What is Recursion?

A function calling itself to solve a problem

Solving a problem using sub-problems of the same type is called Recursion

sum(N) = Sum of first N natural numbers

= 1 + 2 + 3 + - - - . . N-2 + N-1 + N

sum(N) = sum(N-1) + N

sum(4)

10 ↑ ↳ 4 + sum(3)

6 ↑ ↳ 3 + sum(2)

3 ↑ ↳ 2 + sum(1)

1 ↑ ↳ 1 + sum(0)

↳ 0 + sum(-1)

3 Steps of Recursion

Assumption: Define what your recursive function has to do and just assume it works.

Eg: sum(N) returns sum of N natural nos

Main logic: Using the function defined in step 1, break the bigger problem into smaller problems

sum(N) = sum(N-1) + N

Base Condition: When do you want to stop.

```
if( N == 0) return 0
```

```
int sum(int N) {
    if(N == 0) return 0;

    return sum(N-1) + N;
}
```

Question: Find Factorial

N! : $1 \cdot 2 \cdot 3 \cdot 4 \cdots \underbrace{N-1}_{(N-1)!} \cdot N$

5! : $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$

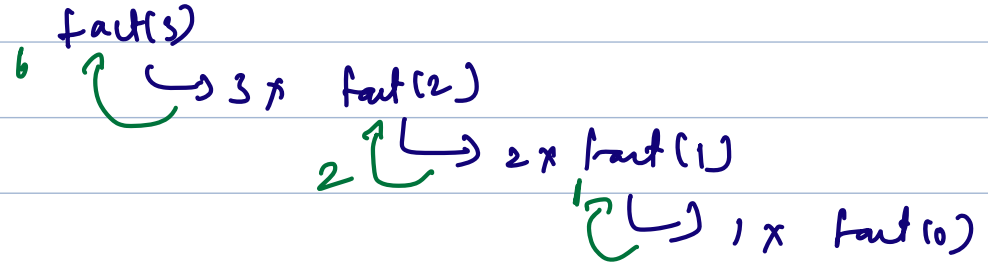
Assumption: fact(N) returns N!

Main Logic:

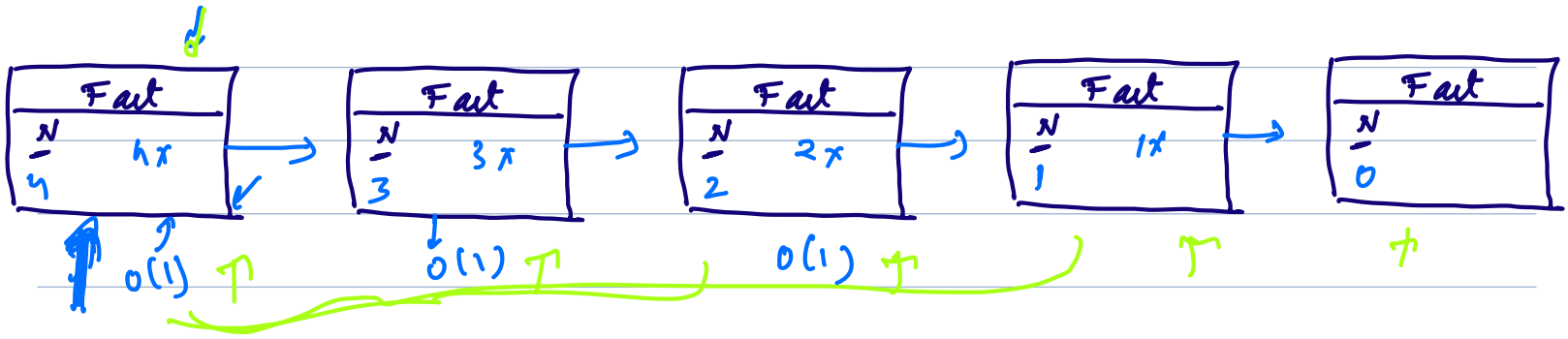
fact(N) = fact(N-1) x N

Base Condition

0! = 1



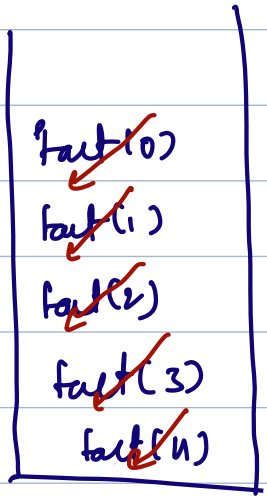

```
int fact(int N) {
    if(N == 0) return 1;
    return N * fact(N-1);
}
```



→ Whenever a function is called, some memory is allocated on top of call stack

→ After function completing execution, it is popped from the call stack

```
int fact(int N) {
1.      if(N == 0) return 1; -> O(1)
2.      return N * fact(N-1); -> O(1)
}
```

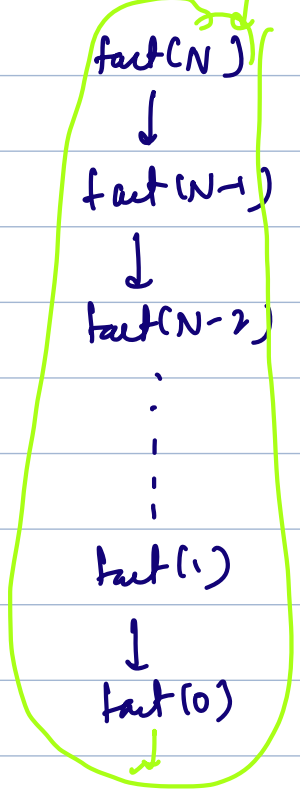


1 x 1 = 1

2 x 1 = 2

3 x 2 = 6

4 x 6 = 24



$[0, N] = N+1$

T.C: O(# Function calls × T.C per call) ✓

$(N+1) \times O(1) \Rightarrow O(N)$

S.C: O(# map ∩ function calls × S.C per active in stack function call)

$(N+1) \times O(1) = O(N)$

↳ Every recursive solution can also be implemented through iterative solutions

Question: Print 1 to N in increasing order

Assumption: printInc(N) prints 1 to N in T order

Main Logic

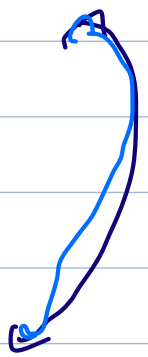
```
printInc(N-1);
print(N);
```

Base Case

```
if(N==1){
    print(1);
    return;
}
```

OR

```
if(N==0){
    return;
}
```



```
void printInc(int N){
    if(N==0) return;

    printInc(N-1);          // 1, 2, 3, ..., N-1
    print(N);
}
```

Question:

Problem

Whirlpool wants to design a timer for their **washing machines**. This feature is a simple countdown timer. When a user sets a time, for example, 10 minutes, the washing machine needs to show each minute passing, counting down until it reaches 0.

Your task is to write a program that takes an integer **A** (the time in minutes set by the user) and then prints out each minute as it counts down to **0**.The requirement is that after a user set a timer for the washing machine for some time say **A** , the washing machine should display each minute after that decremented one by one till the time becomes **0**.

Simplified Problem statement

Given **N**, print all numbers from **1 to N** in decreasing order.

5 : 5 4 3 2 1

```
void printDec(int N){
if(N==0) return;

print(N);
printDec(N-1);
}
```

T.C: $N \times O(1) = O(N)$

S.C: $N \times O(1) = O(N)$

8:23

Question: Find N^{th} number in fibonacci series

0	1	1	2	3	5	8	13	21	34	55	
0	1	2	3	4	5	6	7	8	9	10	.. .. ..
-	-	↑	↑	↑	↑	↑	↑				

fib(N) = fib(N-1) + fib(N-2)

Assumption: fib(N) returns N^{th} Fibonacci Series

Main Logic

fib(N) = fib(N-1) + fib(N-2);

Base Case:

if(N == 0) return 0;

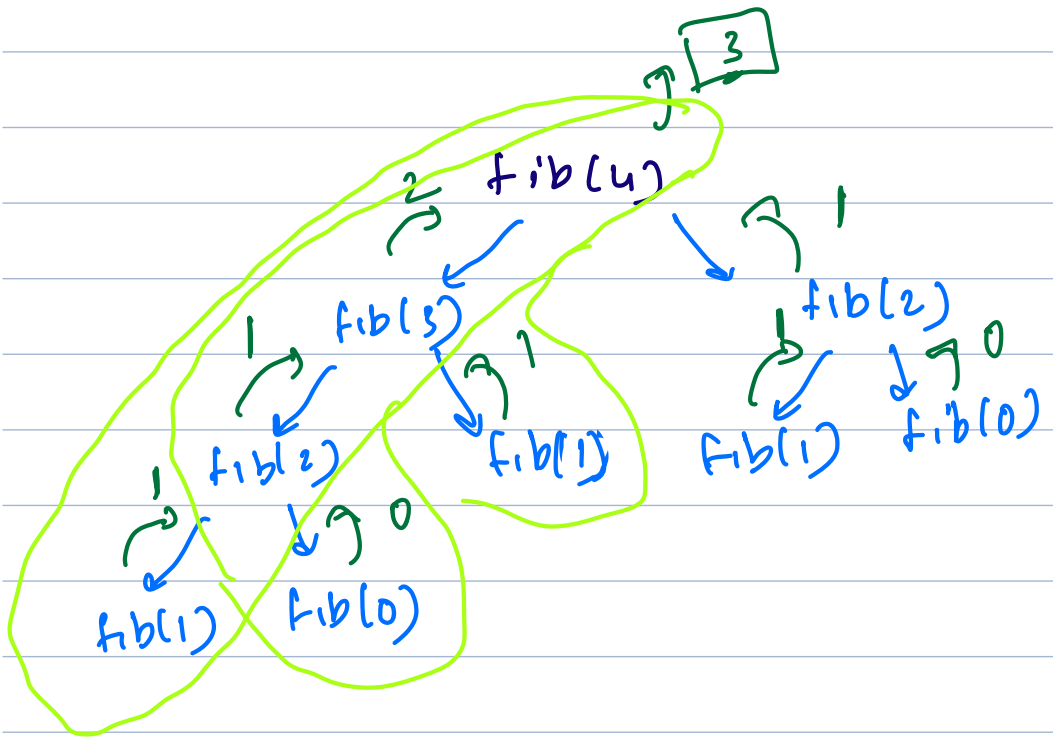
if(N == 1) return 1;

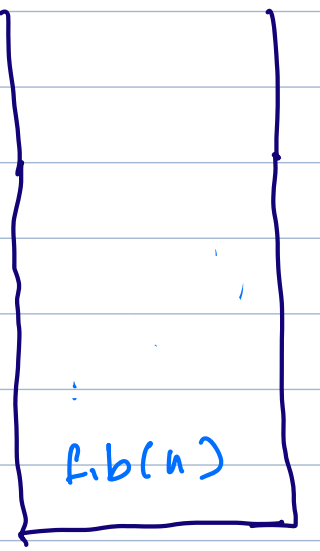
Note: In most of the cases, the # base cases = # distinct recursive sub-problems

```
int fib(int N){
    if(N==0 || N==1) return N;
    return fib(N-1) + fib(N-2);
}
```

T.C: O(1)

S.C: O(1)

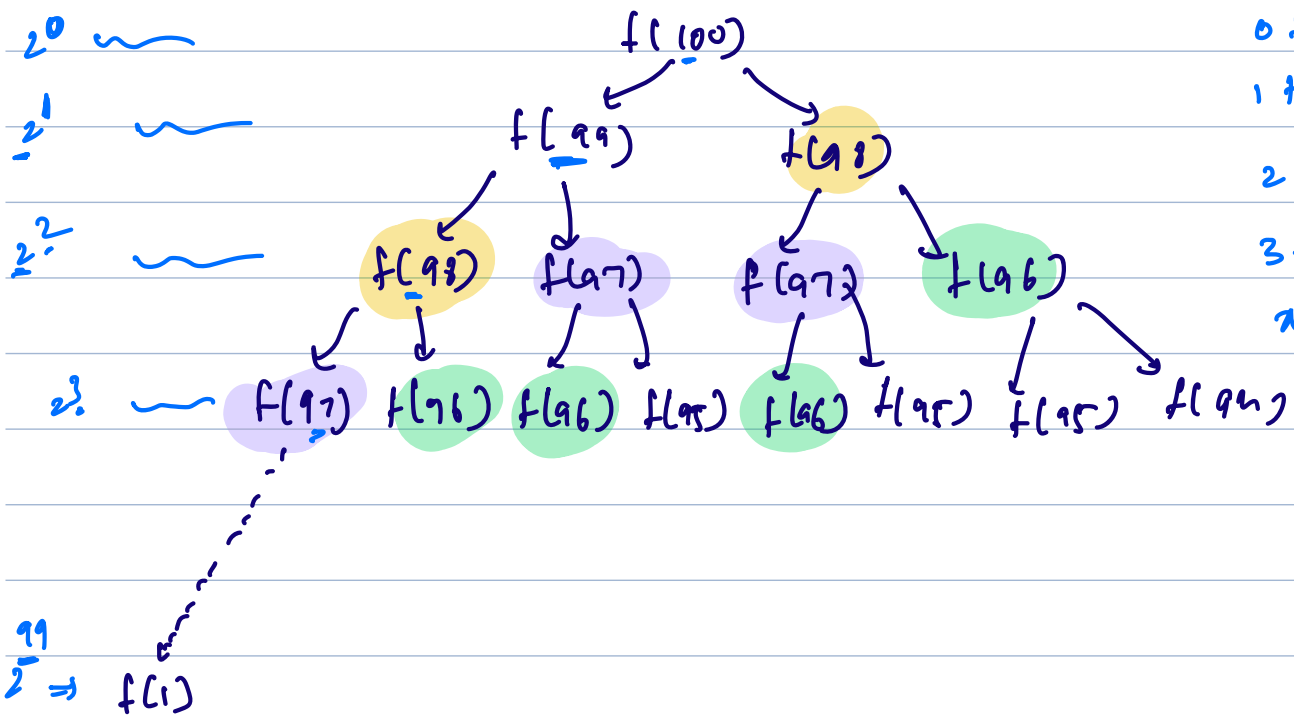




call stack

S.C: N x O(1) ⇒ O(N)

T.C:



$[0, N-1]$

Function calls : $2^0 + 2^1 + 2^2 + \cdots + 2^{N-1}$

$$Sum_K = \frac{a(r^K - 1)}{r - 1}$$

$a = 1$
$r = 2$
$K = N$

$$\therefore \frac{1[2^N - 1]}{2 - 1} = 2^N - 1$$

T.C. : $(2^N - 1) \times O(1) = O(2^N)$

$\Downarrow$ DP

$O(N)$ ✓ $N = 10^6$

$2^{20} \approx 10^6$

$2^{23} \approx 10^7$

```
for(i=1; i<=N; i++){
    print("Hi");
    -> func()                    // N^2
}
```

$O(N^3)$

```
sum(N) {
    return sum(N-1) + N;
}
```

sum(N-1) → sum(N-2) → sum(N-3) . . . . . sum(0)

(N-1)

sum(N) → sum(N-1) → sum(N-2) ⋯ sum(0)

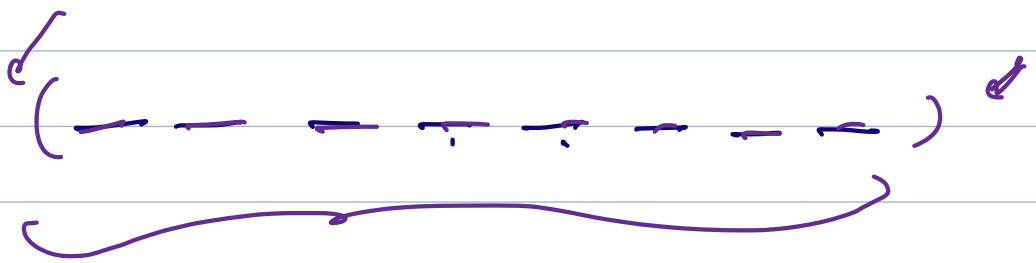
2 operations 2 opers 2 opers 2 opers

$(N+1) \times 2$ ops/iter $= 2N + 2$

$\Downarrow$

$O(N)$

$\simeq 10^6 - 10^8$



-128

$a = 10^5$
$b = 10^6$
$a \times b$

int c = a x b;