Recursion Analysis

1) To solve a recursion complexity we first write a recurrence relation.

Eg:-

a) ① Void func(int n) $\longrightarrow$ input n $\rightarrow T(n)$

```
{
    if(n>=0)
      return;
    print("Red");
    func(n/2);
    func(n/2);
}
```

if(n>=0) return; print("Red"); } constant work $\rightarrow \theta(1)$

func(n/2); $\longrightarrow T(n/2)$

func(n/2); $\longrightarrow T(n/2)$

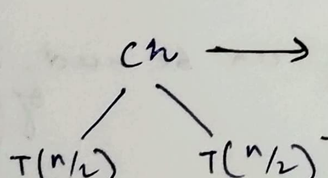
$\therefore$ when $n>0 \Rightarrow T(n) = T(n/2) + T(n/2) + \theta(1) \Rightarrow 2T(n/2) + \theta(1)$

$n<0 \Rightarrow T(n) = \theta(1)$ $\qquad n=0 \Rightarrow T(0) = \theta(1)$

2) Solving recurrence relations

a) Recursion tree method $\Rightarrow$ For ① $[T(n) = 2T(n/2) + cn]$

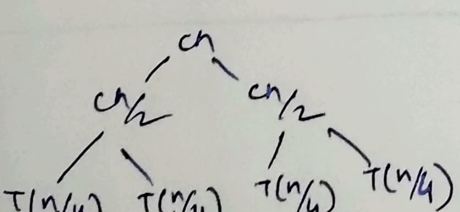
$\Rightarrow$



cn $\longrightarrow$ write the non-recursive as root of the tree.

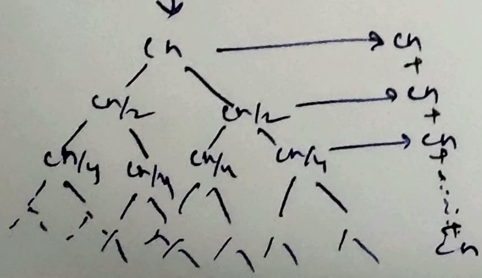
$\longrightarrow$ Divide into as many parts as recursive calls are there, in the next level.

$\downarrow$

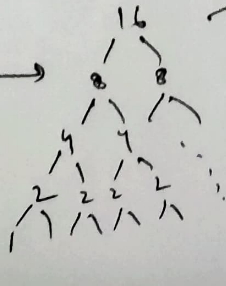


} Identify the pattern.

$\downarrow$



$\longrightarrow$



log(n) terms each of value cn

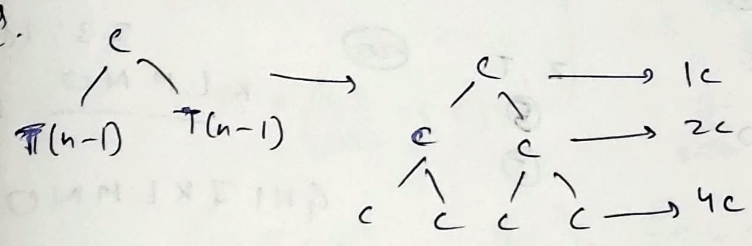
$\hookrightarrow \theta(n\log n)$ Ans.

Example 2

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

$T(1) = c$

Ans.



} Patten.

$\Rightarrow c + 2c + 4c \cdots \rightarrow (2^n \times c)$ } Using geometric progression.

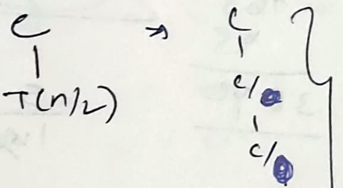
$\Rightarrow \Theta(2^n)$

Example 3

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

$T(1) = c$

Ans.



$\rightarrow$ Pattern. $\Rightarrow \underbrace{c + c + c \cdots c}_{\Theta(\log n)}$.

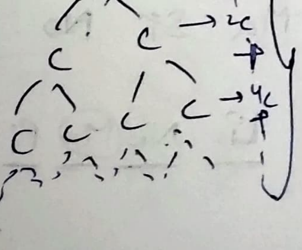
$\Rightarrow$ ~~$c + c/2 + c/4 + c/8 \Rightarrow c(1 + 1/2 + 1/4 + 1/8 \cdots$~~

Eg 4.

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

$T(1) = c$

★ In Eg 1 we had cn not c



$c + 2c + 4c \cdots \rightarrow \log(n)$ terms.

$\Rightarrow \Theta\left(\frac{2^{\log n} - 1}{2 - 1}\right)$

$\Rightarrow \underline{\Theta(n)}$.

Note

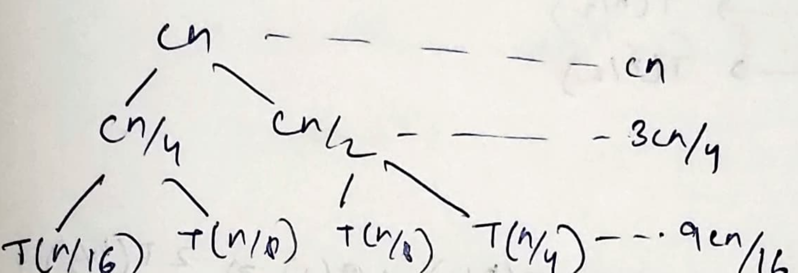
→ Sometimes it's not possible to find the exact bound for a relation.

→ we can just find the upper bound for it.

Eg:-

$$T(n) = T(n/4) + T(n/2) + cn$$

$$T(1) = c$$



though a pattern is there, we can see that right side will end later than left side of tree

→ ∴ we ~~take~~ the longer part of tree, right side in this case and find the upper bound.

→ If tree considered full → log(n) terms.

$$\Rightarrow cn + 3cn/4 + 9cn/16 \dots$$

As we are finding upper bound $\frac{a}{1-r}$ can be used, inst of $\left(\frac{r^n-1}{r-1}\right)$.

$$\Rightarrow O\left(\frac{cn}{1-3/4}\right) \Rightarrow O(n)$$