
      }
  L3  int f1 = fibonacci(n-1);
  L4  int f2 = fibonacci(n-2);
  L5  int ans = f1 + f2;
  L6  return ans;
  }
```

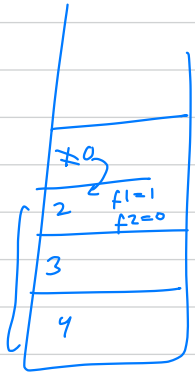


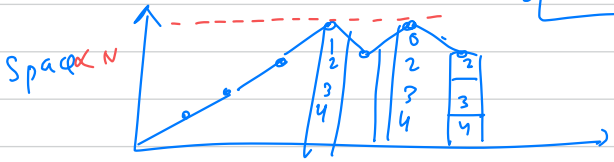
$\propto O(N)$

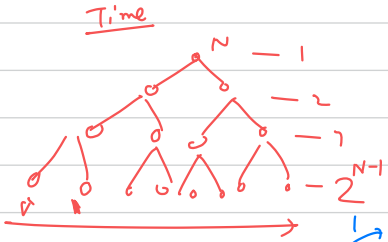


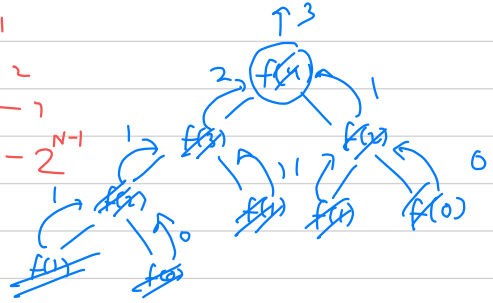










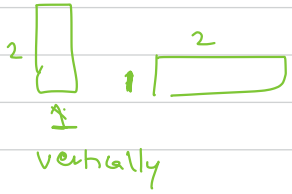


→ Sequential

⇒ NOT Parallel ✗

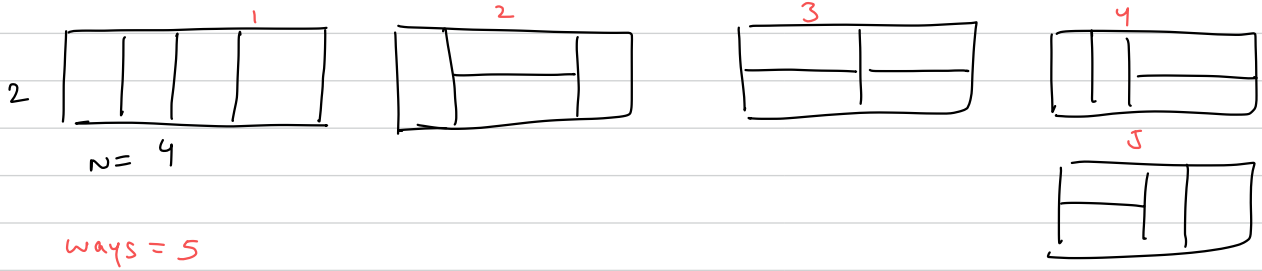
$1 + 2 + 4 + 8 + \ldots\ldots + 2^{(n-1)} \Rightarrow O(2^n)$

Tiles



(a) How many ways to build the floor N^{x2} units using 1x2, 2x1 tiles

N=4



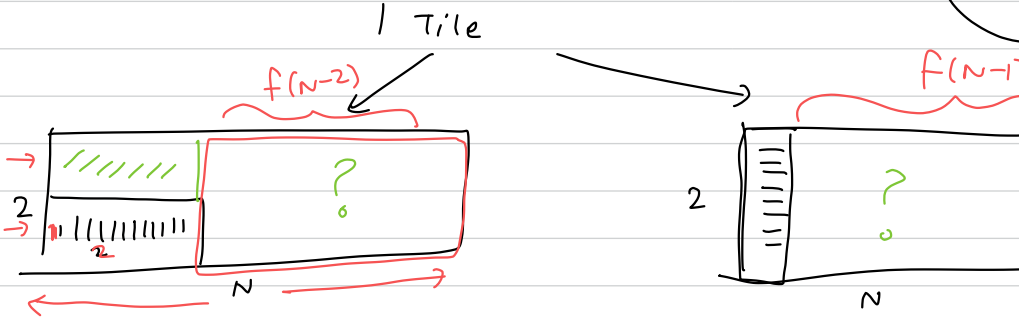
ways = 5

Build the floor

$f(N) = ?$

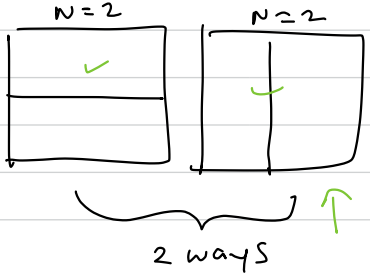
Do 1 step & leave Rest

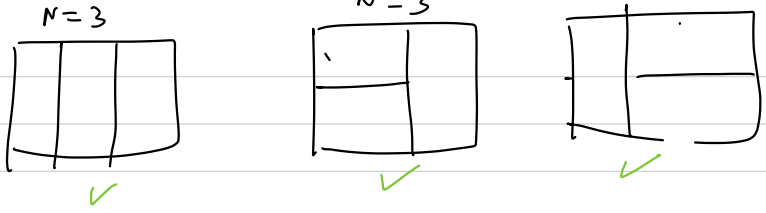
hint:



$f(n) = f(n-1) + f(n-2)$

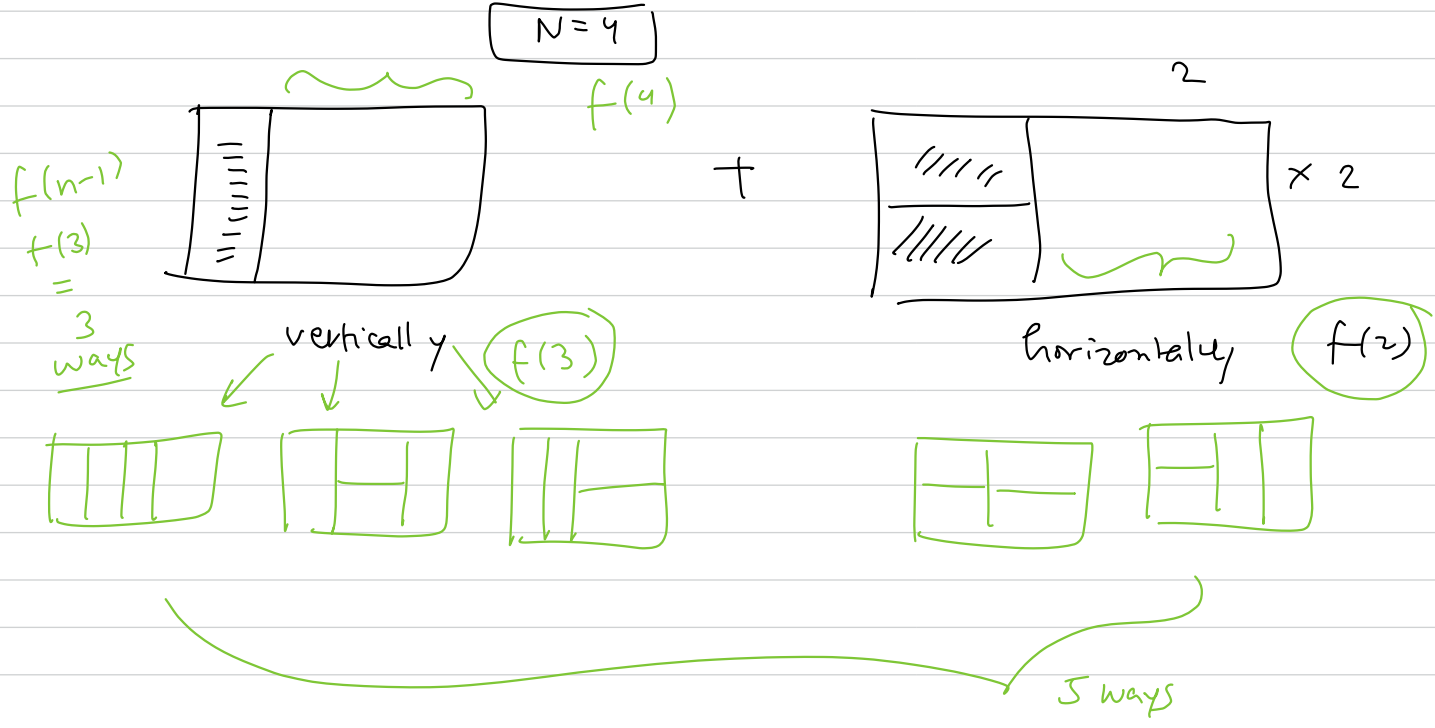
N = 0 → 1 way
N = 1 → 1 way
N = 2 → 2 ways
N = 3 → 3 ways
N = 4 → 5 ways





Think Rec → Sol

Impl Loops → Sol



1, 1, 2, 3, 5 (c) = a + b

a b

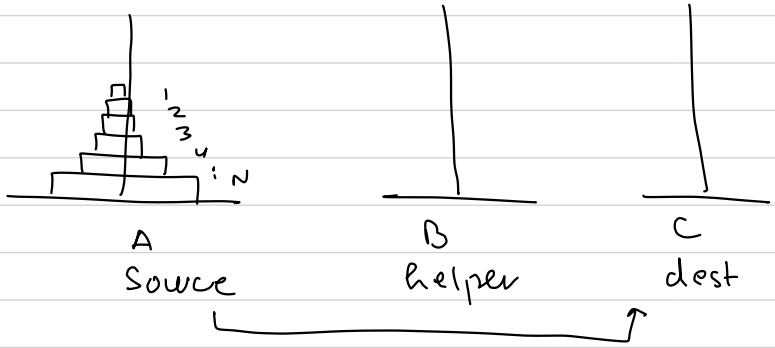
for(———){

=

}

TOWER OF HANOI

→ Game setup



Condition.

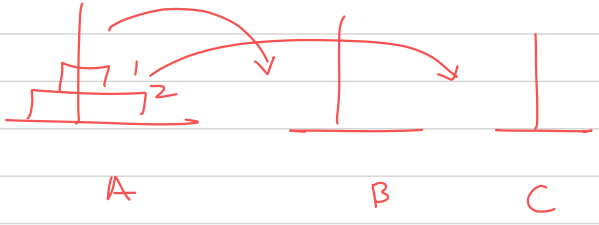
1 You can move 1 disk at time
2 You can't place a bigger disk over a small disk.

Print the steps to transfer N Disks

N=1

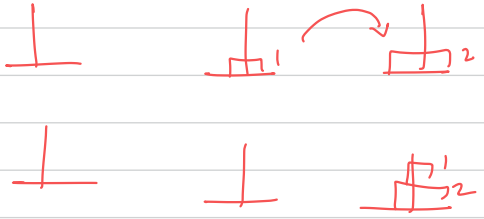
1 step

N=2



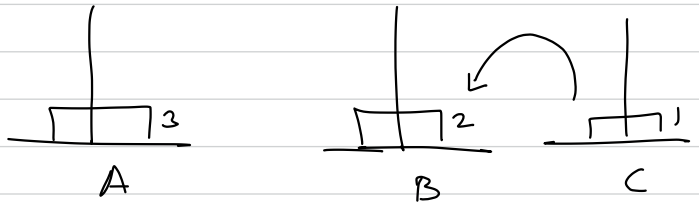
Move 1 A to B
Move 2 A to C
Move 1 B to C

3 steps

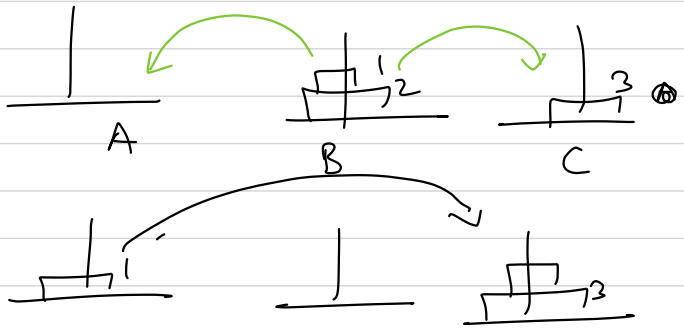


N=3

7 steps



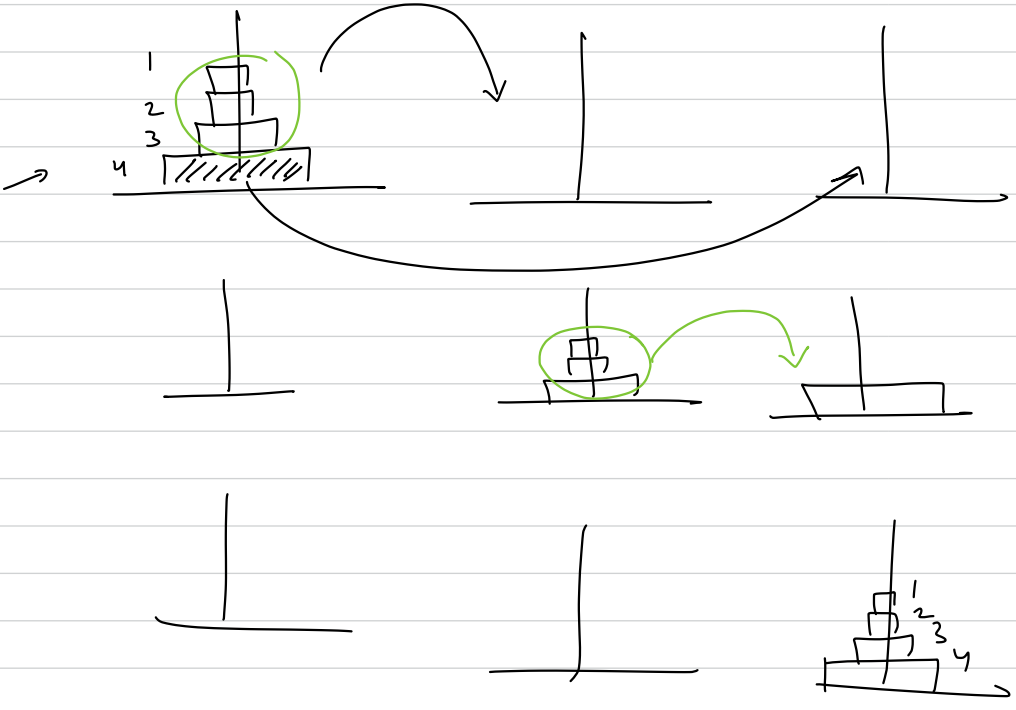
Move 1 to A to C
move 2 A to B
Move 1 C to B



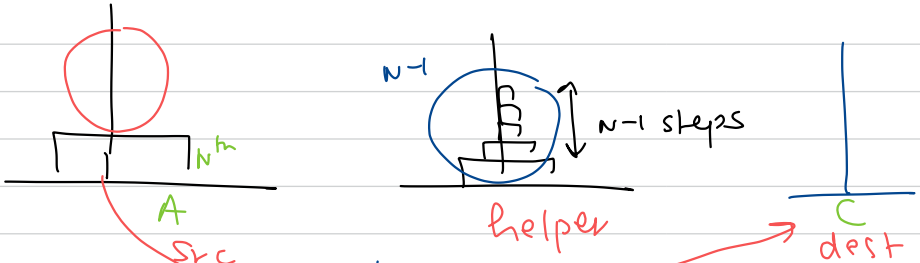
Move 3 A to C

Move 1 B to A
Move 2 B to C
Move 1 A to C

N=4

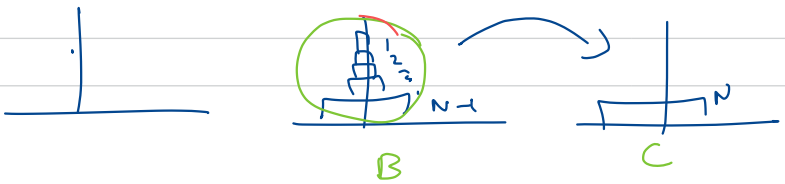


(1) move N-1 disks from A to B TOH(N-1, A, B)



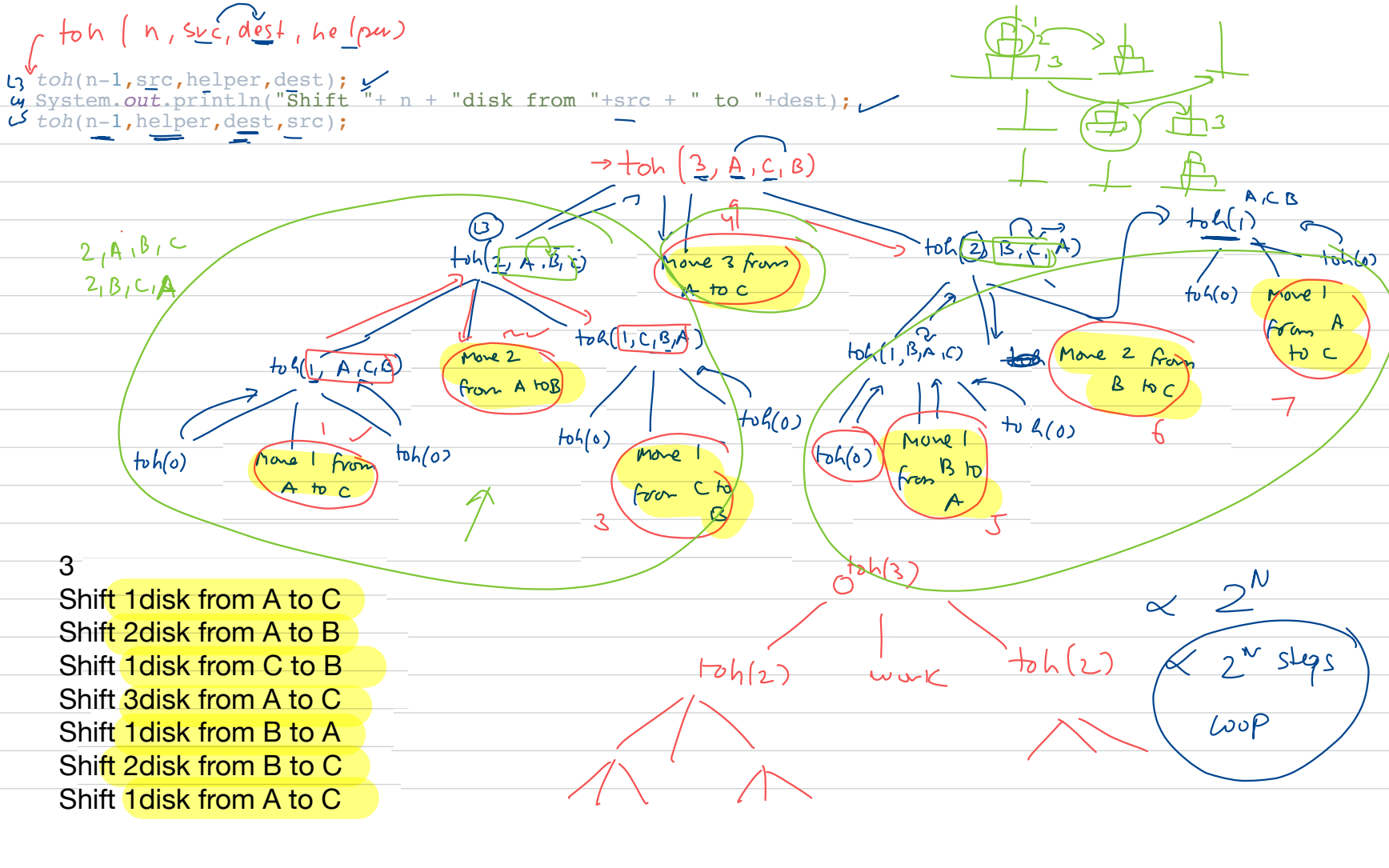
Work for 1 disk

(2) Work. Shift Nth disk from A to C

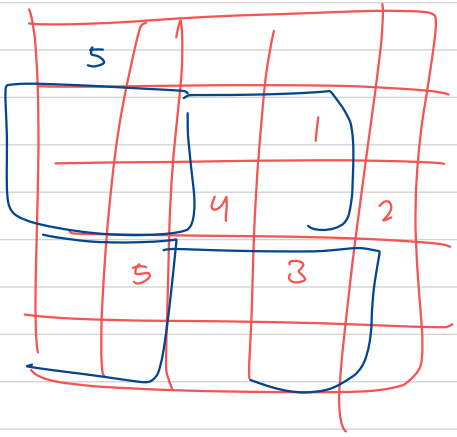


3 Move N-1 disks B to C ToH(N-1, B, C)

2
Shift 1disk from A to B
Shift 2disk from A to C
Shift 1disk from B to C



3
Shift 1disk from A to C
Shift 2disk from A to B
Shift 1disk from C to B
Shift 3disk from A to C
Shift 1disk from B to A
Shift 2disk from B to C
Shift 1disk from A to C



Sudoku