Sum of N numbers :—

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

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

$$= T(N-2) + 2 \quad \text{..............} \quad 2^{nd}$$

$$= T(N-3) + 3 \quad \text{..............} \quad 3^{rd}$$

$$= T(N-4) + 4 \quad \text{..............} \quad 4^{th}$$

$$\vdots$$

$$T(N) = T(N-K) + K \quad \text{..............} \quad K^{th}$$

$$\therefore \underline{T(N-K) = T(1)} \quad (\text{base case} \rightarrow N == 1)$$

$$N - K = 1$$

$$K = N - 1$$

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

$$T(N) = \underline{T(1)} + N - 1$$

$$= 1 + N - 1$$

$$T(N) = N \longrightarrow \underline{TC : O(N)}$$

Factorial of N :-

```
int factorial (int n) {
    if (n==0) return 1;
    return factorial(n-1) * n;
}
```

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

$$T(N) = T(N-1) + 1$$
$$= T(N-2) + 2 \quad \text{.................} \quad 2^{nd}$$
$$= T(N-3) + 3 \quad \text{.................} \quad 3^{rd}$$
$$= T(N-4) + 4 \quad \text{.................} \quad 4^{th}$$

$$T(N) = T(N-K) + K \quad \text{.................} \quad K^{th}$$

$\therefore \underline{T(N-K) = T(1)}$ (base case → N==0)

$$N-K = 1$$
$$K = N-1$$

$$T(N) = T(N-K) + K$$
$$T(N) = \underline{T(1)} + N$$
$$= 1 + N$$

$$T(N) = N \longrightarrow \underline{\underline{TC : O(N)}}$$

Calculate Power f^n :-

```
long power(int a, int n) {        a^n
    if (n == 0) return 1;

    long halfpower = power(a, n/2);

    if (n % 2 == 0) {
        return halfpower * halfpower;
    } else {
        return halfpower * halfpower * a;
    }
}
```

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

$= T(N/4) + 2$ .... 2^{nd}

$= T(N/8) + 3$ .... 3^{rd}

$= T(N/16) + 4$ .... 4^{th}

$T(N) = T(N/2^K) + K$ .... K^{th}

$T(N/2^K) = T(1)$ (base case → n==0)

$N/2^K = 1$

$2^K = N$

$K = \log_2(N)$

$$T(N) = T\left(\frac{N}{2^{\log N}}\right) + \log N$$

$$= T(N/N) + \log N$$

$$= T(1) + \log N$$

$$T(N) = 1 + \log N$$

$\longrightarrow$ TC: $O(\log N)$

Calculate TC of power f^n (V2 implementation)

```
long power(int a, int n) {
    if (n == 0) return 1;
    if (n%2 == 0) {
        return power(a, n/2) * power(a, n/2);
    } else {
        return power(a, n/2) * power(a, n/2) * a;
    }
}
```

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

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

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

$$T(N) = 4T(N/4) + 3 \rightarrow 2^2 - 1 \quad \text{............... } 2^{nd}$$

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

$$T(N) = 8T(N/8) + 7 \rightarrow 2^3 - 1 \quad \text{............... } 3^{rd}$$

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

$$T(N) = 16T(N/16) + 15 \rightarrow 2^4 - 1 \quad \text{............... } 4^{th}$$

$$T(N) = 2^K T\left(N/2^K\right) + \left(2^K - 1\right) \quad \text{------------} \quad K^{th}$$

$$T\left(N/2^K\right) = T(1)$$

$$\therefore N/2^K = 1$$

$$\therefore 2^K = N$$

$$\therefore K = \log_2 N$$

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

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

$$= N + N - 1$$

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

$$\longrightarrow TC: O(N)$$

Calculate TC of fibonacci :-

```
long fibonacci(int n) {
  if(n < 2) return n;

  return fibonacci(n-1)+fibonacci(n-2);

}
```

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

$T(N-1) \quad + \quad T(N-2) + 1$

$T(N-2) + T(N-3) + 1 \qquad T(N-3) + T(N-4) + 1$

$T(N-3) + T(N-4) + 1 \quad T(N-4) + T(N-5) + 1 \quad T(N-4) + T(N-5) + 1 \quad T(N-5) + T(N-6) + 1$

- ☆ This seems to very complex if we want to calculate in Mathematical expression. So, we will try to calculate the worst performer function's time complexity to get the $O(n)$.
- ☆ Here "$T(n-1)$" seems to be the slowest performer as it's value reduces very slowly.
- ☆ To calculate this we will assume to replace all f^n calls by the slowest f^n.

Substitution f^n	max. TC term	count	$O(1)$
1	$T(n-1)$	$2 \rightarrow 2^1$	1
2	$T(n-2)$	$4 \rightarrow 2^2$	3
3	$T(n-3)$	$8 \rightarrow 2^3$	7
⋮	⋮	⋮	.... K^{th}
K	$T(n-K)$	2^K	2^K-1

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