

## Today's Content

- Recursion
- How to write recursive code / trace
- TC/SC of recursive code in Next class.

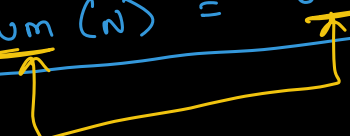
## Why Recursion?

- Merge Sort / Quick Sort
- Binary Tree / BST / Balanced BST / Segment Trees / Tries
- Dynamic Programming
- Backtracking
- Graphs

Recursion ?  $\Rightarrow$  Function calling itself

$\hookrightarrow$  Solving a problem using smaller instance of same problem  
 $\downarrow$   
sub-problem

$$\text{Sum}(N) = 1 + 2 + 3 + \dots + N-1 + N$$

$$\boxed{\text{Sum}(N) = \text{Sum}(N-1) + N}$$


$$\text{Sum}(4) = \text{Sum}(3) + 4$$

$\hookrightarrow$  sub-problem

Qn: How to write recursive code?

Assumption: Fix what your function should do

Main Logic: Solve the assumption using the problem

Base Condition: Inputs for which you want to stop the recursion.

Problems

int sum(N) { Assumption: Given N, calc & return the sum of first N natural numbers.  $(N \geq 1)$   
if (N == 1) { return 1 }  
return sum(N-1) + N  
}

Main Logic:

$$\text{sum}(N) = 1 + 2 + 3 + \dots + N-1 + N$$

$$\text{sum}(N) = \text{sum}(N-1) + N$$

$$\begin{aligned}\text{sum}(1) &= \text{sum}(0) + 1 \\ &= 1.\end{aligned}$$

Qn:  $fact(N) ? (N \geq 1)$   
 $fact(3) = 3 \times 2 \times 1 = 6$  ,  $fact(4) = 4 \times 3 \times 2 \times 1 = 24$ .

int  $fact(N) \{$  Assumption: Given  $N$ , find & return  $N!$

if  $(N == 1) \{ return 1 \}$

Main Logic:

$fact(N) = N \times N-1 \times N-2 \times \dots \times 3 \times 2 \times 1$

$fact(N) = N \times fact(N-1)$

return  $N \times fact(N-1)$

}

### Function Call Tracing

int  $add(N, m) \{$   
 | return  $N + m$   
 }

int  $mul(x, y) \{$   
 | return  $x * y$   
 }

int  $sub(a, b) \{$   
 | returns  $a - b$   
 }

main() {

$x = 10, y = 20$

|  $print(sub(mul(add(x, y), 30), 75))$   
 }

$sub(\overset{900}{mul(add(x, y), 30)}, 75) : \text{returns } 825$

$\hookrightarrow mul(\overset{30}{add(x, y)}, 30) : \text{returns } 900$

$\hookrightarrow add(x, y) : \text{returns } 30$   
 $10, 20$

Output: 825

# // Data Structure

LIFO structure

Last-In  
First Out

add(x, y) : returns (30) [remove]

mul(add(x, y), 30) : returns (900) [remove]

sub(mul(add(x, y), 30), 75) : returns 825  
[removed]

DS: Stack  
↳ System handles the implementation

## Sum Tracing

```
int sum(N=4) {  
    if(N==1) { return 1 }  
    return sum(N-1) + N  
}
```

3

↑ 6 + 4

returns 10

```
int sum(N=3) {  
    if(N==1) { return 1 }  
    return sum(N-1) + N  
}
```

3

↑ 3 + 3

```
int sum(N=2) {  
    if(N==1) { return 1 }  
    return sum(N-1) + N  
}
```

3

↑ 1 + 2

```
int sum(N=1) {  
    if(N==1) { return 1 }  
    return sum(N-1) + N  
}
```

3

## Stack

```
sum(1): return 1  
sum(2): return sum(1) + 2 1 + 2  
sum(3): return sum(2) + 3 2 + 3  
sum(4): returns sum(3) + 4 = 10
```

What happens if base condn not there?

a) TLE      b) MLE ✓

Note: In recursion, if you get MLE, verify base condition.

Qn:  $N \geq 0$  (Sum of prev 2 numbers) : Fibonacci

	0	1	2	3	4	5	6	7	8	9	10
# input(N)	0	1	2	3	4	5	8	13	21	34	55
Fib()	0	1	1	2	3	5	8	13	21	34	55

$$\text{fib}(5) = 5$$

$$\text{fib}(9) = 34$$

int fib(N) { Assumption: Given N, Calc & return  $N^{\text{th}}$  fibonacci no.

if( $n \leq 1$ ) { return n }

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

Main Logic

fib(N) = sum of prev 2 terms

$$\text{fib}(N) = N-1 + N-2$$

$$\text{fib}(N) = \text{fib}(N-1) + \text{fib}(N-2)$$

Note: how to figure out base condition.

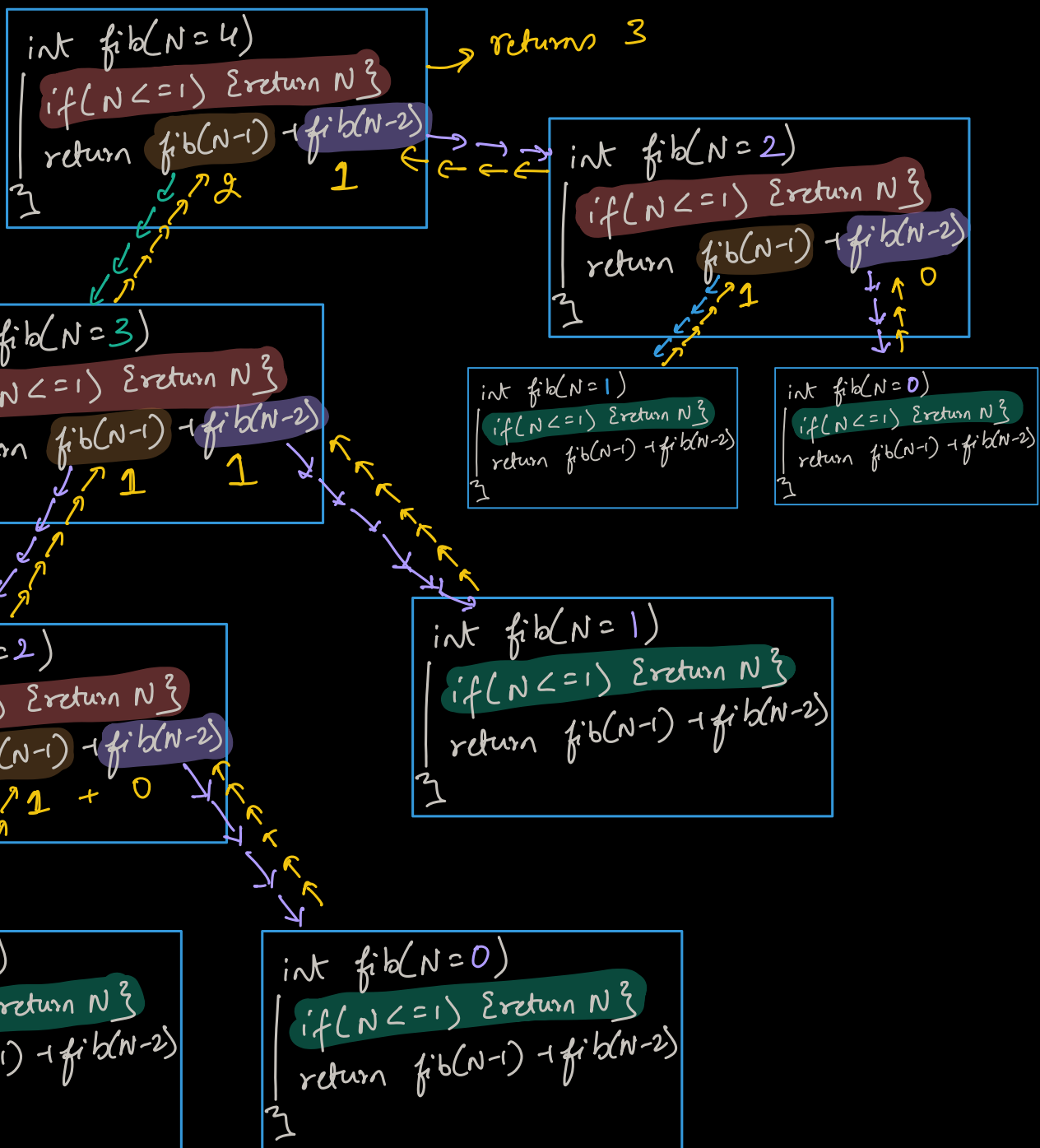
↳ For which valid inputs, expression is invalid.

$$\text{fib}(0) = \text{fib}(-1) + \text{fib}(-2) \times$$

$$\text{fib}(1) = \text{fib}(0) + \text{fib}(-1) \times$$

$$\text{fib}(2) = \text{fib}(1) + \text{fib}(0) \checkmark$$

// Trace  
out:



\* Try out stack tracing  $\Rightarrow$  TODO

Break till: 8:30am

Qn: Given  $N$ , print all numbers from 1 to  $N$  in increasing order

$\text{inc}(3) \rightarrow 1 \ 2 \ 3$

$\text{inc}(4) \rightarrow 1 \ 2 \ 3 \ 4$

$\text{void inc}(N) \{$  Assumption: Given  $N$ , print all numbers from 1 to  $N$  in inc. order.

$\text{if}(N==1) \{ \text{print}(1), \text{return}; \}$

$\text{inc}(N-1)$   
 $\text{print}(N)$

Main Logic:

$\text{inc}(N): 1 \ 2 \ 3 \ 4 \dots N-1 \ N$

$\downarrow$   
 $\text{inc}(N-1)$   
 $\text{print}(N)$

Trace:

$\text{void inc}(N=4)$   
 $\text{if}(N==1) \{ \text{print}(1), \text{return} \}$   
 $\text{inc}(N-1)$   
 $\text{print}(N)$

$\text{void inc}(N=3)$   
 $\text{if}(N==1) \{ \text{print}(1), \text{return} \}$   
 $\text{inc}(N-1)$   
 $\text{print}(N)$

$\text{void inc}(N=2)$   
 $\text{if}(N==1) \{ \text{print}(1), \text{return} \}$   
 $\text{inc}(N-1)$   
 $\text{print}(N)$

$\text{void inc}(N=1)$   
 $\text{if}(N==1) \{ \text{print}(1), \text{return} \}$   
 $\text{inc}(N-1)$   
 $\text{print}(N)$

Output:

$\text{print}(1)$   
 $\text{print}(2)$   
 $\text{print}(3)$   
 $\text{print}(4)$

\* Note: Once the function is completely executed, it'll go back to the parent function.

⊗ Stack trace  $\Rightarrow$  To do

\* We can return from a void fn.



Qn: Given N, print all numbers from N to 1 (dec. order)  
dec(N) H.W.

Qn: Given a substring, check if it's a palindrome or not.  
0 1 2 3 4 5 6  
good dad  $s=4, e=6 \Rightarrow \text{true}$   $s \leq e$

Eg: good dad

0 1 2 3 4 5 6  
good dad

$s=2, e=5 \Rightarrow \text{false}$

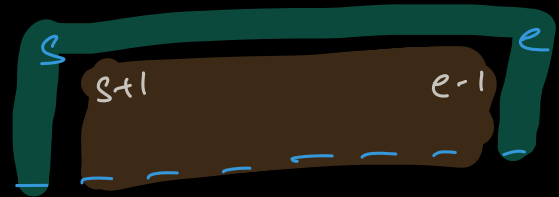
bool isPal(char ch[], int s, int e) {  
    Assumption: Return if substring [s e] is a palindrome.  
    if ( $s > e$ ) { return true; }

    return  $(ch[s] == ch[e] \ \&\& \text{isPal}(ch, s+1, e-1))$

}

Main Logic:

ch[] =



① if ( $ch[s] == ch[e]$ )

② substring [s+1, e-1] should be a palindrome

isPal(ch, s+1, e-1)

s e ✓  
m a d a m

m a l y a l a m

0 1 2 3 4  
m a d a m

$s=0, e=4$   $\rightarrow$  return true

```
bool isPal(ch[], s=0, e=4)
{
    if(s > e) {return true;}
    return (ch[s] == ch[e] && isPal(ch, s+1, e-1))
}
```

```
bool isPal(ch[], s=1, e=3)
{
    if(s > e) {return true;}
    return (ch[s] == ch[e] && isPal(ch, s+1, e-1))
}
```

```
bool isPal(ch[], s=2, e=2)
{
    if(s > e) {return true;}
    return (ch[s] == ch[e] && isPal(ch, s+1, e-1))
}
```

```
bool isPal(ch[], s=3, e=1)
{
    if(s > e) {return true;}
    return (ch[s] == ch[e] && isPal(ch, s+1, e-1))
}
```

return false

```
if(s>e) {return true}
```

T

```
if(s>e) {return true}
```

7

```
if(s>e) {return true}
```

F