Recursion

A function calling itself.

- Merge Sort & Quick sort
- Tree / BST
- Heaps & PQ
- Seg trees
- Backtracking
- Dynamic programming
- Graphs

Sum(N) ⟶ 1 + 2 + 3 + 4 + ---- + (N-1) + N

Sum(N) = Sum(N-1) + N

Returns sum of first N natural no.

$\sum_{i=1}^{N} i$

3 step process of writing recursive code

Step I Assumption

Decide what the fn does. And assume it works correctly.

Step II Main Logic

Solve the bigger problem using ans of subproblems.

Step III Base Condition

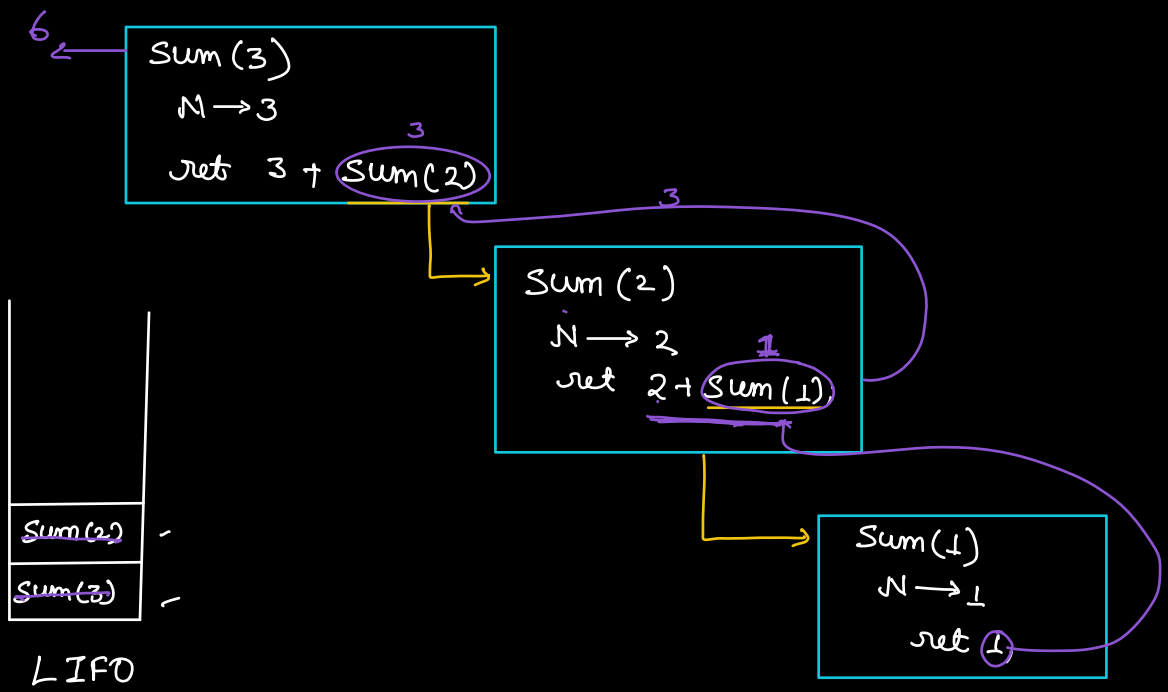
When recursion should stop

or

When main logic should not run

```
int Sum(N) {
// Assumption:
/* Sum(N) ⟶ The correct sum of 1st N natural no.
*/
→ if (N == 1)
      ret 1;

// Main logic
ret Sum(N-1) + N;
}
```



fact(N)

$N! = 1 \times 2 \times 3 \ldots (N-1)! \times N$

$0! = 1$

```
int fact (N) {
    if (N==0)
        ret 1;        ] → O(1)
    ret fact(N-1) x N;   → O(1)
}
```

N = 2	fact(1)x2
N = 1	fact(0)x1
N = 0	fact(-1)x0

fib(N) → (N > 0)

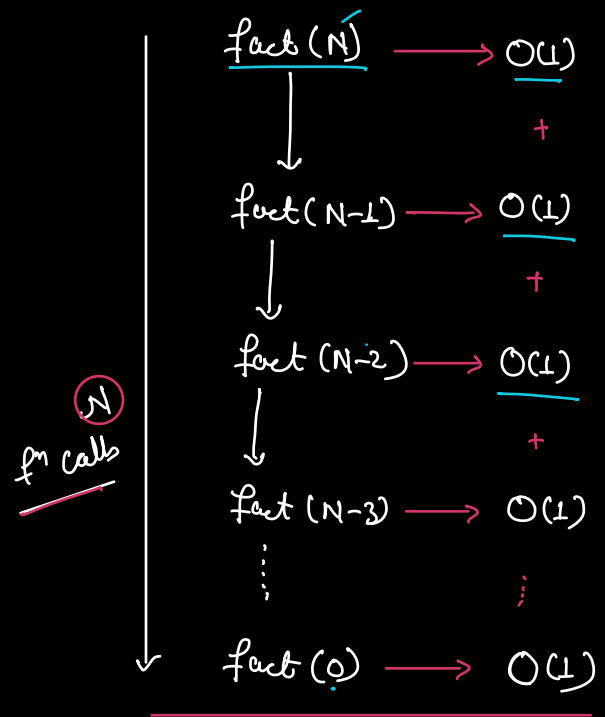
N :	1	2	3	4	5	6	7	8
fib(N);	1	1	2	3	5	8	13	21...

```
int fib (N) {
    if (N<=2) ret 1;   ] →
    ret fib(N-1) + fib(N-2);
}
```

N=3 → fib(2) + fib(1) ✓

✓ N=2 → fib(1) + fib(0) ✗

✓ N=1 → fib(0) + fib(-1)

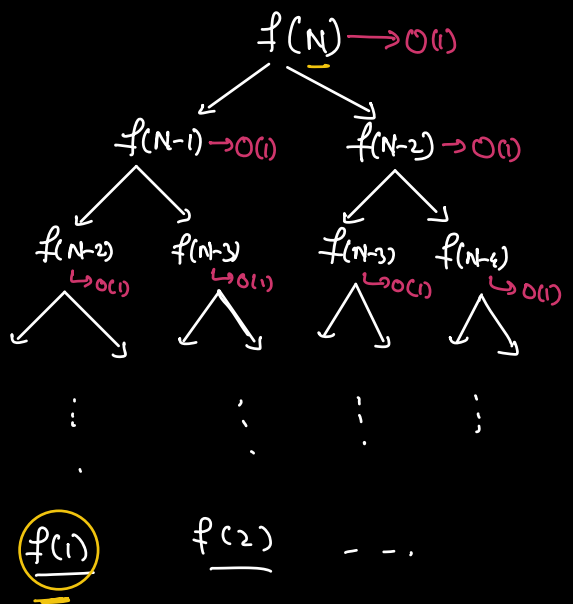


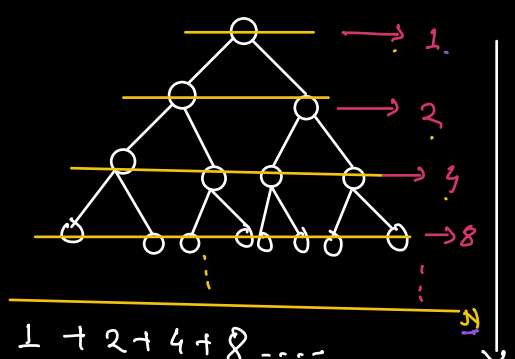
TC : N × O(1)

⇒ O(N)

SC : O(N)

fib → f





$1 + 2 + 4 + 8 \ldots$

$2^0 + 2^1 + 2^2 + 2^3 \ldots \ldots 2^{N-1}$

TC : $O(2^N)$

SC : O(N)

TC of recursive function = Total no of fn calls in recursive tree × TC of the fn (excluding the recursive calls)

Substitution Method

```
Sum(N)  =  Sum(N-1) + N ;
fact(N) =  fact(N-1) * N;
 T(N)       T(N-1)
```

$$T(N) = T(N-1) + T(1)$$

$T(N-1) = T(N-2) + T(1)$;

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

$T(N-2) = T(N-3) + T(1)$;

$$T(N) = T(N-3) + 3T(1)$$

⋮ ⋮

After K steps

$$T(N) = T(N-K) + K\,T(1)$$

$N-K = 0$ (if K is last step)

$N-K = 0$

$K = N$

$$T(N) = T(0) + N\,T(1)$$

$$T(N) = O(1) + N\,O(1)$$

$$T(N) = O(N)$$

Hw

$$T(N) = T(N-1) + T(N-2) + O(1)$$

Break till 10.30 p

Google

Q Given a, N & P

implement $pow(a, N, P) \longrightarrow a^N \% P$ $\{ a > 0$

(Do it recursively)

0^0

$a^N \longrightarrow a \times a^{N-1}$ =

$\hookrightarrow a^{N/2} \times a^{N/2}$

// $a^N = a \times a^{N-1}$

SC: O(N)

```
int pow(a, N, p) {
    if (N==0 || a==1) ret 1;
    ret (a%p x pow(a, N-1, p)) % p;
}
```

$(a^N) \% P = (a \times a^{N-1}) \% P$

$= (a\%P \times a^{N-1} \% P) \% P$

↳ $T(N) = T(1) + T(N-1)$

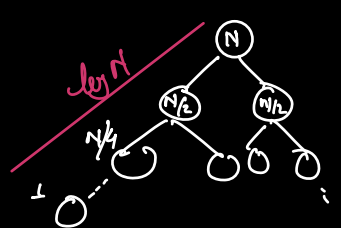
// $a^N = a^{N/2} \times a^{N/2}$

```
int pow(a, N, p) {
    if (N==0 || a==1) ret 1;
    if (N%2==0)
        ret (pow(a, N/2, P) x pow(a, N/2, P)) % P;
    else
        ret ((pow(a, N/2, P) x pow(a, N/2, P)) % P x a % P) % P
}
```

$N = 8$ $a^8 = a^4 \times a^4$

$N = 9$ $a^9 = (a^4 \times a^4) \times a$

SC: O(log N)



$T(N) = T(1) + 2\,T(N/2)$

$$T(N) = T(N-1) + T(1)$$

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

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

$$T(N-2) = T(N-3) + T(1)$$

$$T(N) = T(N-3) + 3T(1)$$

$$\vdots$$

$$T(N) = T(N-K) + KT(1)$$

$$N - K \to 0$$

$$T(N) = O(N)$$

$$T(N) = 2T(N/2) + T(1)$$

$$T(N/2) = 2T(N/4) + T(1)$$

$$T(N) = 2\left[2T(N/4) + T(1)\right] + T(1)$$

$$= 4T(N/4) + 3T(1)$$

$$T(N/4) = 2T(N/8) + T(1)$$

$$T(N) = 4\left[2T(N/8) + T(1)\right] + 3T(1)$$

$$= 8T(N/8) + 7T(1)$$

$$T(N/8) = 2T(N/16) + T(1)$$

$$T(N) = 16T(N/16) + 15T(1)$$

$$\vdots$$

$$T(N) = 2^K T(N/2^K) + (2^K - 1)T(1)$$

$$\frac{N}{2^K} = 1$$

$$N = 2^K$$

$$\Rightarrow \boxed{K = \log_2 N}$$

$$T(N) = 2^{\log_2 N} T(1) + (2^{\log_2 N} - 1)T(1)$$

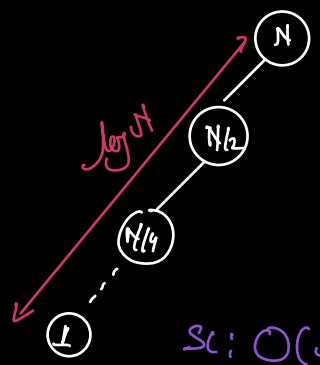
$$= N\,T(1) + (N-1)\,T(1)$$

$$= T(1)(2N-1)$$

$$T(N) = O(N)$$

// $a^N = a^{N/2} \times a^{N/2}$

```
int pow(a, N, P) {
    if (N == 0 || a == 1) ret 1;
    halfPow = pow(a, N/2, P);
    if (N % 2 == 0)
        ret (halfPow x halfPow) % P;
    else
        ret ((halfPow x halfPow) % P x a % P) % P
}
```



SC: O(log N)

$$T(N) = T(N/2) + T(1)$$

$$T(N/2) = T(N/4) + T(1)$$

$$T(N) = T(N/4) + 2T(1)$$

$$T(N/4) = T(N/8) + T(1)$$

$$T(N) = T(N/8) + 3T(1)$$

$$T(N/8) = T(N/16) + T(1)$$

$$T(N) = T(N/16) + 4T(1)$$

$$\vdots$$

$$T(N) = T(N/2^K) + KT(1)$$

$$\frac{N}{2^k} = 1 \qquad T(N/2^k) \longrightarrow T(1)$$

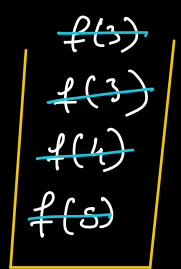
$$k = \log_2 N$$

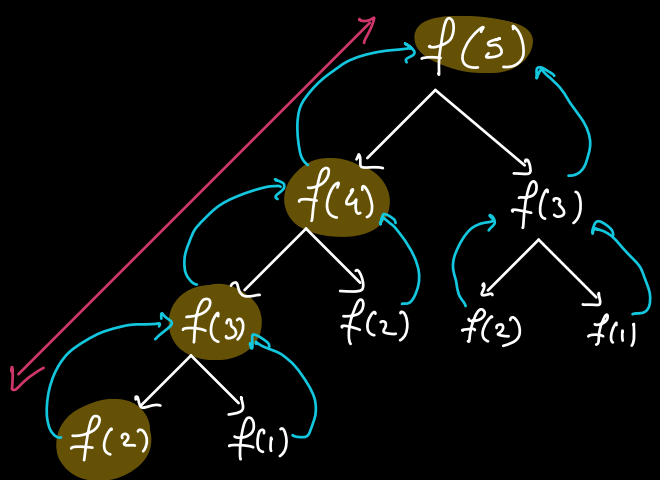
$$T(N) = T(1) + \log_2 N \; T(1)$$

$$T(N) = O(\log N)$$

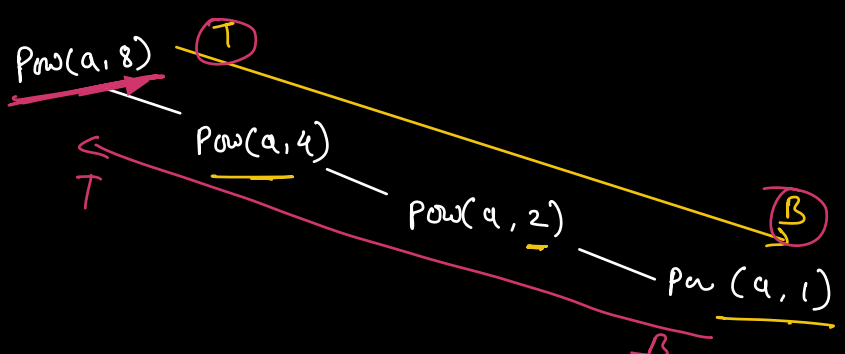
Space Complexity

3





SC of Recursive code = Max height of the rec tree at point of time



$a^1 \rightarrow a$

$a^2 \rightarrow a^1 \times a^1$

$a^4 \rightarrow a^2 \times a^2$

$a^8 \rightarrow a^4 \times a^4$