Analysis of Recursion

1. eg -
```
void fun (int n)
{
    if (n<=0)
        return;
    print ("GFG");
    fun (n/2);
    fun (n/2);
}
```

Here we will write the recursive relations

when,
- $n > 0$

$T(n) = T(n/2) + T(n/2) + \Theta(1)$

$= 2T(n/2) + \Theta(1)$

$n <= 0$

~~T(n)(0~~ $T(n) = \Theta(1)$

recurrence relation.

2.
```
void fun(int n)
{
    if(n<=0)
        return;
    for(i=0; i<=n; i++)
        print ("GFG");
    fun (n/2);
    fun (n/3);
}
```

$\Theta(n)$

$n > 0$

$T(n) = T(n/2) + T(n/3) + \Theta(n)$

$n <= 0$

$T(0) = \Theta(1)$

recurrence relation

```
3. void fun(int n)
{
    if (n<=1)
        return;
    print("GFG");
    fun(n-1);
}
```

when $n > 1$

$T(n) = T(n-1) + \Theta(1)$

$n <= 1$

$T(1) = \Theta(1)$

} recurrence relation

We use Recursion Tree Method for solving recurrences.

$$T(n) = \underbrace{2T(n/2)}_{\text{Recursive}} + \underbrace{Cn}_{\text{Non recursive}}$$

$T(1) = C$

We will consider a tree and compute total work done.

Cn → consider non recursive as the root of the tree

$T(n/2)$ $T(n/2)$

Cn — Cn

$Cn/2$ $Cn/2$ — Cn

$T(n/4)$ $T(n/4)$ $T(n/4)$ $T(n/4)$

$c(n/4)$ $c(n/4)$ $c(n/4)$ $c(n/4)$ — Cn

c ... c c ... c c ... c c ... c — Cn

P.T.O.

$Cn + Cn + Cn \cdots Cn$

$\Theta(\log n)$

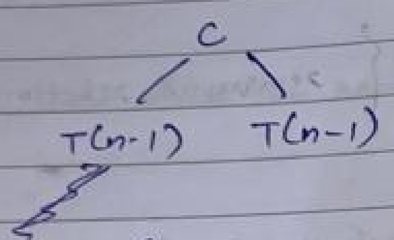
Happening for n times so,

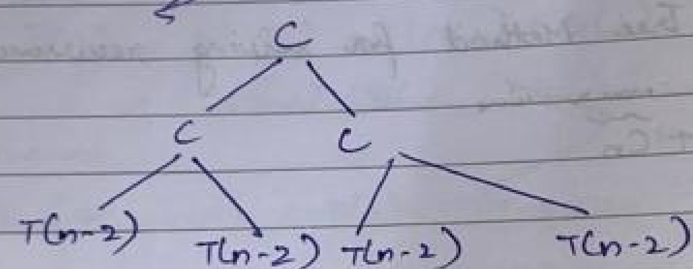
$\boxed{\Theta(n \log n)}$

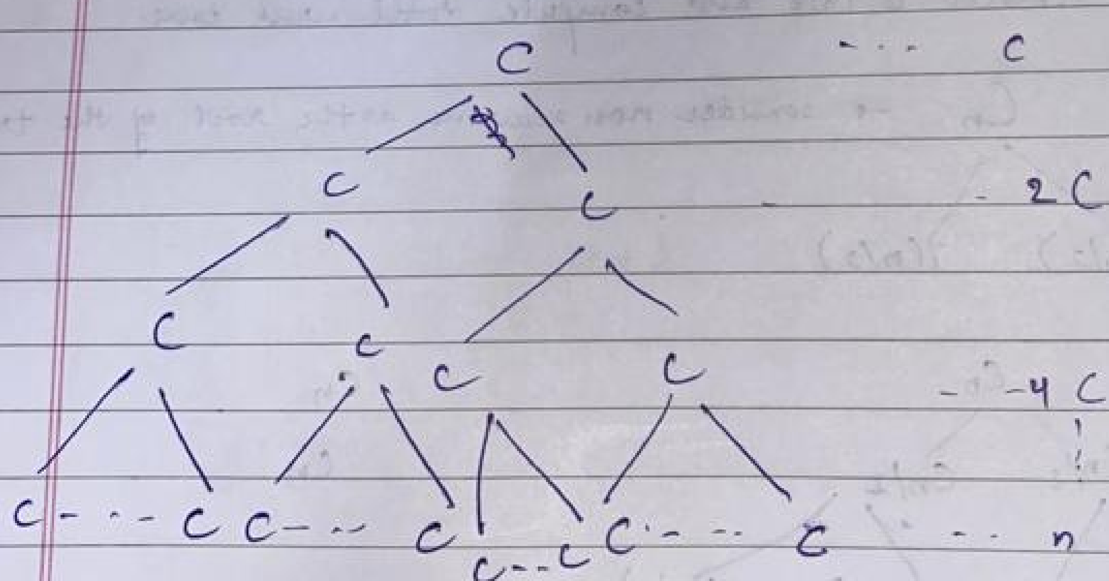
(Q) Recurrences –

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

$T(1) = C$







$- - - (C + 2C + 4C - - -)$

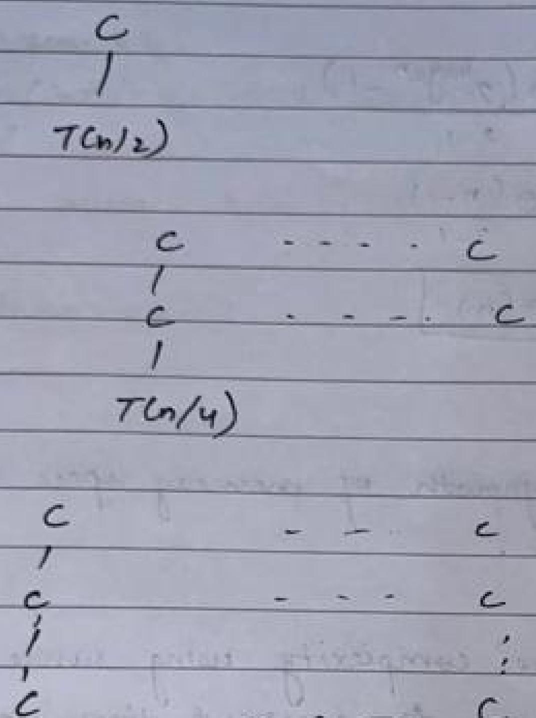
$G.P = n \quad C(1 + 2 + 4 \ldots)$

$C\left(\frac{1 \times (2^n - 1)}{2 - 1}\right)$

$\Theta(2^n)$?.?

(Q) $T(n) = T(n/2) + c$

$T(1) = c$

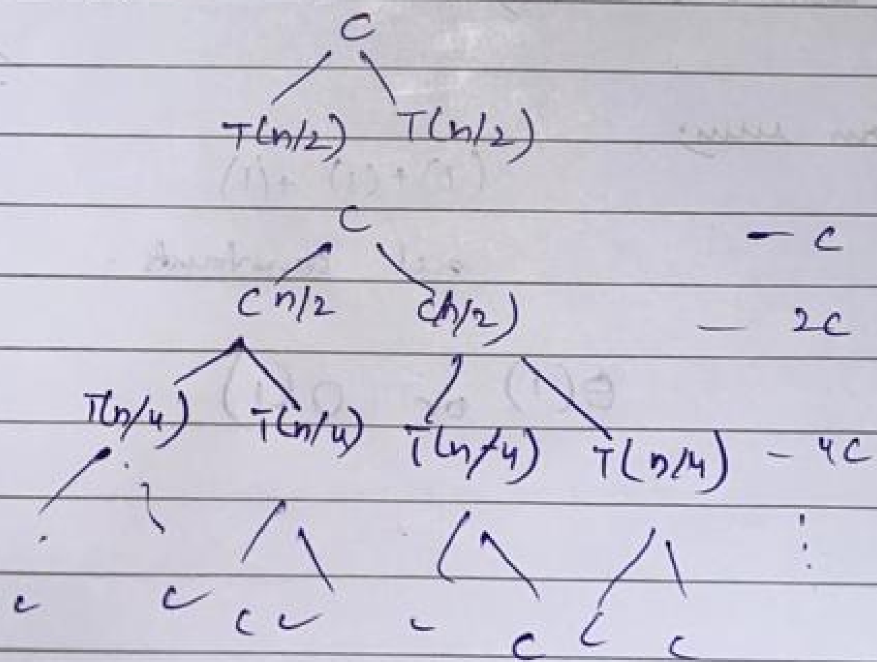


$$\underbrace{c + c + c \cdots c}_{\lceil \log_2 n \rceil + 1}$$

$$= \boxed{\theta(\log n)}$$

(Q) $T(n) = 2T(n/2) + c$

$T(1) = c$



$\underline{c + 2c + 4c} \ldots$

$\Theta(\log_2 n)$

$\cancel{\Theta(2\log}$ $\dfrac{\Theta(2^{\log_2 n} - 1)}{2-1}$

$\dfrac{\Theta(n-1)}{2-1}$

$\boxed{\Theta(n)}$