Recursion :

- Recursion Basics
- sum( ) / fact( ) / fib( ) /
- 3 Steps of Recursion & Recursive Flow
- Pow( ) / gcd( )
- Recursive Relations
- Master's Theorem

Recursion Basics

- Function calling itself
- Solving a problem using smaller instance of same problem
  - subproblem

$sum(N) = 1 + 2 + 3 + .. + N-1 + N$

$sum(N-1) + N$

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

- $sum(N)$: Sum of N Natural Numbers
- $sum(N-1)$: Smaller instance of same problem.

Recursion: Solving a problem using subproblem.

Recursive Steps:

1) Assumption : Decide what your function does:
2) Main Logic : Solving assumption using subproblems
3) Base Condition : When to stop

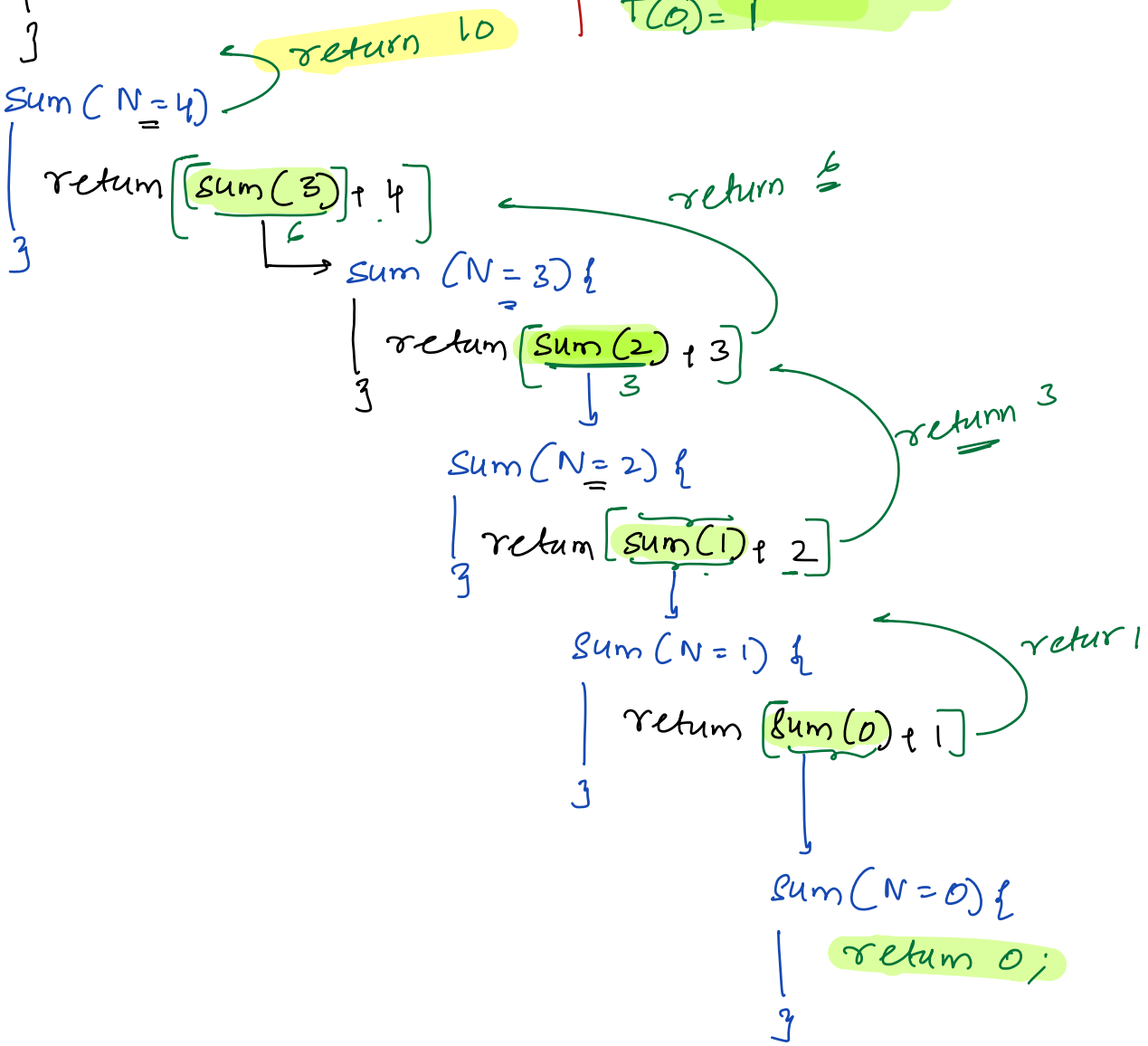
Ass: Calculate & return Sum of N N atural numbers.

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

Recursive

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

$T(0) = 1$



Ass: Calculate & return product of N Natural numbers.

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

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

$T(0) = 1$

// fibanacci Series

N:	0	1	2	3	4	5	6	7	8	9	N-2	N-1	N
= :	0	1	1	2	3	5	8	13	21	34			

fib(N-2) fib(N-1)

Return N^{th} fib Number.

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

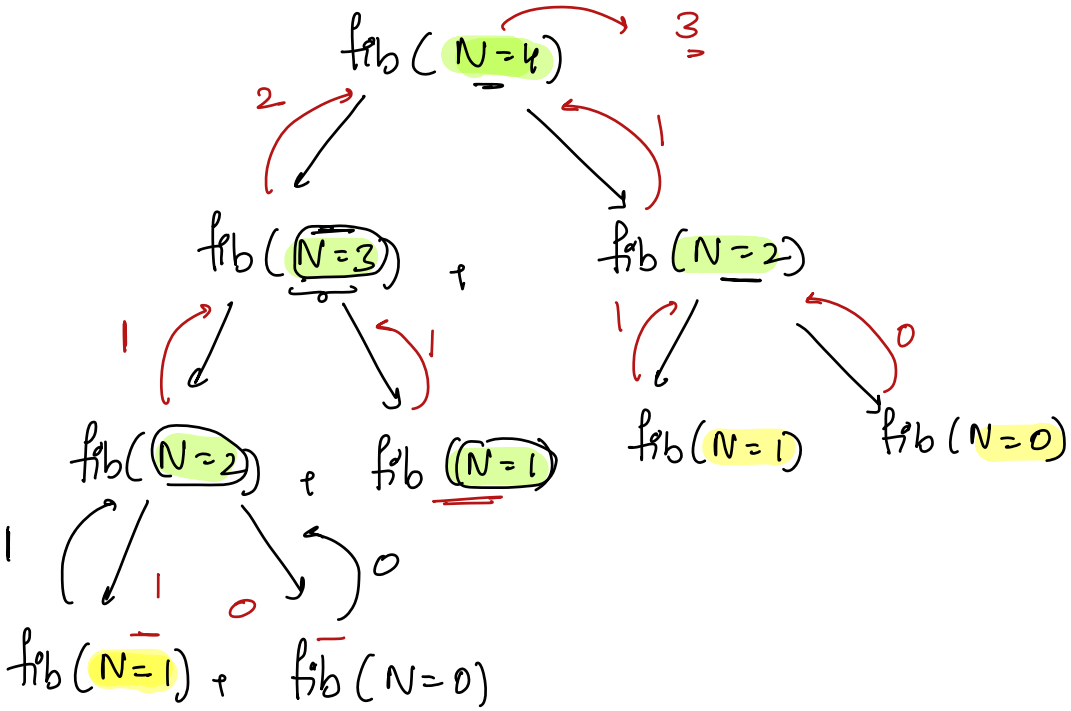
fib(0): fib(-1) + fib(-2) ✗

fib(1): fib(0) + fib(-1) ✗

fib(2): fib(1) + fib(0) ✓

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

$T(0) = 1, \quad T(1) = 1$


fib(N=4)
3
2
1
fib(N=3)
+
fib(N=2)
1
1
1
0
fib(N=2)
+
fib(N=1)
fib(N=1)
fib(N=0)
1
1
0
0
fib(N=1)
+
fib(N=0)

Power func

$a^{10} = a^9 * a$ } $a^7 = a \times a^6$ }

→ Calculate & return a^N

```
int pow(int a, int N)
{
    if(N==0) return 1;
    return pow(a, N-1) * a;
}
```

a^{N-1}

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

$T(0) = 1$

$\Rightarrow a^{10} = a^5 * a^5$

$a^{12} = a^6 * a^6$

$a^{15} = a * a^7 * a^7$

$a^{11} = a * a^5 * a^5$

$15/2 = 7$

Integer division

```
pow2(a, N) {
    if N==0 return 1
    if N is even {
        return pow(a, N/2) * pow(a, N/2)
    }
    if N is odd {
        return a * (pow(a, N/2) * pow(a, N/2))
    }
}
```

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

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

$T(0) = 1, \quad T(1) = 1$

```
pow(a, N) {
    if N == 0 return 1
    p = pow(a, N/2)
    if N is even {
        return {p * p}
    }
    if N is odd {
        return a * p * p
    }
}
```

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

$T(0) = 1, \quad T(1) = O(1)$

$T(N) = T(N/2) + O(1)$ →

// GCD

Infinite loop.

Ass: Calculate & return gcd a & b

```
gcd(a, b) {
    if (a == 0) return b
    if (b == 0) return a
    return gcd(a, b % a)
}
```

a > b assu

$gcd(a, b) = gcd(a, b \% a)$

small, bigger

gcd(10, 3)
↳ gcd(10, 3)
↳ gcd(10, 3)
↳ gcd(10, 3)

gcd(10, 17) → gcd(10, 7) → gcd(10, 7) → gcd(10, 7)

```
gcd (a, b) {
    if (a == 0) return b
    if (b == 0) return a
    gcd (b % a, a)
}
```

1st = 2nd =

We won't be needing this 2nd Base condition

a b

gcd (20, 7)

↳ gcd (7, 20)

gcd (20 % 7, 7)

= gcd (6, 7)

gcd (7 % 6, 6)

gcd (1, 6)

gcd (6 % 1, 1)

gcd (0, 1)

Recursive Relations

1) Sum : $T(N) = T(N-1) + 1$, $T(0) = 1$ } TC: $O(N)$

2) Fact : $T(N) = 1 + T(N-1)$, $T(0) = 1$ } TC: $O(N)$

3) Fib : $T(N) = 1 + T(N-1) + T(N-2)$, $T(0) = 1$, $T(1) = 1$ } TC:

4) Pow :

1) $T(N) = T(N-1) + 1$, $T(0) = 1$ } TC: $O(N)$

2) $T(N) = 2T(N/2) + O(1)$, $T(0) = 1$, $T(1) = 1$ } TC: $O(N)$

3) $T(N) = T(N/2) + O(1)$, $T(0) = 1$, $T(1) = O(1)$ } TC: $O(\log_2 N)$

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

$N-2$

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

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

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

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

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

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

$T(N) = T(N-4) + 4$

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

if $k = N$

$T(N) = T(0) + N$

$= 1 + N$

$T(N) = O(N)$

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

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

$T(N) = T(N/4) + 2 \rightarrow T(N/2^2) + 2$

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

$T(N) = T(N/8) + 3 \rightarrow T(N/2^3) + 3$

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

$T(N) = T(N/16) + 4 \rightarrow T(N/2^4) + 4$

$T(N) = T(N/2^k) + k$

$T(0) = 1$

$T(1) = 1$

$\log_2^n 2 = n$

$N/2^k = 1$

$2^k = N$

Apply $\log_2$ on both sides

$k = \log_2^N$

$T(N) = T(N/2^{\log_2^N}) + \log_2^N \rightarrow T(N/N) + \log_2^N$

$= 1 + \log_2^N$

$T(N) = O(\log_2^N)$

$T(N) = 2[T(N/2)] + 1$, $T(0) = 1$, $T(1) = 1$

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

$= 2 * [2T(N/4) + 1] + 1$

$= 4[T(N/4)] + 2^2 - 1 \rightarrow 2^2 T(N/2^2) + 2^2 - 1$

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

$= 4 * [2T(N/8) + 1] + 3$

$= 8T(N/8) + 2^3 - 1 \rightarrow 2^3 T(N/2^3) + 2^3 - 1$

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

$= 8 * [2T(N/16) + 1] + 7$

$= 16T(N/16) + 15 \rightarrow 2^4 T(N/2^4) + 2^4 - 1$

$T(N) = 2^k T(N/2^k) + 2^k - 1$ $\quad k = \log_2 N$

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

$= N * T(1) + N - 1 \rightarrow 2N \rightarrow O(N)$

TODO:

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

→ Merge Sort Recursive Relation.

TODO: $T(N) = 2T(N-1) + O(1)$ $\qquad T(1) = 1$

Fib: $T(N) = T(N-1) + T(N-2) + 1$ } $T(N) > T(N-1)$

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

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

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

```
pow3(a, N) {
    if N == 0 return 1
    p = pow(a, N/2)
    if N is even {
        return {p * p}
    }
    if N is odd {
        return a * p * p
    }
}
```

```
pow3(a, N=10)
    p = pow3(a, 5)      = a^5
    return p*p
```

```
pow3(a, N=5)
    p = pow3(a, 2)      : a^2
    return p*p*a :
```

```
pow3(a, N=2)
    p = pow3(a, 1)
    retu p*p
```

```
pow3(a, N=1)
    p = pow3(a, 0)
    p = 1.
    return
    p*p*a
```