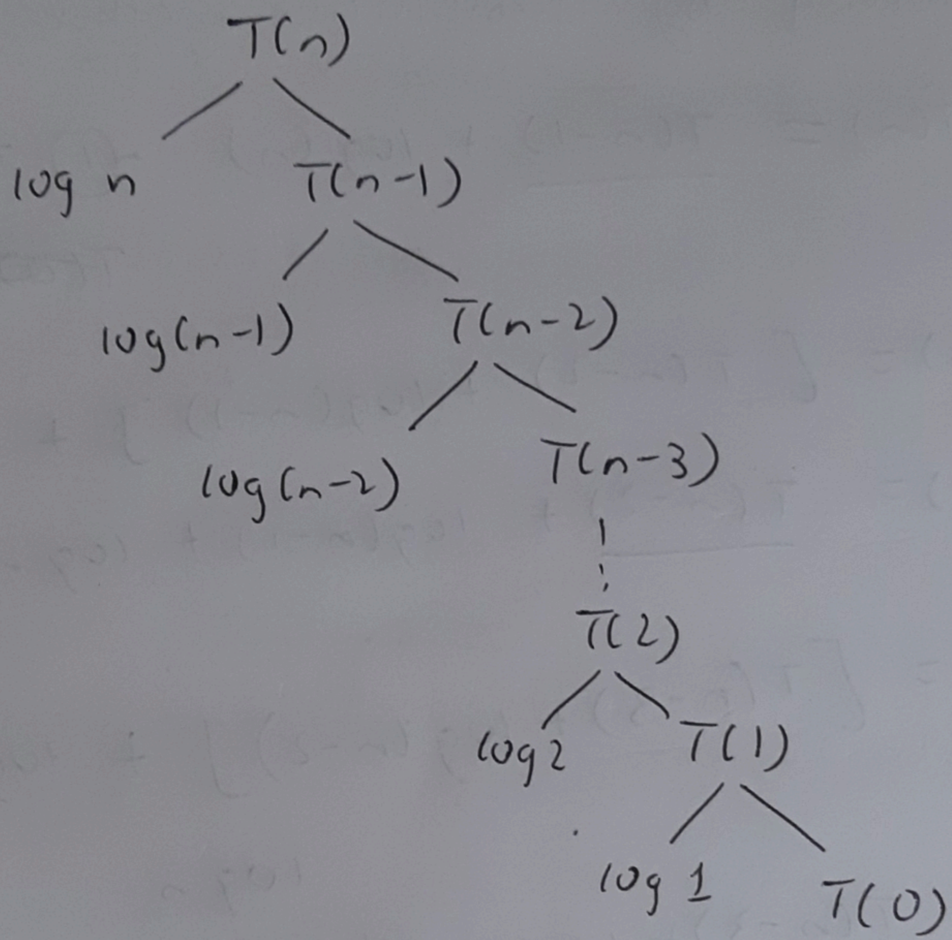


$$\therefore T(n) = \log n + \log(n-1) + \log(n-2) + \dots + \dots \log 2 + \log 1$$

$$= \log(n \times (n-1) \times 2 \times 1)$$

$$\therefore \boxed{T(n) = \log n!}$$

— For $\log n!$ we don't have Θ bound. Only upper bound.

$$= O(n \log n)$$

→ Solving by substitution method –

- $T(n) = \underline{T(n-1)} + \log(n)$ —(1)

$\underline{T(2) = 0}$

- $T(n) = [\, T(n-2) + \log(n-1) \,] + \log n$

$T(n) = \underline{T(n-2)} + \log(n-1) + \log n$ —(2)

- $T(n) = [\,T(n-3) + \log(n-2)\,] + \log(n-1) + \log n$

$T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log n$ —(3)

⋮

continue for k times. n is reducing to 2.

- $T(n) = T(n-k) + \log 1 + \log 2 + \dots + \dots \log(n-1) + \log n$

$\because \; n - k = 0$

$\therefore \; n = k$

$\therefore n = k$ Substitute $n = k$

$\therefore T(n) = T(0) + \log n!$

$T(n) = 1 + \log n!$

$= O(n \log n)$

Summerise –

- $T(n) = T(n-1) + 1$ — $O(n)$
- $T(n) = T(n-1) + n$ — $O(n^2)$
- $T(n) = T(n-1) + \log n$ — $O(n \log n)$

Similarly

- $T(n) = T(n-2) + 1$ — $O(n)$
- $T(n) = T(n-100) + n$ = $O(n^2)$
- $T(n) = T(n-1) + n^2$ ≈ – $O(n^3)$

These are similer type of recurrence relations

IV Recurrence Relation

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

```
void Test(int n)            —— T(n) time.
{
    if(n>0)
    {
      printf("%d", n);      —— 1
      Test(n-1);            —— T(n-1)
      Test(n-1);            —— T(n-1)
    }
}
```

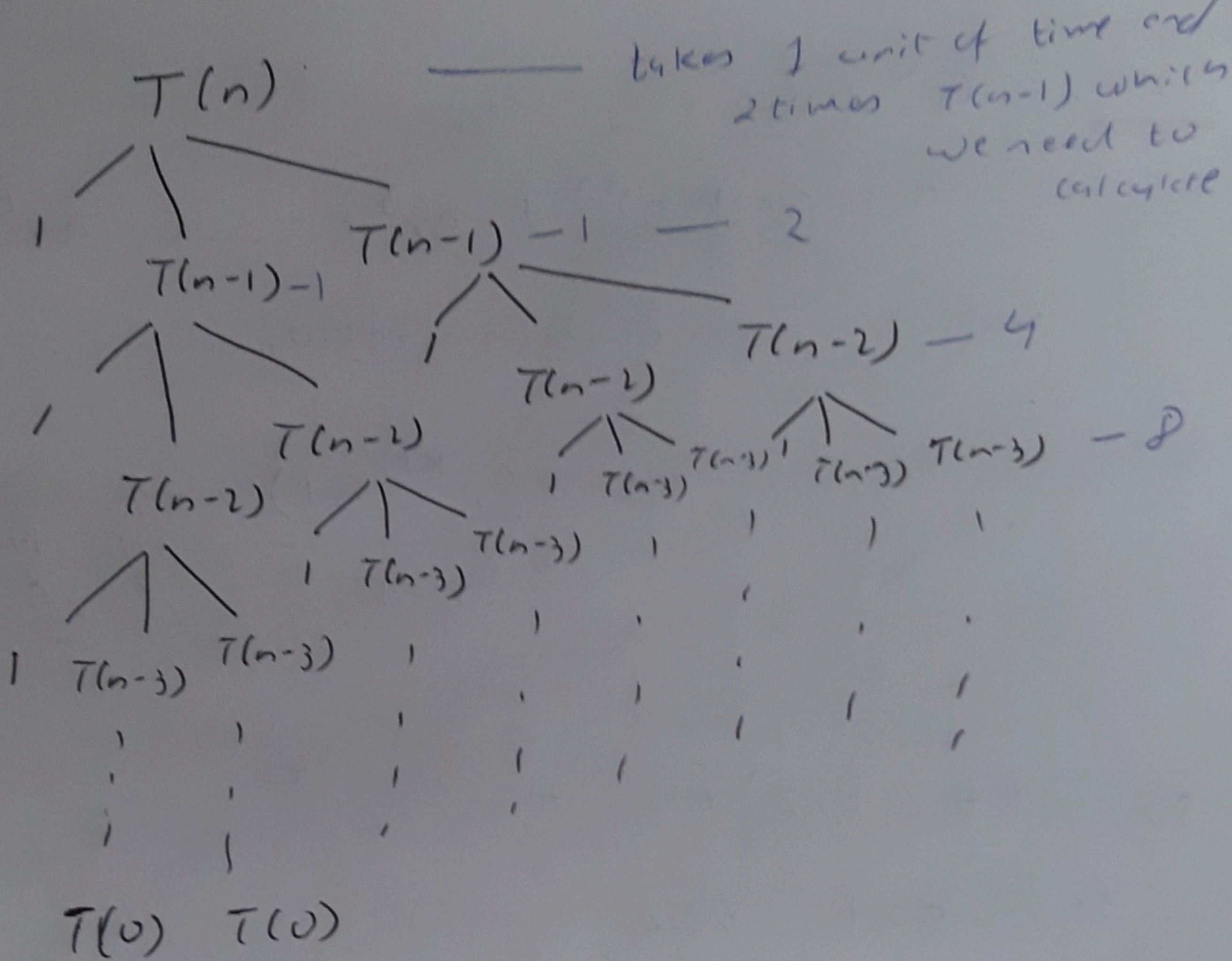
$$T(n) = 2T(n-1) + 1$$

∴ Recurrence relation is

$$T(n) = \begin{cases} 1 & n=0 \\ 2T(n-1)+1 & n>0 \end{cases}$$

Solving by recurrence tree method –

- The algorithm is printing n − 1 time and calling itself (n−1) times twice at every step.



- So each step takes 1 unit of time and twice $T(n-1)$.

So time = $1 + 2 + 4 + 8$ ---- K times.

Till K time $n = 0$. as we proceed further.

- Total. Time = $\underbrace{1 + 2 + 2^2 + 2^3 + 2^4 + \cdots 2^k}$

This is sum of Geometric progression series

- $= 2^{k+1} - 1$

Note: GP series –

$$a + ar + ar^2 + ar^3 + \cdots ar^k =$$

where r is common ratio,

$$= \frac{a(r^{k+1} - 1)}{r - 1}$$ } Ans formula

- So $1 + 2 + 2^2 + 2^3 + \cdots 2^k$

Here $a = 1$ $r = 2$

$$\therefore \frac{a(r^{k+1} - 1)}{r-1} = \frac{1(2^{k+1} - 1)}{2-1} = 2^{k+1} - 1$$

→ Assume $n - k = 0$

$\therefore n = k$

$$\therefore 2^{k+1} - 1 = 2^{n+1} - 1 = \boxed{O(2^n)}$$

(13)

→ Solving by back substitution method -

→ $$T(n) = \begin{cases} 1 & n = 0 \\ 2T(n-1)+1 & n > 0 \end{cases}$$

→ $T(n) = 2T(n-1) + 1$ — (1) — 1st substitute

- $T(n) = \cancel{2T(}\ 2[2T(n-2)+1]+1$

$= 2^2 T(n-2) + 2 + 1$ — (2)

- $T(n) = 2^2 T[2T(n-3)+1] + 2 + 1$ — 2nd substitute

$= 2^3 T(n-3) + 2^2 + 2 + 1$ — (3)

⋮ continue for k times

- $T(n) = 2^k T(n-k) + 2^{k-1} + 2^{k-2} + \ldots 2^2 + 2 + 1$ — (4)

- ~~T(n) =~~ Assume $n - k = 0$

$n = k$

$\therefore T(n) = 2^n T(0) + \underbrace{1 + 2 + 2^2 + \ldots 2^{k-1}}$

$= \cancel{2^n * 1 + 2^{n-1}}$

$= \cancel{2^n * 1 +}\ 2^n * 1 + 2^k - 1$

$= 2^n + 2^n - 1$

$$T(n) = 2^{n+1} - 1$$
$$= O(2^n)$$ solved.

Observations

$T(n) = T(n-1) + 1$ — $O(n)$

$T(n) = T(n-1) + n$ — $O(n^2)$

$T(n) = T(n-1) + \log n$ — $O(n \log n)$

$T(n) = 2T(n-1) + 1$ — $O(2^n)$

$T(n) = 3T(n-1) + 1$ — $O(3^n)$ — similarly.

$T(n) = 2T(n-1) + n$ — $O(n2^n)$ — similarly

→ whatever func. we are having on extreme right, it is multiplied by ~~a~~. first term n.

eg: in (1)

$1 . n = n$

$n . n = n^2$

$\log n . n = n \log n$

$1 . 2^n = 2^n$

$1 . 3^n = 3^n$

$n . 2^n = n2^n$

for n-1.