

## Recursion

→ function is calling itself

- ① Divide & conquer      ↓  
Recursion
- ② Dynamic Programming ←  
factorial of given number

$$5! = 5 \times 4 \times 3 \times 2 \times 1$$

$$= 120$$

Bare call

<u>condition</u>
$1! = 1$
$0! = 1$

$$6! = 6 \times 5!$$

$$|$$

$$5 \times \frac{4!}{1}$$

$$4 \times \frac{3!}{1}$$

$$3 \times \frac{2!}{1}$$

$$2 \times 1!$$

Recursion

1

Recursive code

T(n)

fact(n) ↗

(<sup>C</sup>)  
Bare call condition

if( <u>n == 0 OR n == 1</u> )
return 1;

else

Recursive call  
fact(n-1);

Recurrence Relation

$$T(n-1) + c$$

Complete

DSA in

Python

$$\begin{aligned} T(n) &= T(n-1) + c \\ &= O(n) \end{aligned}$$

$$\begin{array}{r} 120 \\ \hline - fact(5) \\ \hline 24 \end{array}$$

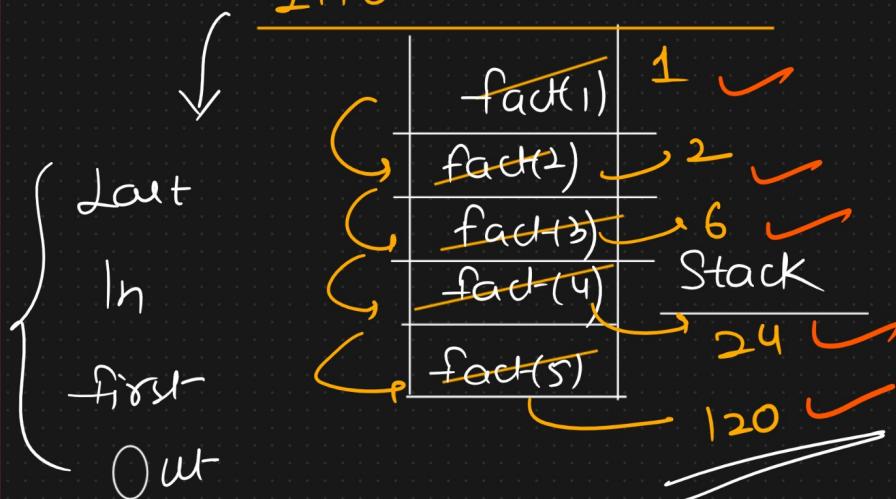
$$S * fact(4)$$

$O(n)$

$$4 * \frac{6}{fact(3)}$$

Space Complexity

LIFO Architecture



$$3 * fact(2)$$

$$2 * \frac{1}{fact(1)}$$

Bare Case Condition

Repetitive

-task

Recursion

Iterative

for, while,

do-while

Calling the  
same function  
again

different  
set of parameters

- Easy to understand
- Lesser line of code

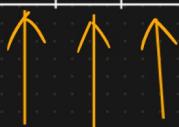
(2)

Fibonacci Series

$n = 12$

0	1	2	3	4	5	6	7	8	9	10	11	$(n+1)$
---	---	---	---	---	---	---	---	---	---	----	----	---------

0	1	1	2	3	5	8	13	21	34	55	89
---	---	---	---	---	---	---	----	----	----	----	----



$n = 8$

$\text{fib}(n)$

Recursive Approach

$O(2^n)$

Recursive tree

Recursive fib(4)

tree approach

$C_3$

1

$C_4$

$\text{fib}(2)$

(S)

1

$\text{fib}(0)$

$C_7$

1

$\text{fib}(1)$

$\text{fib}(2)$

$\text{fib}(1)$

$\text{fib}(0)$

$C_1$   
 $\text{fib}(5)$

$C_2$

Y

Time limit  
Exceeded

↳ Recursion

Memoization

Dynamic  
Programming

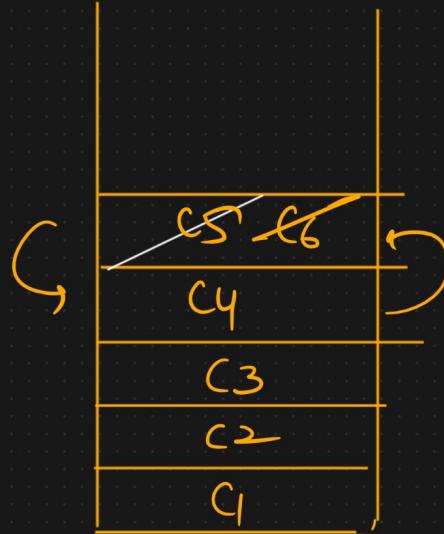
Overlapping  
Subproblems

if ( $n == 0$  OR  $n == 1$ )  
return  $n$ ;

else

$T(n-1) + T(n-2)$   
return  $\underline{\text{fib}(n-1) + \text{fib}(n-2)}$ ;

Y



Trade off b/w time & space complexity

$$T(n) = \begin{cases} C & n \propto 1 \\ T(n-1) + T(n-2) & n \propto 1 \end{cases}$$

*Exponential*

$\mathcal{O}(2^n)$

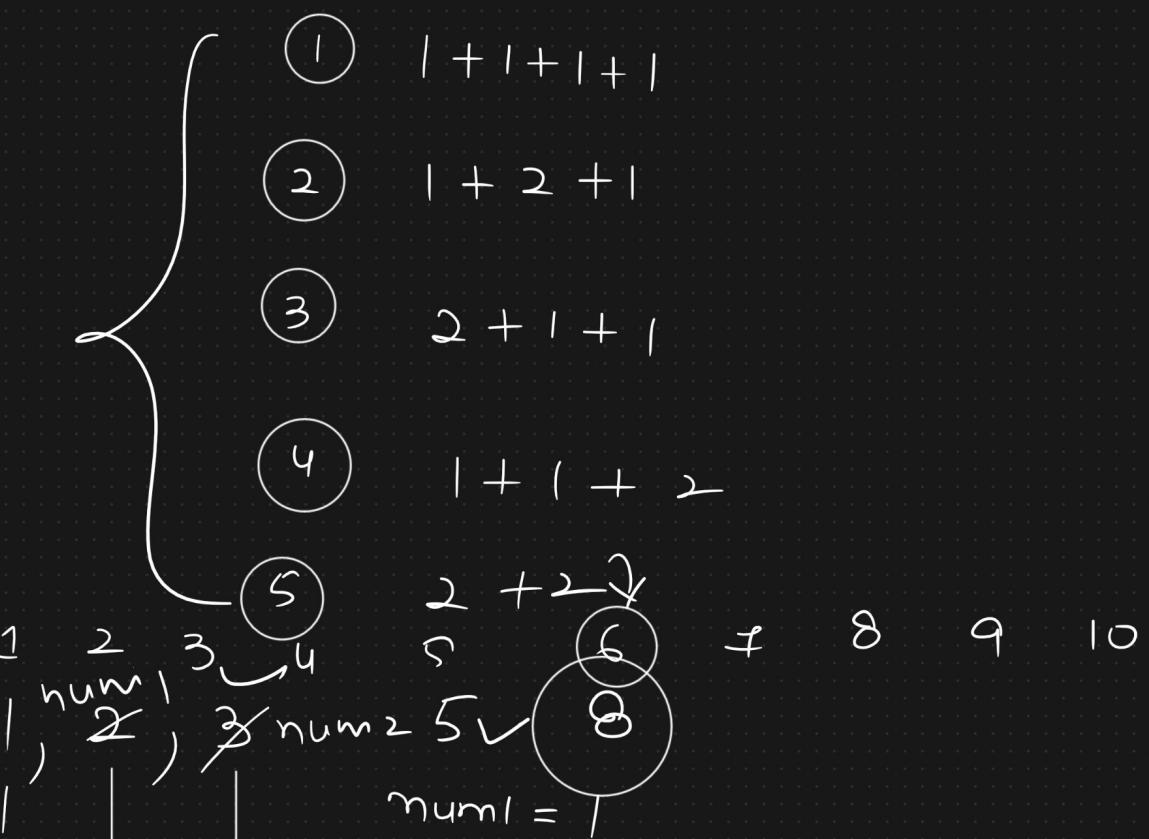
DP

$\downarrow$

$\mathcal{O}(n)$

*Linear*

$n=4$



$\text{for } (i=3 \text{ to } n) \text{ do}$

$i=3$

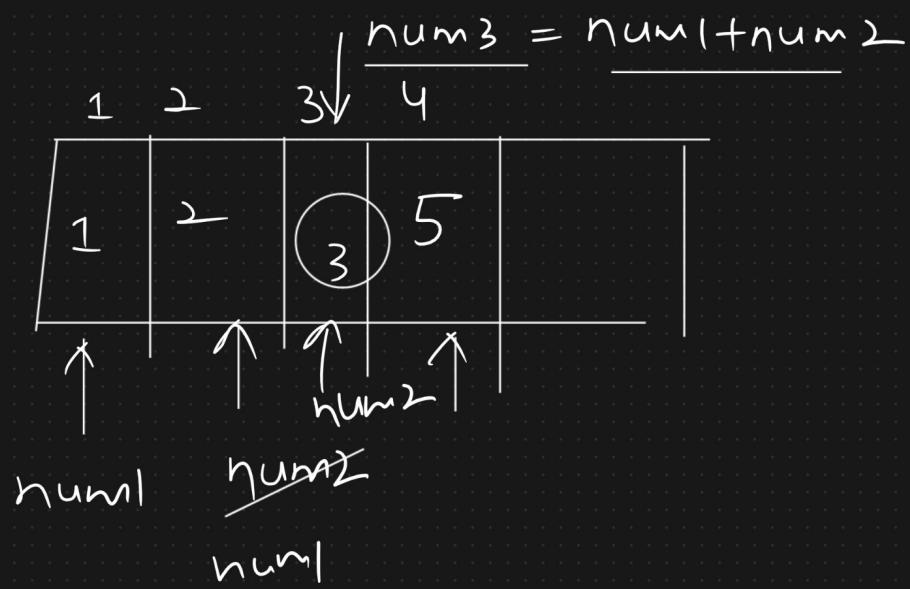
$\text{num}_3 = \text{num}_1 + \text{num}_2;$

$\text{num}_1 = \text{num}_2$  \_\_\_\_\_

$\underline{\text{num}_2} = \underline{\text{num}_3}$  \_\_\_\_\_

$O(n)$

return  $\text{num}_2;$



## Applications of Divide &

### Conquer

No problem of Overlapping Subproblem → MergeSort  
 → QuickSort  
 → Binary Search

## Sum of Digits

Interview  
Question

$$\text{num} = 1234$$

$$\begin{aligned}\text{Output} &= \underline{1} + \underline{2} + \underline{3} + \underline{4} \\ &= 10\end{aligned}$$

$$\begin{array}{r} \text{num}/10 \\ \text{num} \% 10 \end{array} \quad \begin{array}{l} \text{Remainder} \\ \swarrow \end{array}$$
$$\text{num} \% 10 = \boxed{1234} \% 10 = 4$$

SumofDigits( int n )  
 $n = 1234$

$$n = 1234 \quad \text{return } (\underline{n \% 10}) + \text{sumofDigits}(n/10)$$

$$4 + \underline{\text{sum}(123)}$$



$$3 + \underline{\text{sum}(12)}$$



$$2 + \underline{\text{sum}(1)}$$



10  
\_\_\_\_\_

$$\begin{array}{c}
 \text{num} = 1234 \\
 \text{num} \% 10 \\
 \downarrow \\
 4 + \frac{\text{sum}(123)}{\text{num}/10} \\
 \downarrow \\
 3 + \frac{\text{sum}(12)}{\text{num}/10} \\
 \downarrow \\
 123 \% 10 \\
 \downarrow \\
 2 + \frac{\text{sum}(1)}{\text{num}/10} \\
 \downarrow \\
 1
 \end{array}$$

① Add digits

$$\begin{array}{c}
 \text{num} = 38 \\
 \xrightarrow{\text{Assignment}} \\
 3 + 8 = 11 \\
 \xrightarrow{\text{Return the answer at } 2.}
 \end{array}$$

## Alternating Digit Sum

Question 2

num = 15

Alternating sign manner

$$15 - 14 + 13 - 12 + 11 - 10 + 9 - 8 + 7 - 6 + 5 - 4 + 3 - 2 + 1$$

Recursion

{ Even  $\rightarrow$  -ve  
 Odd  $\rightarrow$  +ve

Mentor

100%

Students

60-70%

Self study  
+

Issue

the class

Extra Efforts



100%

10-20%

Monthly  $\rightarrow$  0

Haening

17th July