
Recursion

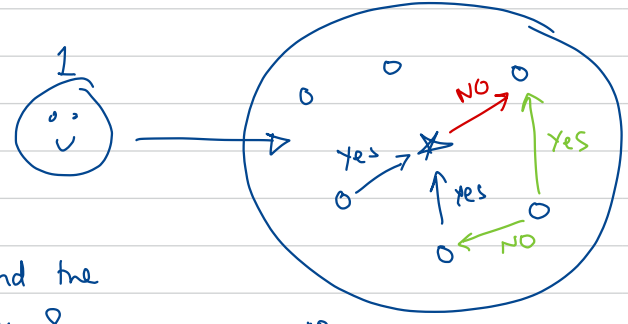
Find the Celebrity

You + Celebrity + N-1

N

Amazon Int

N people



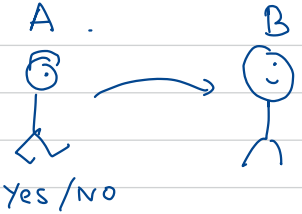
→ Everyone Knows ☆

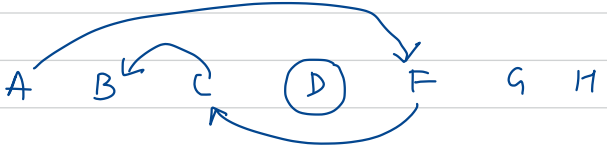
→ ☆ Knows no-one

→ Random ppl may / may not know each other

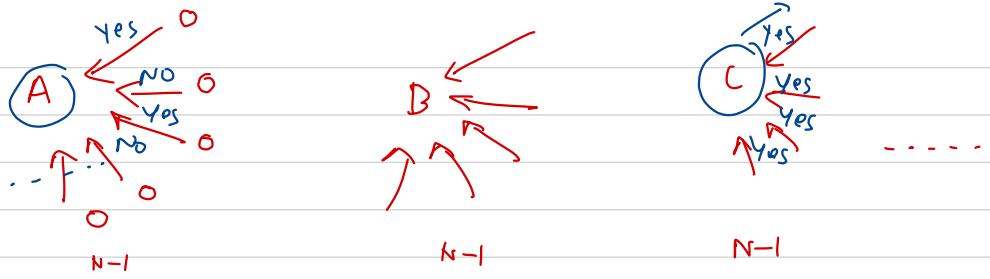
Q How to find the celebrity & how many min questions ?

"Do you Know him/her?"



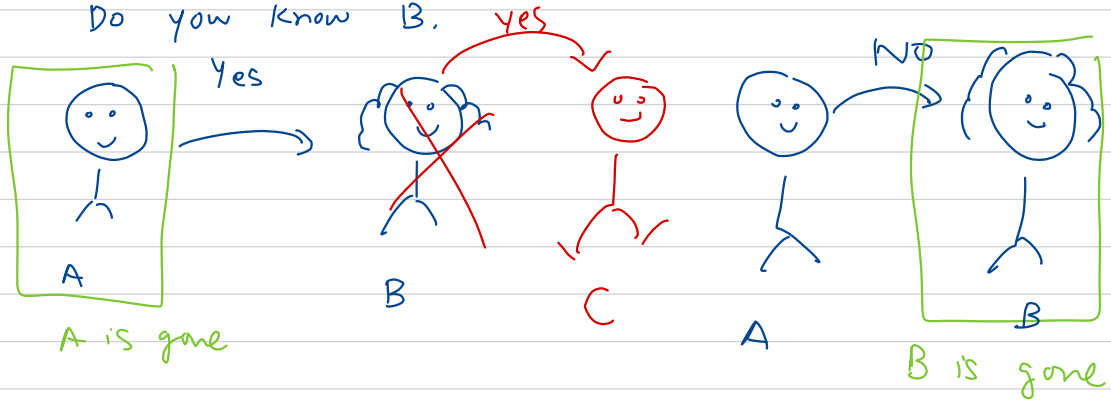


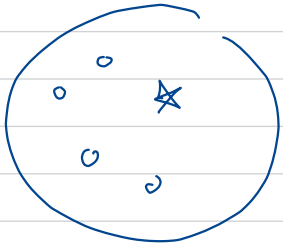
Approach

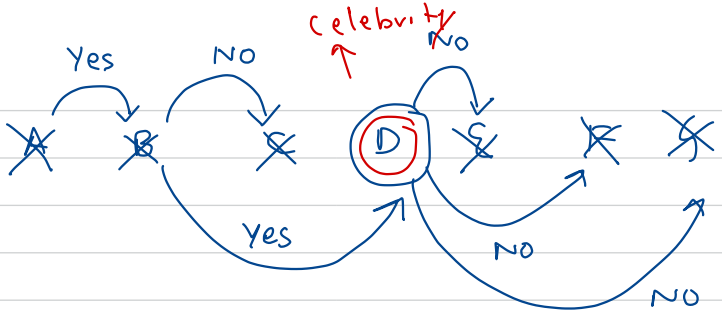


$\propto N(N-1)$ Questions

1 question
↓
1 person







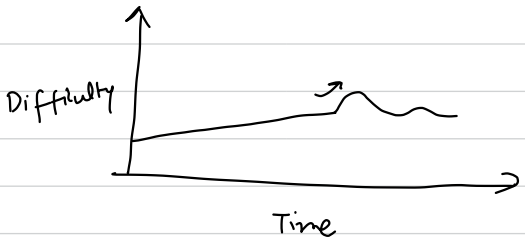
N-1 questions → N-1 ppl

fn knowsPersons (A , B)

↓

- [True / False]

Recursion



→ Problem

↓

Simpler Problems of the "Same Type"

1 + 1 + 1 + 1 + 1 = 5

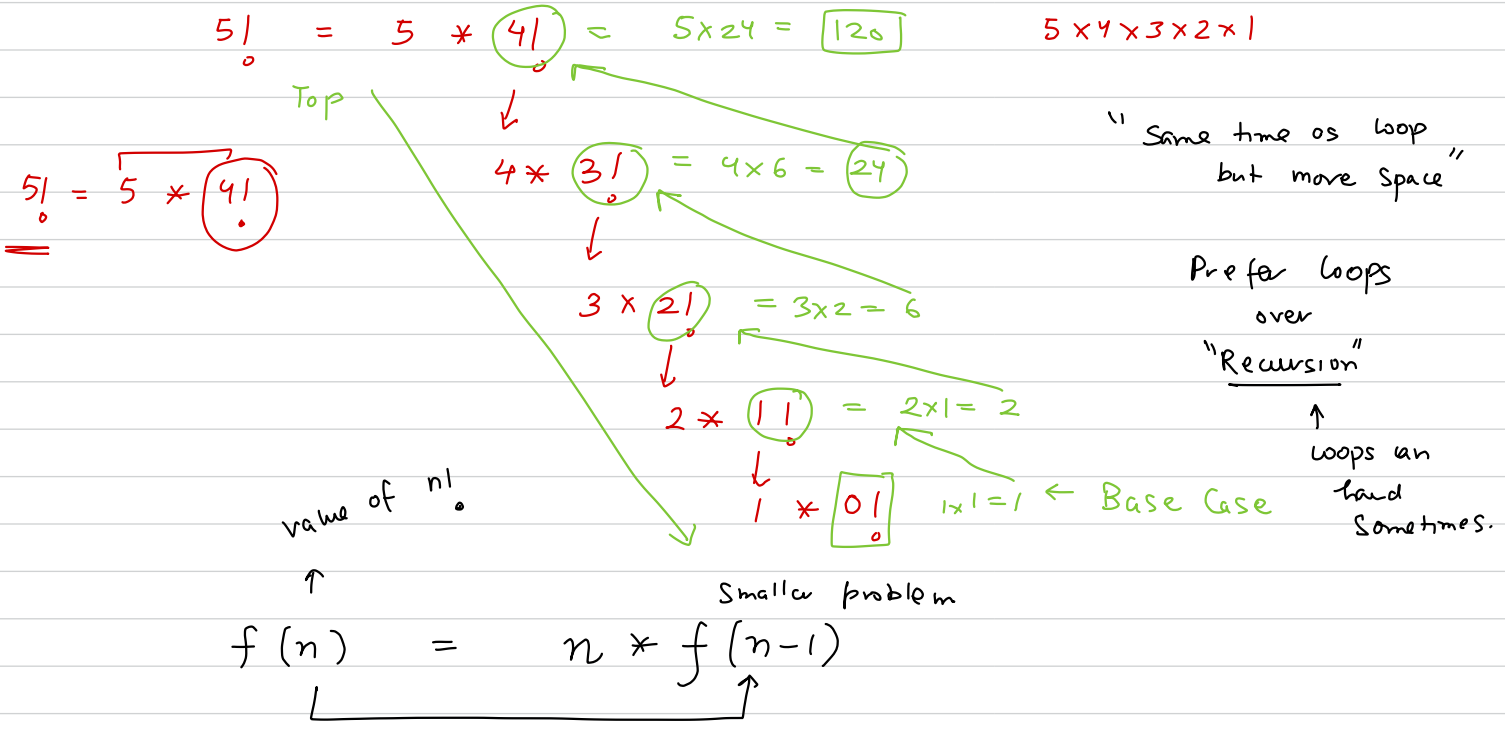
1 + 1 + 1 + 1 + 1 + 1 = 6

$f(6) = 1 + f(5)$

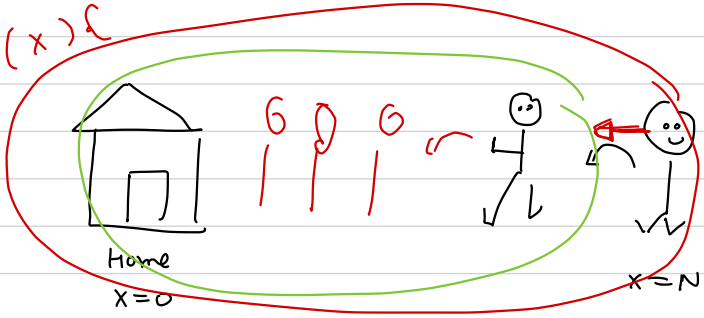
$1 + 5$

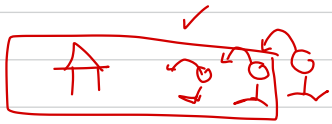
Smaller problem

Bigger


5! = 5 * 4! = 5x24 = 120
5 x 4 x 3 x 2 x 1
Top
4 * 3! = 4x6 = 24
3 x 2! = 3x2 = 6
2 * 1! = 2x1 = 2
1 * 0! 1x1=1 ← Base Case
5! = 5 * 4!
" Same time as loop but more Space "
Prefer loops over "Recursion"
↑
loops can land Sometimes.
value of n!
↑
Smaller problem
f(n) = n * f(n-1)

walk to Home (x) {





}

Base Case

```
if (x==0) {
    print (Reached Home).
    return;
}
```

Rec Case

```
walk 1 step.
walk To Home (x-1);
```

Recursion. Principle of Mathematical Induction

1 Find out the smallest case / Stop Solving

Hypothesis 2 ASSUME the problem can be solved for any given k. $k \in (1 \ldots N-1)$ $k < N$

f(k) → known

3 Solve the problem for

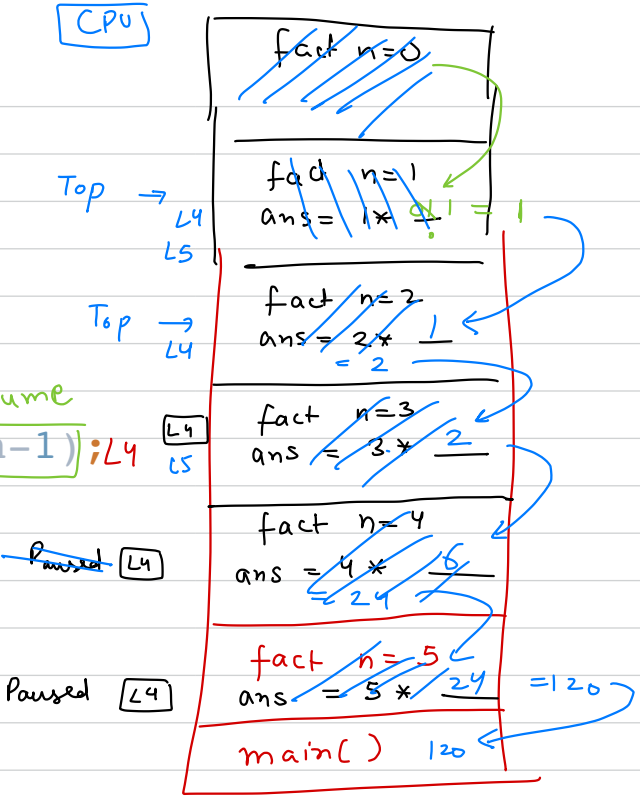
f(N) → f(k) ← Smaller sub problems.

f(k) $k < N$

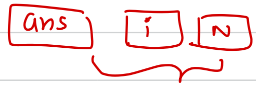
6, 7, 8, 9, 10

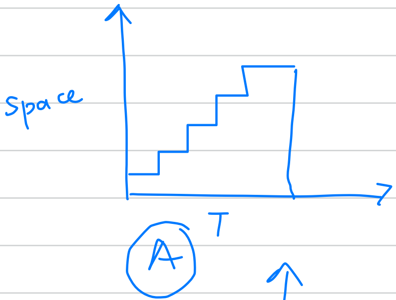
```
static int factorial(int n){
    // base case
    if(n==0){                    L1
        return 1;                L2
    }

    // recursive case
    int ans = n * factorial(n-1);  L4
    return ans;                  L5
}

main(){
    factorial(5);

}
```

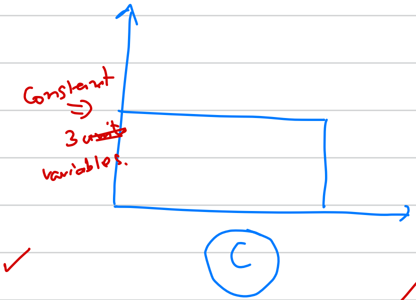
① ② L3 Assume ③ → fn call

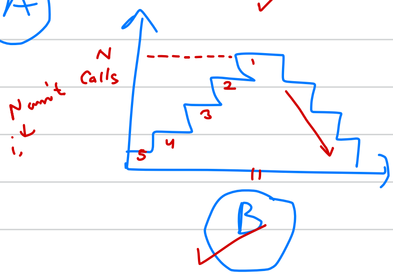


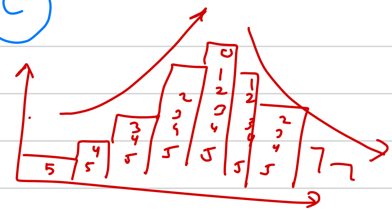
```
for (i=1; i<= N; i++)
    ans = ans * i
```



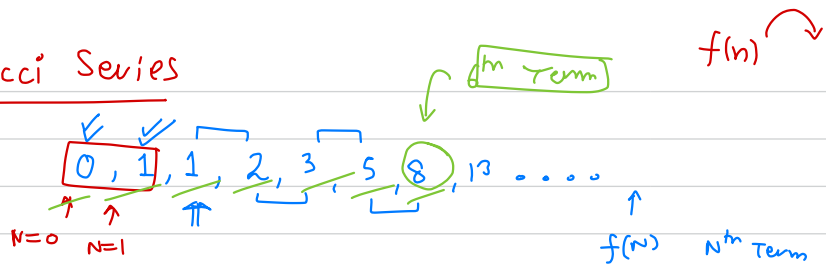


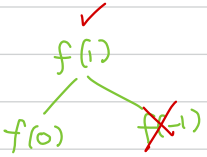






Fibonacci Series





N^{th} Term = (N-1) Term + (N-2) Term

② $f(N) = f(N-1) + f(N-2)$ Rec. Case

① $f(0) = 0$, $f(1) = 1$] → 2 values you need

Base case

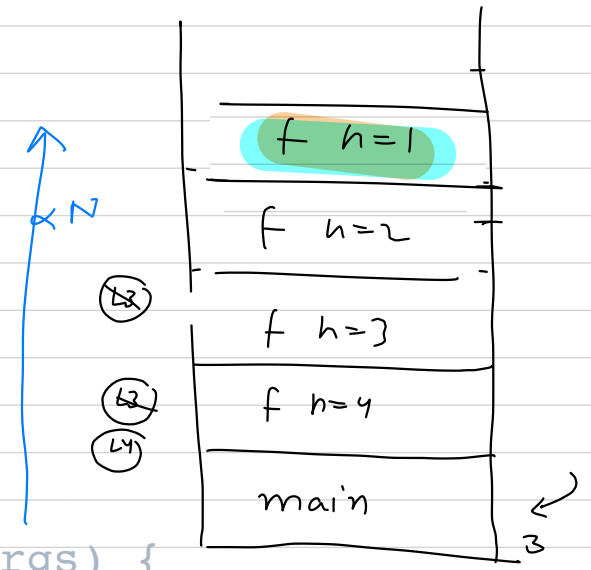
Assume $k < N$

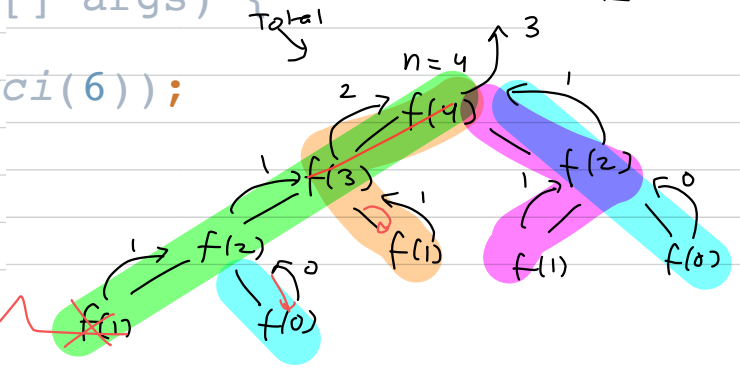
↓

Know

```java
static int fibonacci(int n){
    //base case
    if(n==0 || n==1 ){
        return n;
    }
    //recursive case
    int f1 = fibonacci(n-1);
    int f2 = fibonacci(n-2);
    int ans = f1 + f2;
    return ans;
}

public static void main(String[] args) {
    int n = 6;
    System.out.println(fibonacci(6));
}
```

L1 L2 L3 L4 L5 L6

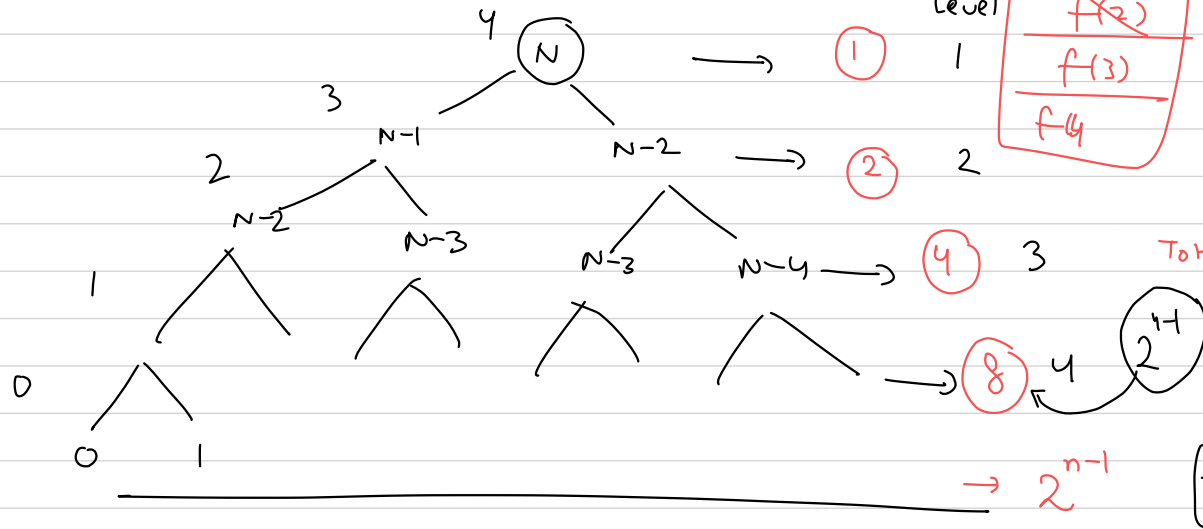


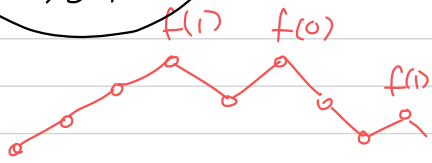


Time $\propto$ No of Rec Calls $\times$ Work (Addition)

GP

$\frac{a(r^n-1)}{r-1}$





Total = $1+2+4+8+\ldots 2^{N-1}$

$= \frac{2^n-1}{2-1}$

$\rightarrow 2^{n-1}$

Time $\propto 2^n$

Bad .c

N

Loop [10] steps Recursion $2^{10} = 1024$ steps

11 steps $= 2048$ steps

$$2^{N+1} = 2 \cdot (2^{N})$$

$$\underline{\underline{2}}^{N+1} = 2 \cdot (\underline{\underline{2}}^{N})$$

T

N+1 $= 2T$

N=100 5s

=101 10s

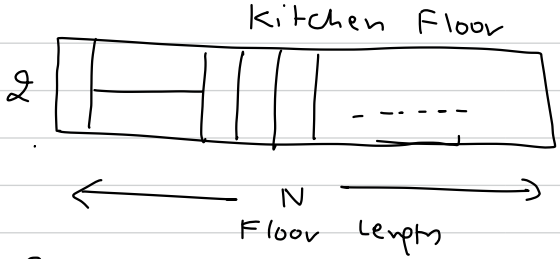
=102 20s

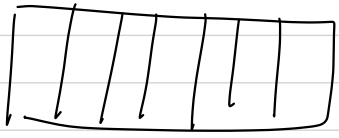
→ No of ways to build kitchen floor

(a) $f(N)$

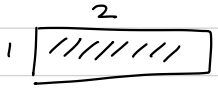
How many ways to build a floor of size N X 2 using tiles of size 1 X 2 ft placed horizontally or vertically.

ways ?

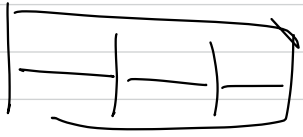




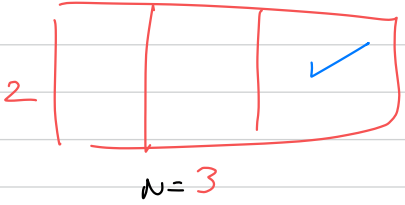
Tiles

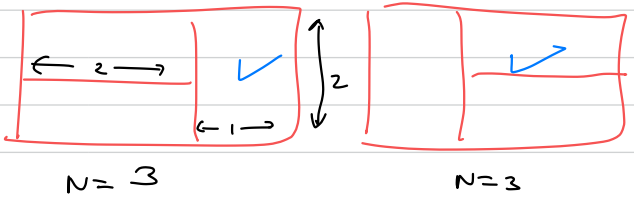


→ Horizontal
or Vertical Placement

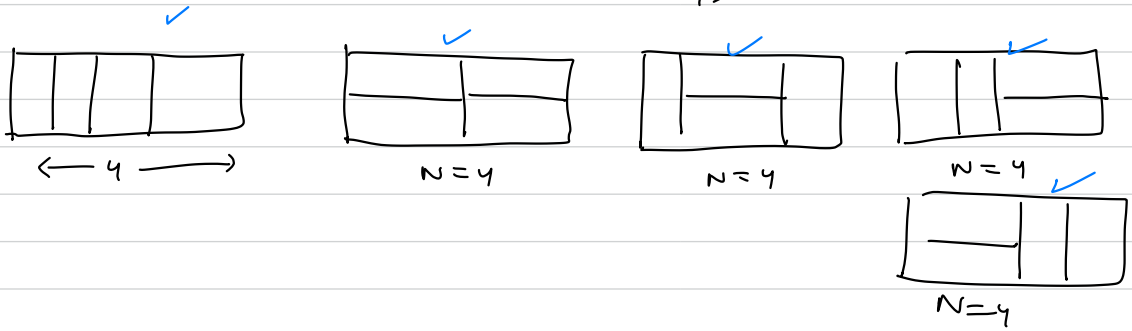


N = 3





5 ways



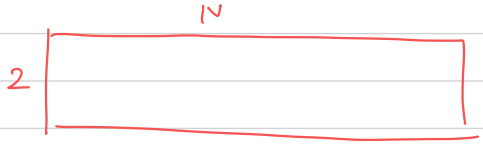
Recursively

HINT

Place 1 Tile + Can Recursion Solve for remaining Floor ?

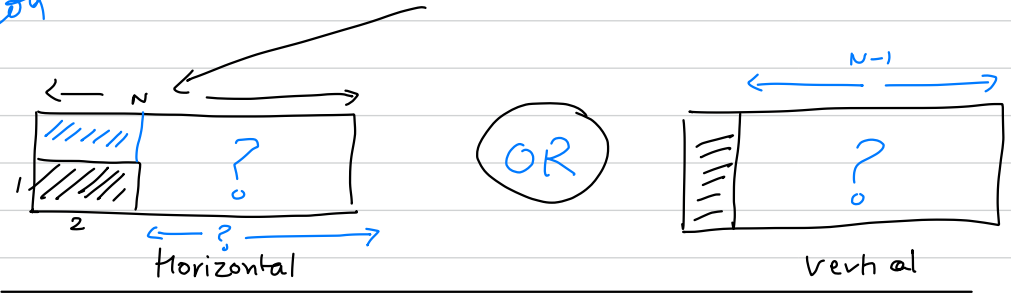
Think Rec + implement using Loop

f(N)



Floor

① Take one step + leave everything Rest

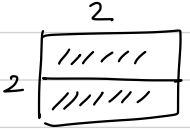


Time ↑ Space ↑ Loop ↓

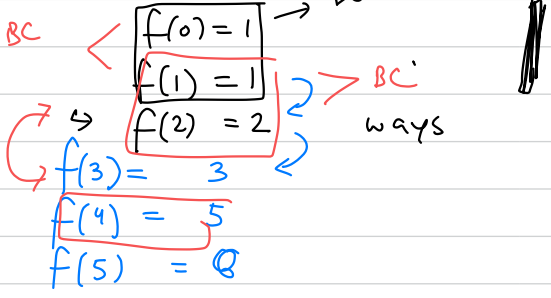
Recurrence

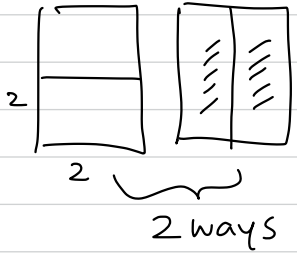
$$f(N) = f(N-2) + f(N-1)$$

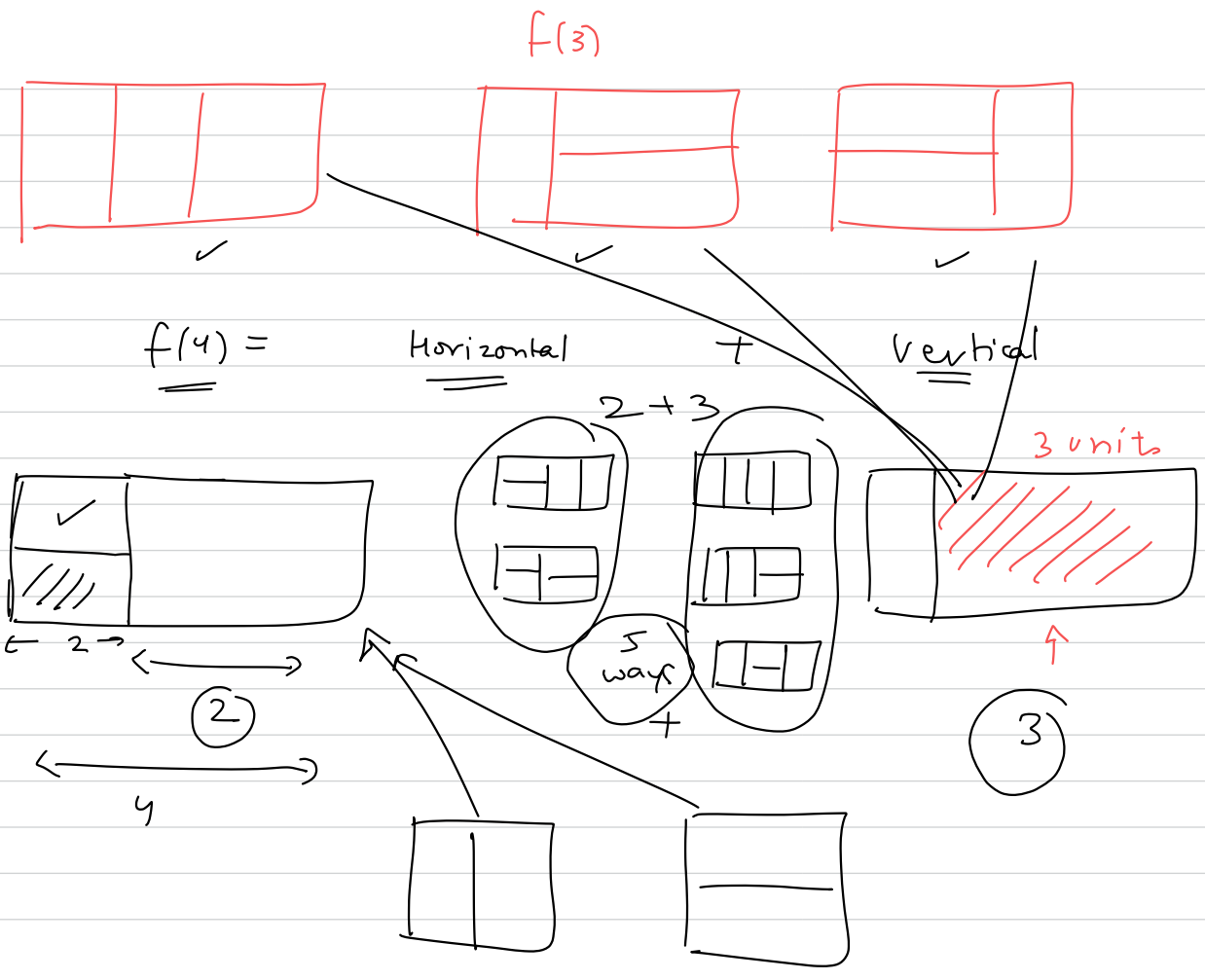
N=2



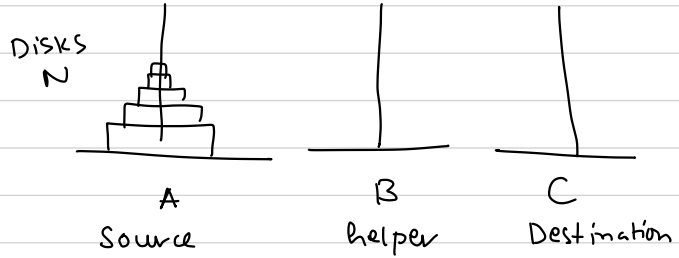
Smallest case






f(3)
f(4) =
Horizontal
+
Vertical
2 + 3
3 units
5 ways
2
4
3

Fun Problem (TOWER OF HANOI)

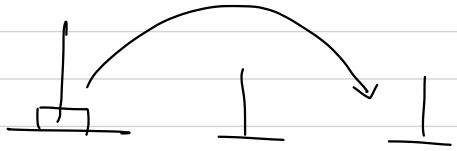


Goal. Move all N Disks to Dest from Source.

Conditions :

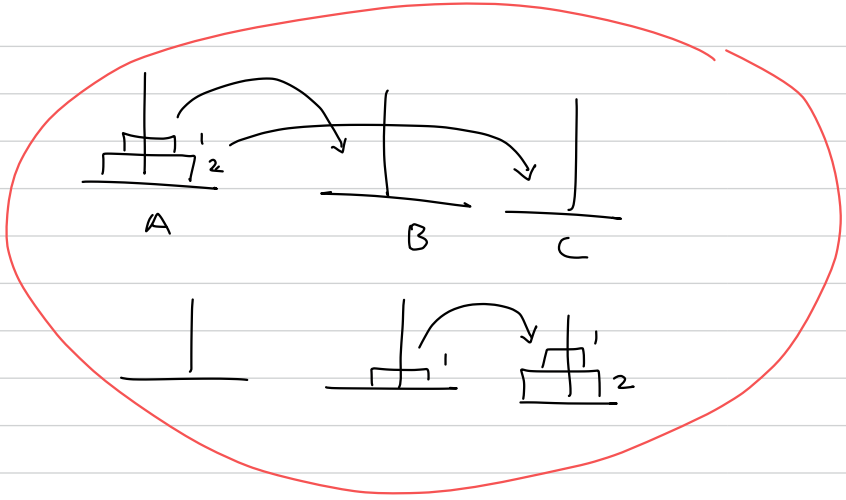
1. Pick only 1 Disk at time
2. You can't a place big Disk over small disk during the transfer

N = 1



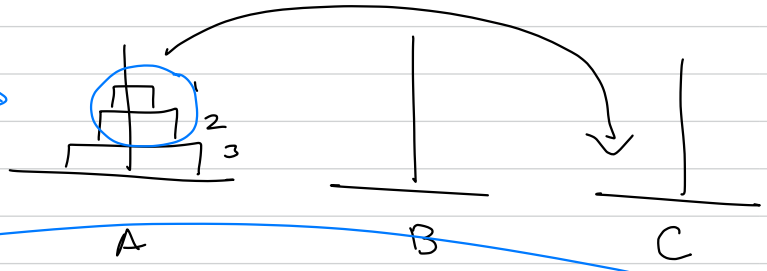
1 step

N = 2

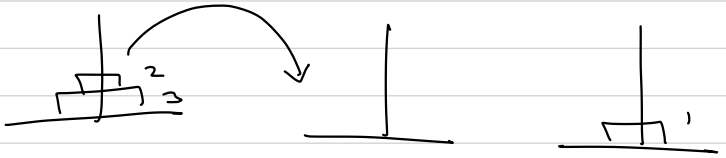


3 steps

N=3



S-1



TOH (2, Source to helper)

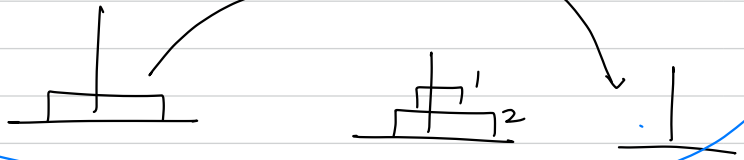
↳ Recursion

3 steps

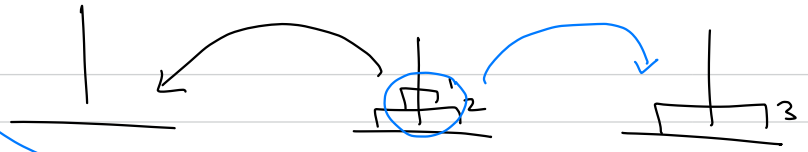
(Moved 2 disks)

S-2

S-3



S-4

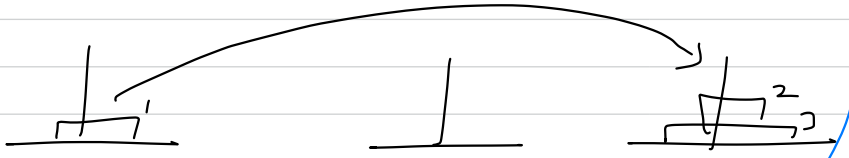


1 Step for Nth Disk

S-5

Recursion

S-6



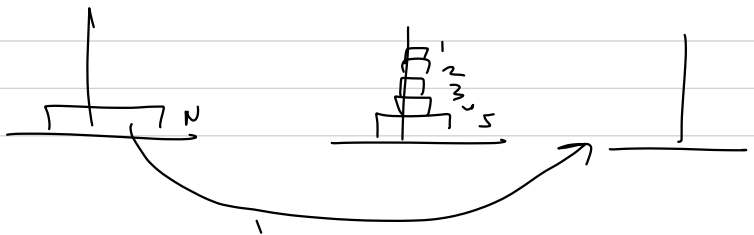
$2^N - 1$ steps for N Disks

S-7

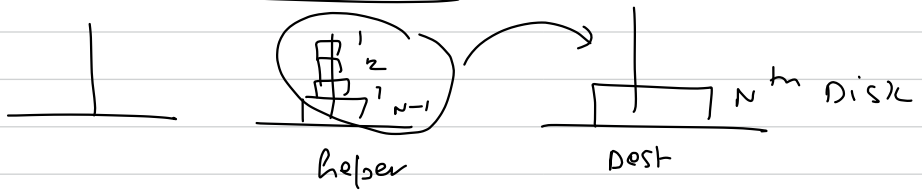
N=4 → 15 steps $(2^N - 1)$

Recursion

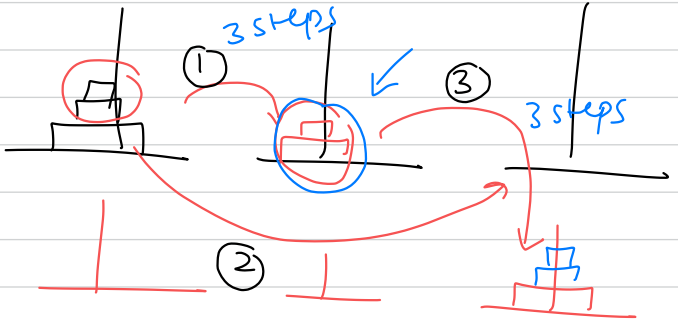
1. Recursively move N-1 disks from Source to Helper => TOH(N-1, Source, Helper)



2. Take disk N from source to dest



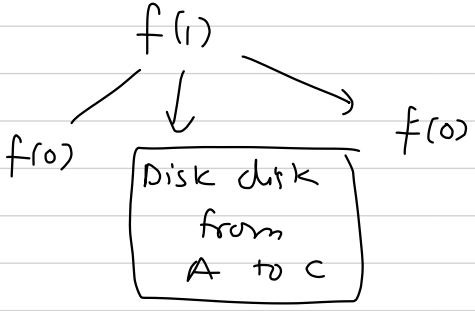
3. Recursively Move N-1 Disks from Helper to Destination => TOH(N-1, helper, dest)

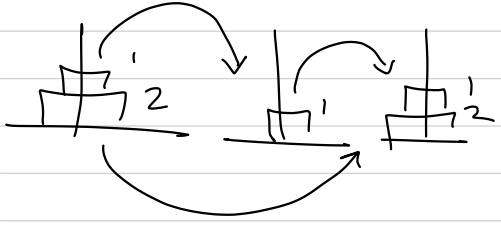


A C B

```
static void toh(int N, String source, String dest, String helper){
    //base case
    if(N==0){
        return;
    }

    //Otherwise
    toh(N-1,source,helper,dest);
    System.out.println("I am taking disk from "+source + " to " + dest);
    toh(N-1,helper,dest,source);
}
```

N=1





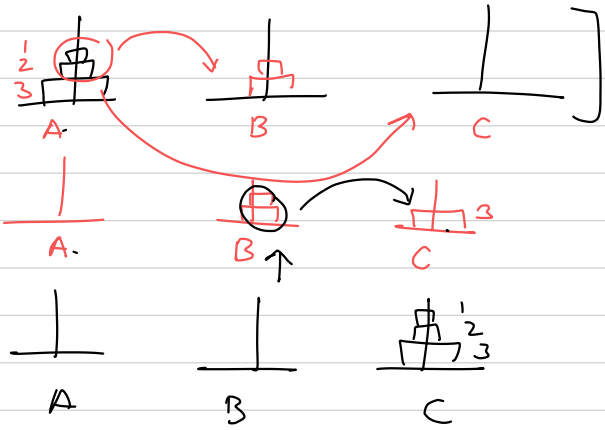
I am taking disk 1fromA to B
I am taking disk 2fromA to C
I am taking disk 1fromB to C

3

I am taking disk 1 from A to C
I am taking disk 2 from A to B
I am taking disk 1 from C to B
I am taking disk 3 from A to C
I am taking disk 1 from B to A
I am taking disk 2 from B to C
I am taking disk 1 from A to C

N-1
3
+
1 ← largest Pick
+
3
N-1

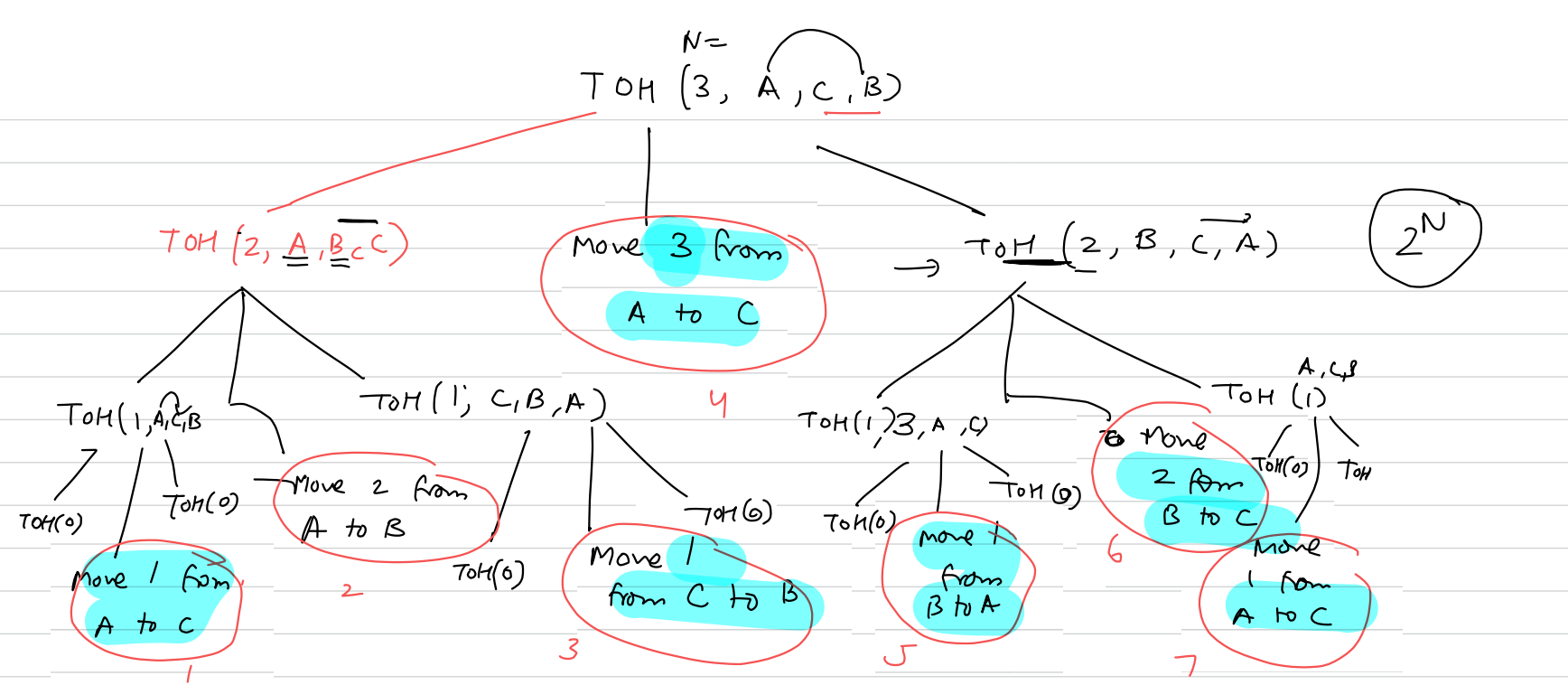
7 steps



TOH (N, S, D, H)

if (n == 0)
return

- TOH (N-1, S, H, D)
- move N from S to D
- TOH (N-1, H, D, S)



3

I am taking disk 1 from A to C
I am taking disk 2 from A to B
I am taking disk 1 from C to B
I am taking disk 3 from A to C
I am taking disk 1 from B to A
I am taking disk 2 from B to C

2^N

NO $2^N - 1$ (left Shift)

$= 1 << N - 1$

Tiles

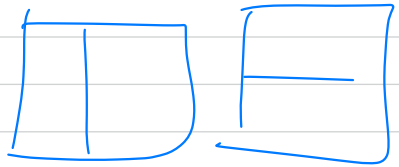
$f(1) = 1$

$f(0) = 1$

$f(2) = 2$

1 1

$f(1)$ $f(0)$



1 + 1

= 2