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,

- $n>0$

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

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

$n<=0$

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

→ 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 $n>0$ 1

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

$n<=1$

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

} recurrence relation

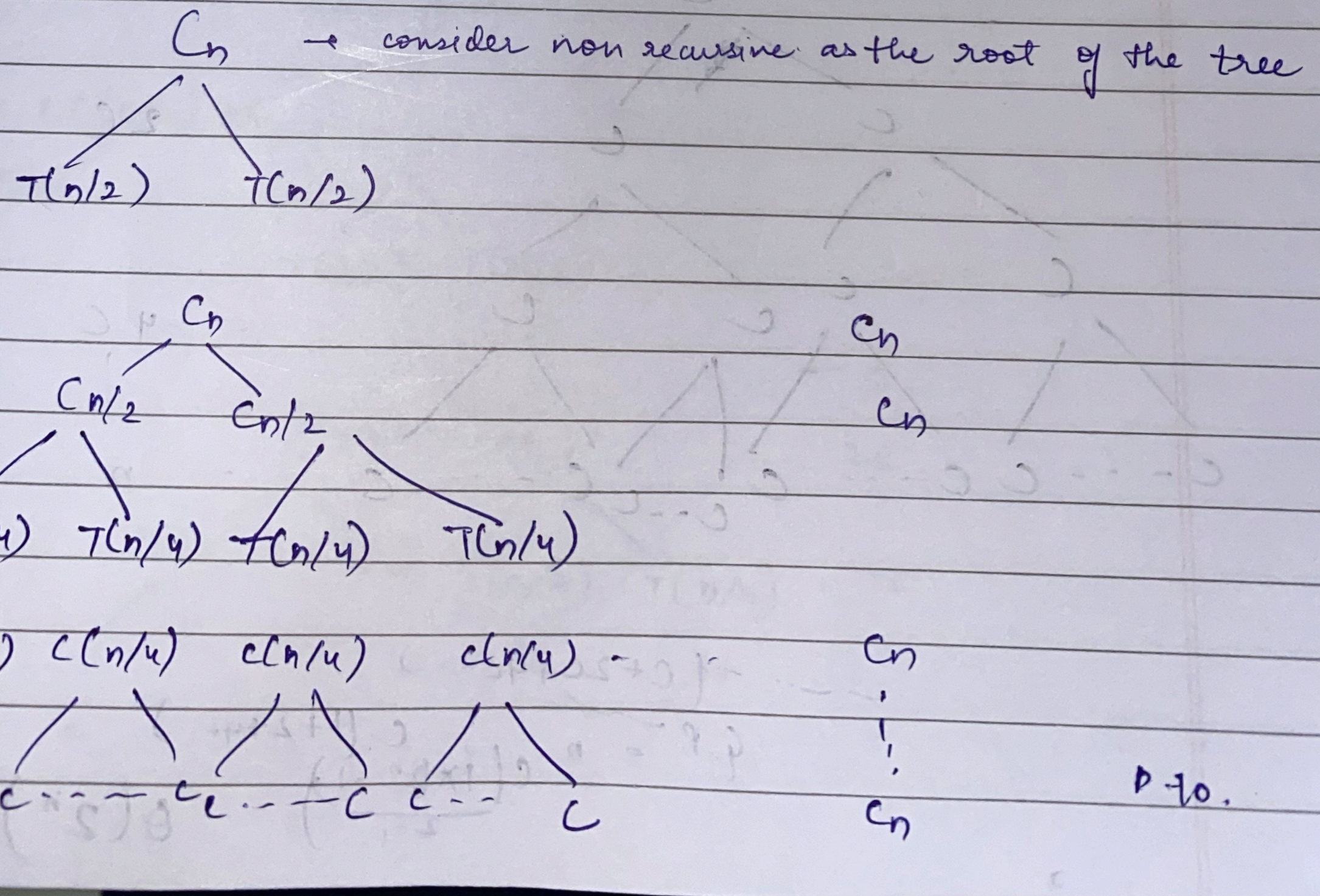
We use Recursion Tree Method for solving recursences.

Recursive Non recursive.

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

$T(1) = C$

we will consider a tree and compute total work done.



P.T.O.

(Q)

$cn + cn + cn \ldots cn$

$\theta(\log n)$

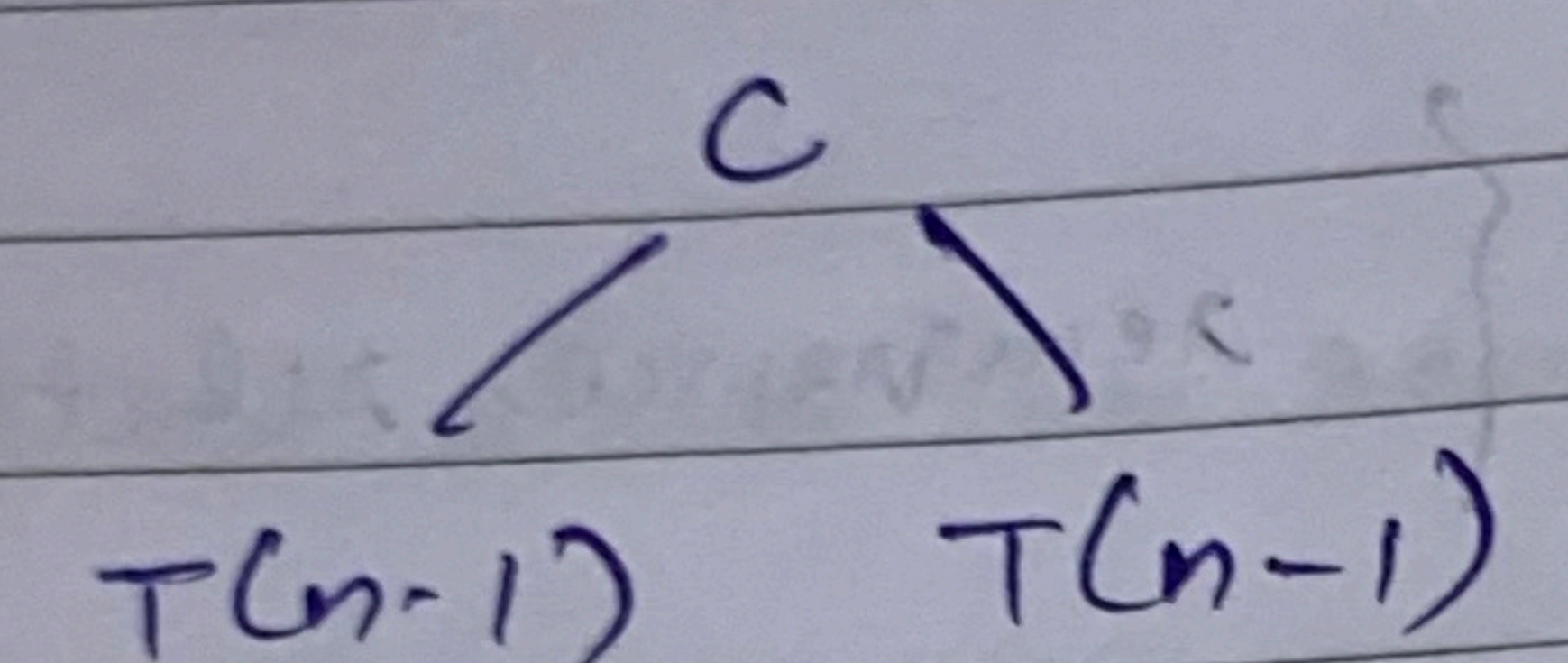
Happening for n times so,

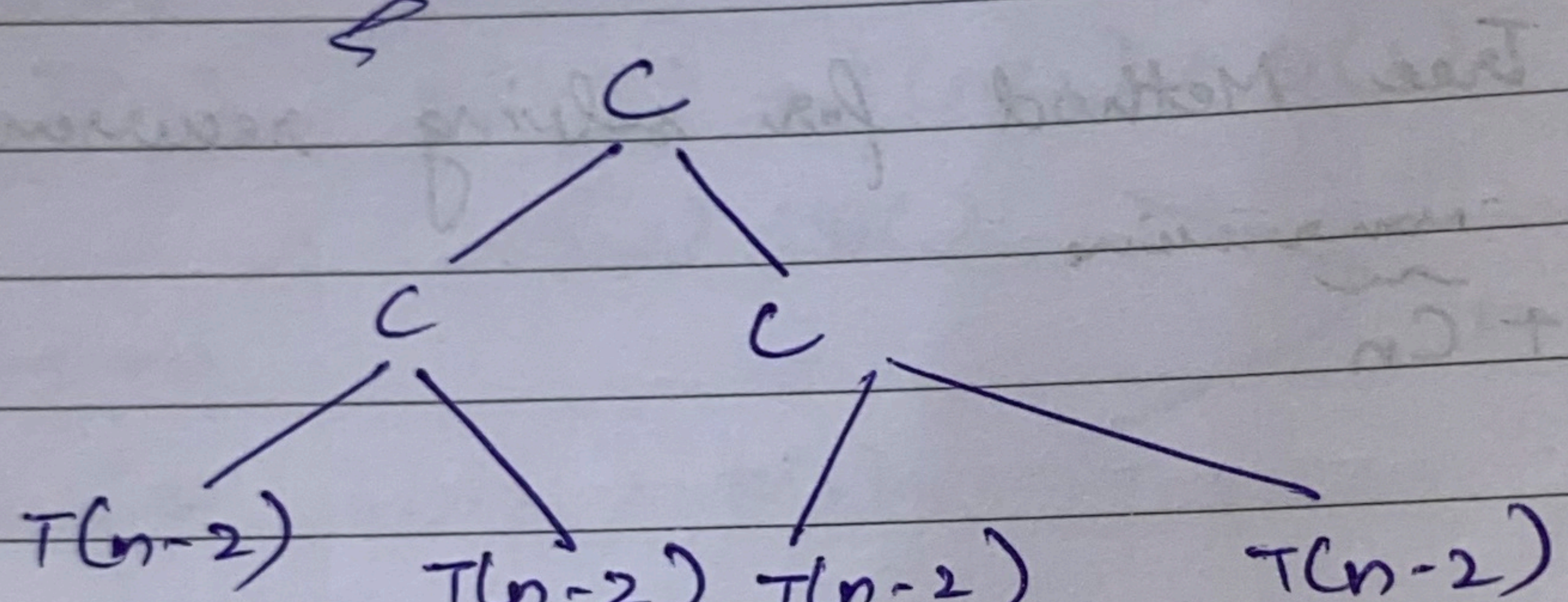
$\boxed{\theta(n \log n)}$

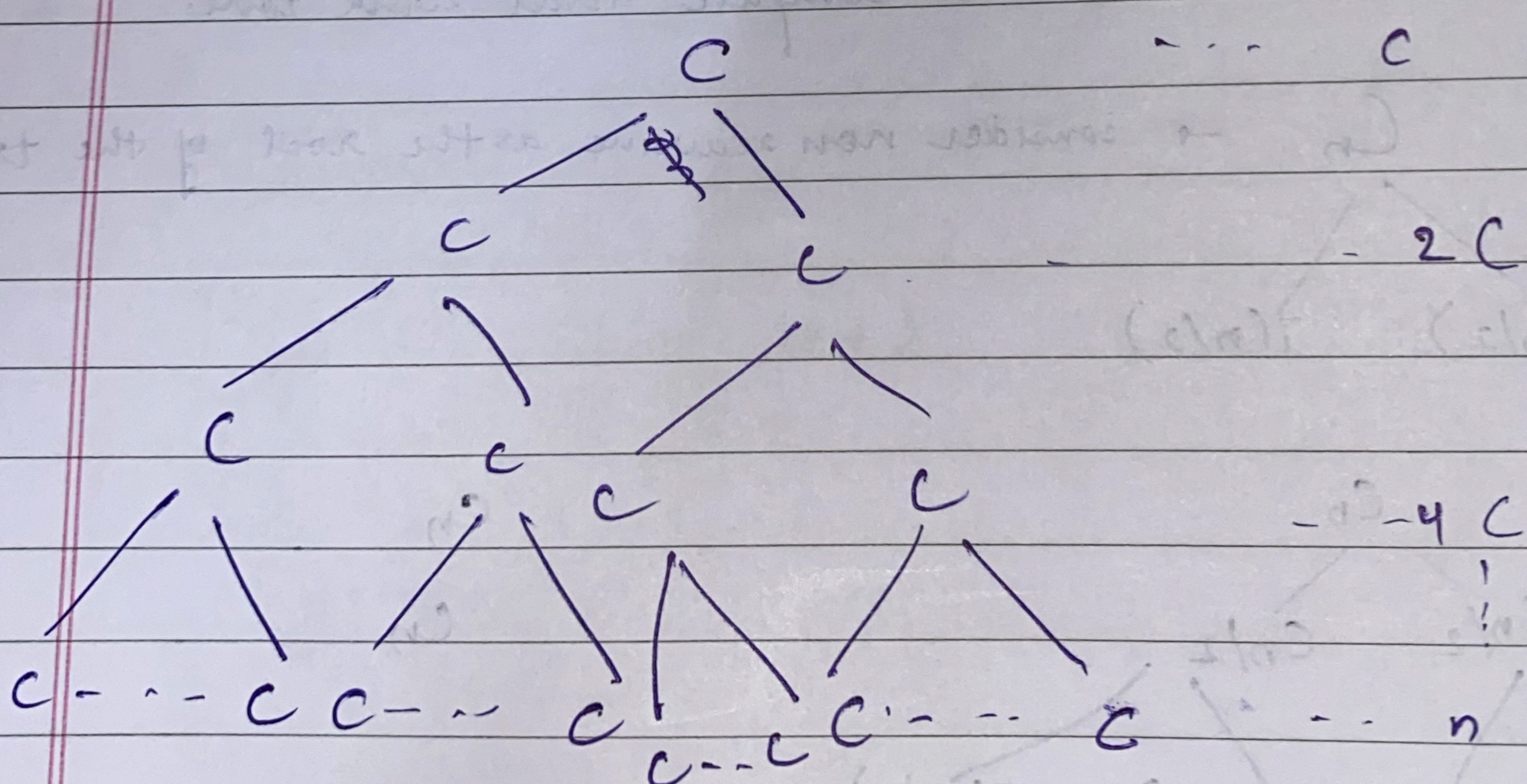
(Q) Recurrences –

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

$T(1) = C$







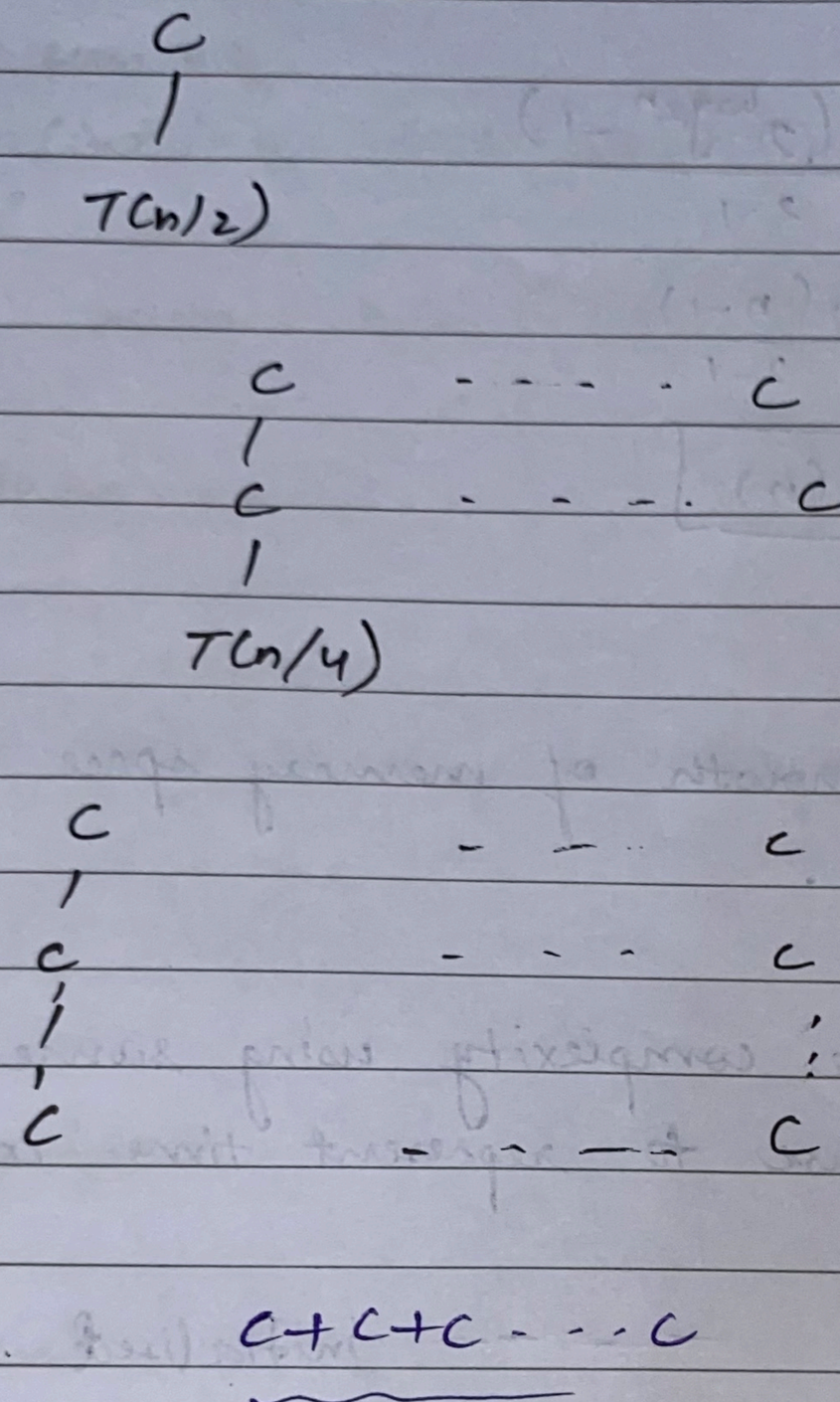
$\{c + 2c + 4c \ldots\}$

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

$c\left(\frac{1 \times (2^n - 1)}{2-1}\right) \quad \theta(2^n)$??

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

$T(1) = c$



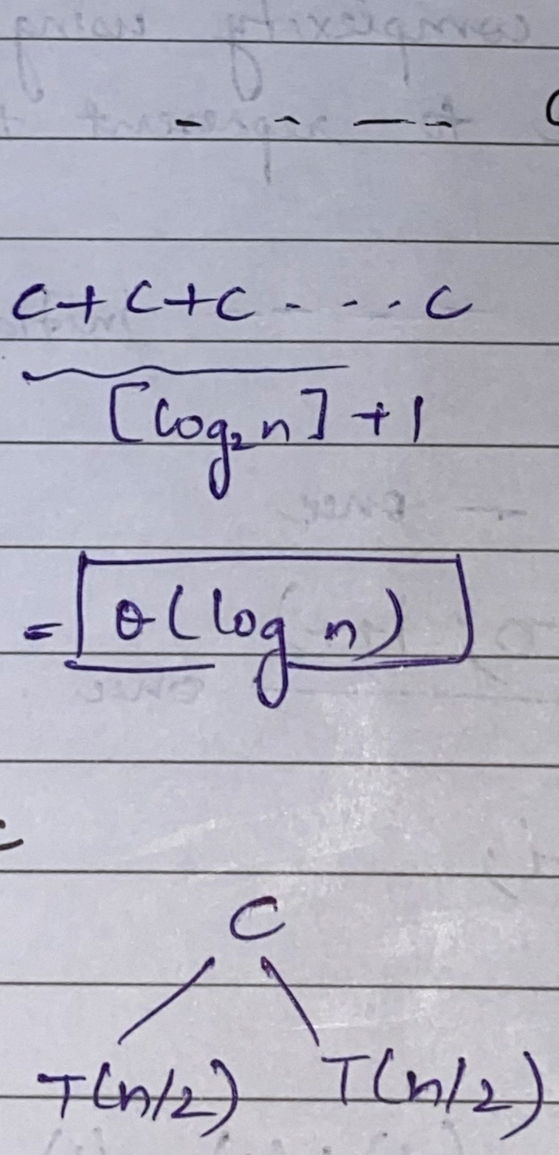
$c + c + c \cdots c$

$\underbrace{\qquad\qquad}_{[\log_2 n] + 1}$

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

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

$T(1) = c$



$c + 2c + 4c - \cdots$

$\Theta(\log_2 n)$

~~$\Theta(2^{\log}$~~ $\frac{\Theta(2^{\log_2 n} - 1)}{2-1}$

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

$\neq \Theta(n)$