Q. Write a recursive program & recurrence relation to find GCD of m, n.

```
GCD(m,n)
{
  if(m==0)
    return n;
  else
    if(n==0)
      return m;
    else
      return (GCD(n-1, m, m));
}
```

$$GCD(m,n) = \begin{cases} m & \text{if } n=0 \\ n & \text{if } m=0 \\ GCD(n-1, m, m) & \text{otherwise} \end{cases}$$

NOTE:- Input-size doesn't decide best and worst case, logic decides.

- In this ~~case~~ program best and worst cases are introduced, as on the basis of input sometimes program stops immediately (best-case), & sometimes it runs more (worst-case).

best case :- when m and n are multiple of each other.

$T(n) = O(1)$.

worst-case :- when m and n are consecutive numbers

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

Recurrence Relation Solving :-

(i) Substitution method

(ii) Recursive tree method

(iii) Master Theorem

(i) Substitution method :- Substituting in the given func^n repeatedly until given func^n removed completely.

eg:-

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

termination condition

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

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

$= T(n-3) + n-2 + n-1 + n$

$n-1$ times

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

$= T(1) + 2 + 3 + 4 + \cdots + n$

$= 1 + 2 + 3 + \cdots n$

$= \frac{n(n+1)}{2}$

$= O(n^2) = \Omega(n^2) = \Theta(n^2)$

eg:-

$$Tn = \begin{cases} 1 & \text{if } n=1; \\ T(n-1) * n & \text{if } n>1; \end{cases}$$

$T(n) = T(n-1) * n$

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

$= T(n-3) * n-2 * n-1 * n$

K-times

$= T(n-K) * (n-(K-1)) * (n-(K-2)) * \cdots\cdots * n$

$n-K = 1$; so, $(K = n-1)$

$= T(n-(n-1)) * (n-(n-2)) * \cdots\cdots * n$

$= T(1) * 2 * 3 * \cdots\cdots * n$

$= 1 * 2 * 3 * \cdots\cdots * n$

$= n!$

$= O(n!)$

$= O(n^n)$

$= \Omega(2^n)$

$2^n < n! < n^n$

eg:-

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

$T(n) = T(n-2) + \log(n)$

$= T(n-4) + \log(n-2) + \log(n)$

$= T(n-6) + \log(n-4) + \log(n-2) + \log(n)$

$=$ ↓ K

$= T(n-2^K) + \log(n-2^{K-1}) + \log(n-2^{K-2}) + \cdots + \log(n)$

$n - n + 2$

$T(2)$

$n - 2^K = 0$

so, $n = 2^K$

$(K = \log_2 n)$

$= T(0) +$

$= T(n-2K) + \log(n-(2K-2)) + \log(n-(2K-4)) + \cdots + \log_2$

$n - 2K = 0$

$(K = n/2)$

$$= T(0) + \log_2 2 + \log_2 4 + \cdots\cdots + \log_2 n$$

$$= 1 + \log_2 2 + \log_2 4 + \cdots + \log_2 n$$

$$= 1 + \log_2(2\times 1) + \log_2(2\times 2) + \cdots\cdots + \log_2(2\times n/2)$$

$$= 1 + \log_2 2 + \log_2 1 + \log_2 2 + \log_2 2 + \cdots + \log_2 2 + \log n/2$$

$$= 1 + n/2 \log_2 2 + (\log_2 1 + \log_2 2 + \cdots \log_2 n/2)$$

$$= 1 + n/2 + \log_2(1\times 2\times 3\times \cdots \times n)$$

$$= 1 + n/2 + \log_2(n/2!) \rightarrow \text{Actual } T(n)$$

$n/2 \Rightarrow n! < n^n$

$n! = O(n^n)$

$$= 1 + n/2 + \log_2((n/2)^{n/2})$$

$$= 1 + n/2 + n/2 \log n/2$$

$$= O(n \log n) \rightarrow \text{asymptotic UB of } T(n)$$

Eg:-

$$T(n) = \begin{cases} 10 & \text{if } n = 0 \\ T(n-2) + n^2 & \text{if } n > 0 \end{cases}$$

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

$$T(n) = T(n-4) + (n-2)^2 + n^2$$

$$= T(n-6) + (n-4)^2 + (n-2)^2 + n^2$$

$$=$$

K-times

$$= T(n-2K) + (n-2K-2)^2 + (n-2K-4)^2 + \cdots + n^2$$

$$= \text{so, } n-2K = 0 \quad \text{so } (K = n/2)$$

$$= T(0) + 2^2 + 4^2 + 6^2 + \cdots n^2$$

$$= 10 + 2^2 + 4^2 + 6^2 + \cdots + n^2$$

$$= 10 + 2^2(1^2 + 2^2 + 3^2 + \cdots (n/2)^2)$$

$$= 10 + 4\left(\frac{n/2(n/2+1)(n/2+1)}{6}\right)$$

S_n^2

$$S_n^2 = \frac{n(n+1)(2n+1)}{6}$$

$$= O(n^3)$$

Eg:-

$$T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(n/2) + n & \text{if } n>1 \end{cases}$$

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

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

$$= T(n/8) + n/4 + n/2 + n$$

$$\vdots$$

$$= T(n/2^k) + n/2^{k-1} + n/2^{k-2} + \cdots + n$$

$$= n/2^k = 1 \qquad 2^k = n$$

$$(k = \log_2 n)$$

$S_n = \frac{a(r^n - 1)}{r-1}$

$S_n = \frac{a(1-r^n)}{1-r}$

$$= T(1) + 2 + 2^2 + 2^3 + \cdots n$$

$$= 2^0 + 2^1 + 2^2 + 2^3 + \cdots n$$

$$= 1 + n\left[\frac{1(1 - 1/2^{\log_2 n})}{1 - 1/2}\right]$$

$$= O(n)$$

$$= \frac{a(a^n - r)}{a - r}$$

$$\frac{2^{\log_2 n} - 1}{2-1} = n$$

Eg:-

$$T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(n/2) + c & \text{if } n>1 \end{cases}$$

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

$T(n) = T(n/4) + c + c$

$= T(n/8) + c + c + c$

$\vdots$

$= T(n/2^k) + kc$

$k = \log_2 n$

$= T(1) + \log_2 n \ (c)$

$= O(\log n)$

Eg:-

$$T(n) = \begin{cases} 1 & \text{if } n=2 \\ 8T(n/2) + n^2 & \text{if } n>2 \end{cases}$$

$$T(n) = 8T(n/2) + n^2$$

$$= 8\left(8T(n/4) + (n/2)^2\right) + n^2$$

$$= 8 \cdot 8^2 T(n/4) + 8n^2/2^2 + n^2$$

$$= 8^2\left(8T(n/8) + n^2/4^2\right) + 8n^2/2^2 + n^2$$

$$= 8^3 T(n/8) + 8n^2/4^2 + 8n^2/2^2 + n^2$$

$$= 8^3 T(n/8) + 4n^2 + 2n^2 + n^2 \cdot 2^0$$

$\vdots$ K rounds

$\downarrow$

$n/2^k = 2$

$2^{k+1} = n$

$k+1 = \log_2 n$

$(k = \log_2 n - 1)$

$$= 8^k T(n/2^k) + n^2\left[2^{k-1} + 2^{k-2} + \cdots 2^1 + 2^0\right]$$

$$= 8^{\log_2 n - 1} T(2) + n^2\left[2^{\log_2 n - 2} + 2^{\log_2 n - 3} + \cdots 2^1 + 2^0\right]$$

$$= \frac{n^3}{8} + n^2\left[1 + 2^1 + 2^2 + \cdots 2^{\log_2 n - 3} + 2^{\log_2 n - 2}\right]$$

S_n

$$= n^3/8 + n^2\left[\frac{1\left(2^{\log_2 n - 1} - 1\right)}{2 - 1}\right]$$

$$= n^3/8 + n^2\left(n/2 - 1\right)$$

$$= O(n^3)$$

Eg:-

$$T(n) = \begin{cases} 5 & \text{if } n=1 \\ 2T(n/2) + n\log n & \text{if } n>1 \end{cases}$$

$$T(n) = 2T(n/2) + n\log n$$

$$= 2\left[2T(n/4) + n/2 \cdot \log n/2\right] + n\log n$$

$$= 4T(n/4) + n \cdot \log n/2 + n\log n$$

$$= 4\left[2T(n/8) + n/4 \log n/4\right] + n\log n/2 + n\log n$$

$$= 8T(n/8) + n\log n/4 + n\log n/2 + n\log n$$

$\downarrow$

$(k = \log_2 n)$

$$= 2^k T(n/2^k) + n\left[\log\left(n/2^{k-1}\right) + \log\left(n/2^{k-2}\right) + \dots + \log\left(n/2^0\right)\right]$$

$$= 5n + n\left[\overbrace{\log n + \log n + \dots \log n}^{k\text{-times}} - \log 2^{k-1} - \log 2^{k-2} - \dots - \log 2^0\right]$$

$$= 5n + n\left[\log_{\log n} k \log n - \left((k-1) + (k-2) + \dots + 1 + 0\right)\right]$$

$$= 5n + n\left[(\log n)^2 - \frac{k(k+1)}{2}\right]$$

$$= 5n + n\left[(\log n)^2 - \frac{\log_n(\log_n + 1)}{2}\right]$$

$$= 5n + n\left[\frac{(\log_n)^2 - \log n}{2}\right]$$

$$= O(n(\log n)^2)$$

eg:-

$$T(n) = \begin{cases} 2 & \text{if } n = 2 \\ \sqrt{n}\,T(\sqrt{n}) + n & \text{if } n > 2 \end{cases}$$

$$T(n) = \sqrt{n}\,T(\sqrt{n}) + n$$

$$= n^{1/2}\,T(n^{1/2}) + n$$

$$= n^{1/2}(n^{1/4}\,T(n^{1/4}) + n^{1/2}) + n$$

$$= n^{1/2+1/4}\,T(n^{1/4}) + n + n$$

$$= n^{1/2+1/4}\,T(n^{1/4}) + 2n$$

$n^{1/2+1/4}$

$$= \left[n^{1/8}\,T(n^{1/8}) + n^{1/4}\right] + 2n$$

$\frac{6+1}{8} = \frac{7}{8}$

$$= n^{3/4+1/8}\,T(n^{1/8}) + n^{3/4+1/4} + 2n$$

$$= n^{7/8}\,T(n^{1/8}) + 3n$$

$$= n^{\frac{2^k-1}{2^k}} T(n^{1/2^k}) + kn$$

$n^{1/2^k} = 2$

$\frac{1}{2^k} \log_2 n = 1$

$2^k = \log_2 n$

$k = \log_2(\log_2 n)$

$$= n^{\frac{\log n - 1}{\log_2 n}} T(2) + n \log_2(\log_2 n)$$

$$= n^{1 - 1/\log_2 n} + n \log(\log n)$$

$$= \frac{n}{(n^{1/\log_2 n})} + n \log_2(\log_2 n)$$

$n^{1/2^k}$

$$= n/2 + n \log(\log n)$$

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

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

Eg:- $T(n) = 1 \cdot 2^1 + 2 \cdot 2^2 + 3 \cdot 2^3 + 4 \cdot 2^4 + \cdots\cdots (n-2) \cdot 2^{n-2} + (n-1) 2^{n-1} + n \cdot 2^n$

combination of A.P & G.P $\longrightarrow$ G.P

$$T(n) = 1 \cdot 2^1 + 2 \cdot 2^2 + 3 \cdot 2^3 + \cdots\cdots + n-2 \cdot 2^{n-2} + n-1\, 2^{n-1} + n 2^n$$

$\times 2$

$$2T(n) = 0 + 1 \cdot 2^2 + 2 \cdot 2^3 + \cdots\cdots + n-3\, 2^{n-2} + n-2\, 2^{n-1} + n-1\, 2^n$$

$$T(n) - 2T(n) = 1 \cdot 2^1 + 1 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4 + \cdots - 1 \cdot 2^{n-2} + 1 \cdot 2^{n-1} + 1\, 2^n - n 2^{n+1}$$

$$T(n) = \left[\frac{2(2^n-1)}{2-1}\right] - 2^{n+1}$$

$$T(n) = n \cdot 2^{n+1} - 2^{n+1} - 2$$

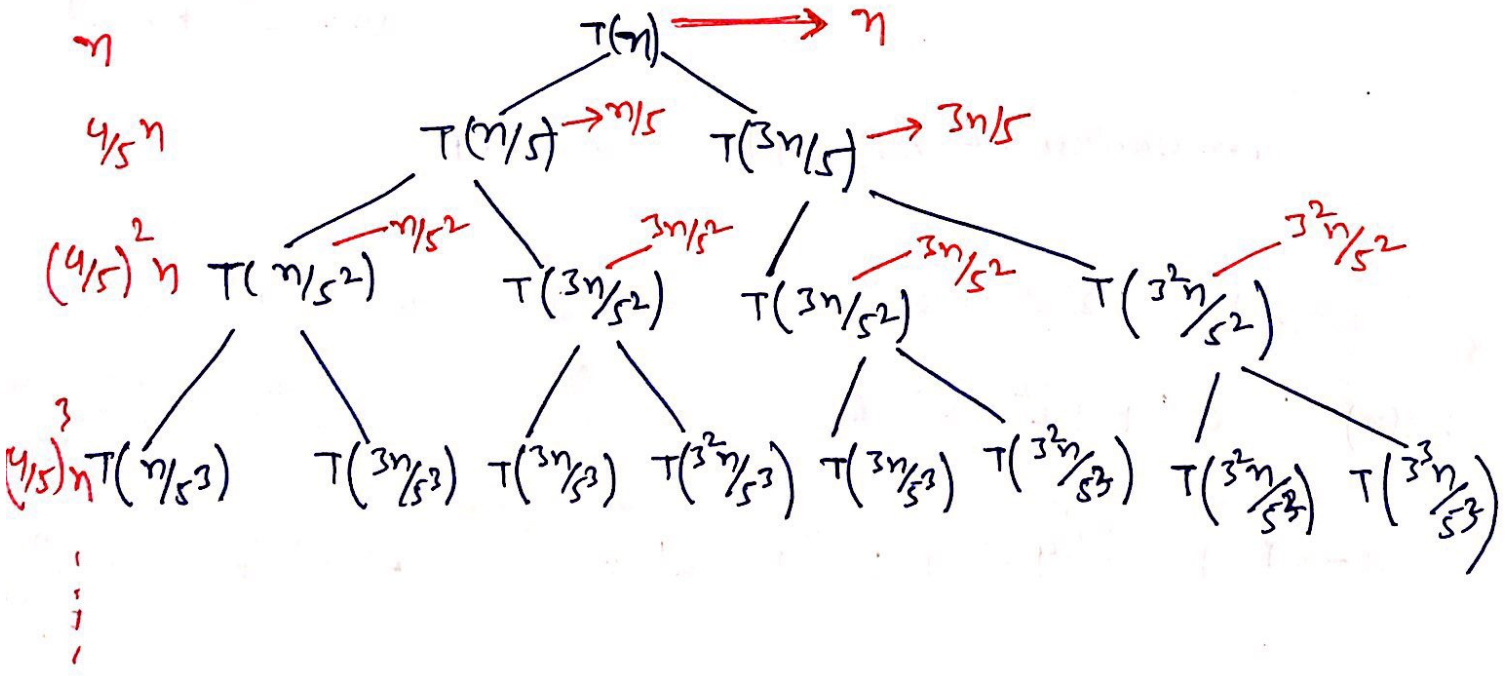
$$\boxed{T(n) = O(n \cdot 2^n)}$$

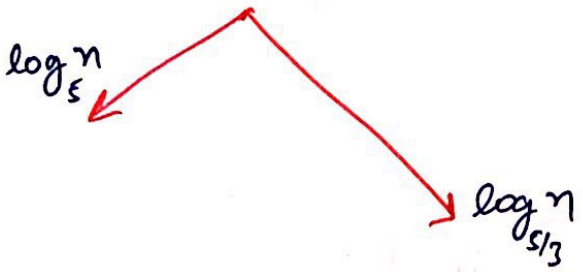
(ii) Recurrence Tree method :-

$$T(n) = T(n/5) + T(3n/5) + n$$

- $T(n/5)$ → I func^n call
- $T(3n/5)$ → II func^n call
- n → Time taken by each of func^n execution

* Whenever recurrence relation has more than 1 func^n call with different parameters then this method is applied.





$$T(n) \leq (4/5)^0 n + (4/5)^1 n + (4/5) n^2 + \cdots (4/5)^{\log_{5/3} n} \cdot n$$

$$T(n) \leq n \left(\underbrace{(4/5)^0 + (4/5)^1 + \cdots (4/5)^{\log_{5/3} n}}_{\text{sum of decreasing GP} = O(1)} \right)$$

$$\boxed{T(n) = O(n)} \quad \text{UB}$$

$$T(n) \geq (4/5)^0 n + (4/5)^1 n + \cdots + (4/5)^{\log_5 n}$$

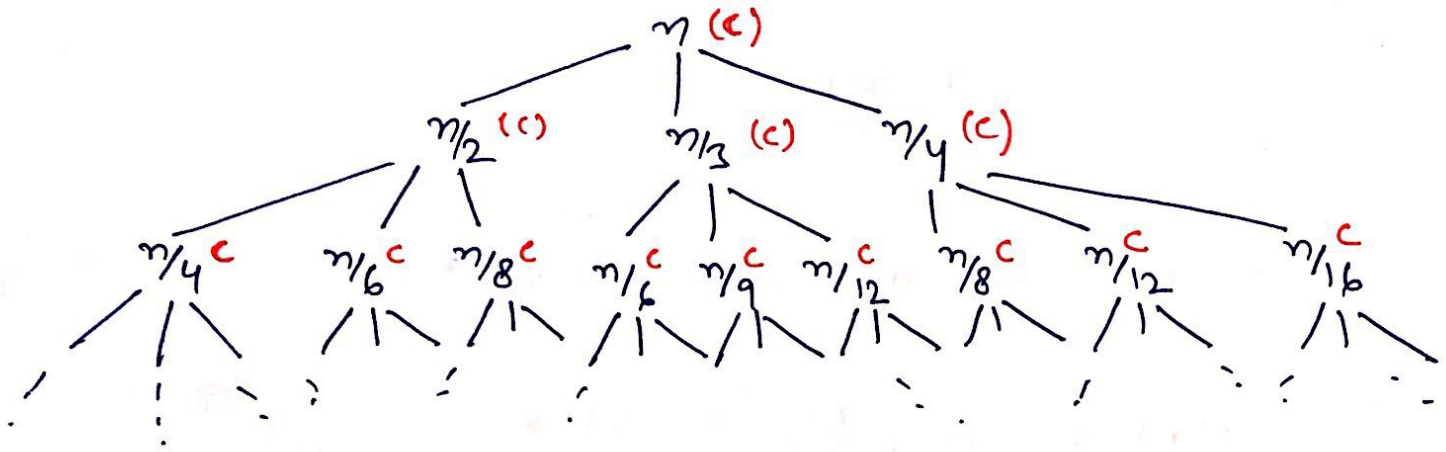
$$\geq n \left[\underbrace{(4/5)^0 + (4/5)^1 + \cdots + (4/5)^{\log_5 n}}_{\Omega(1)} \right]$$

$$\boxed{T(n) = \Omega(n)} \quad \text{LB}$$

* Exact $T(n)$ can't be calculated as height of tree is different at RHS & LHS so,

UB and LB are used.

Eg:- $T(n) = T(n/2) + T(n/3) + T(n/4) + c$



$(c + 3c + 9c + 27c + \cdots + 3^k c)$

$\log_2 n$ (UB)

$\log_4 n$ (LB)

$$T(n) \geq (c + 3c + 9c + \cdots 3^{\log_4 n} c)$$

$$\geq c(1 + 3 + 3^2 + 3^3 + \cdots 3^{\log_4 n})$$

$$\geq c\left(\frac{1(3^{\log_4 n} - 1)}{3-1}\right)$$

$$\geq c\left(\frac{n^{\log_4 3} - 1}{2}\right)$$

$T(n) = \Omega(n^{\log_4 3})$ LB

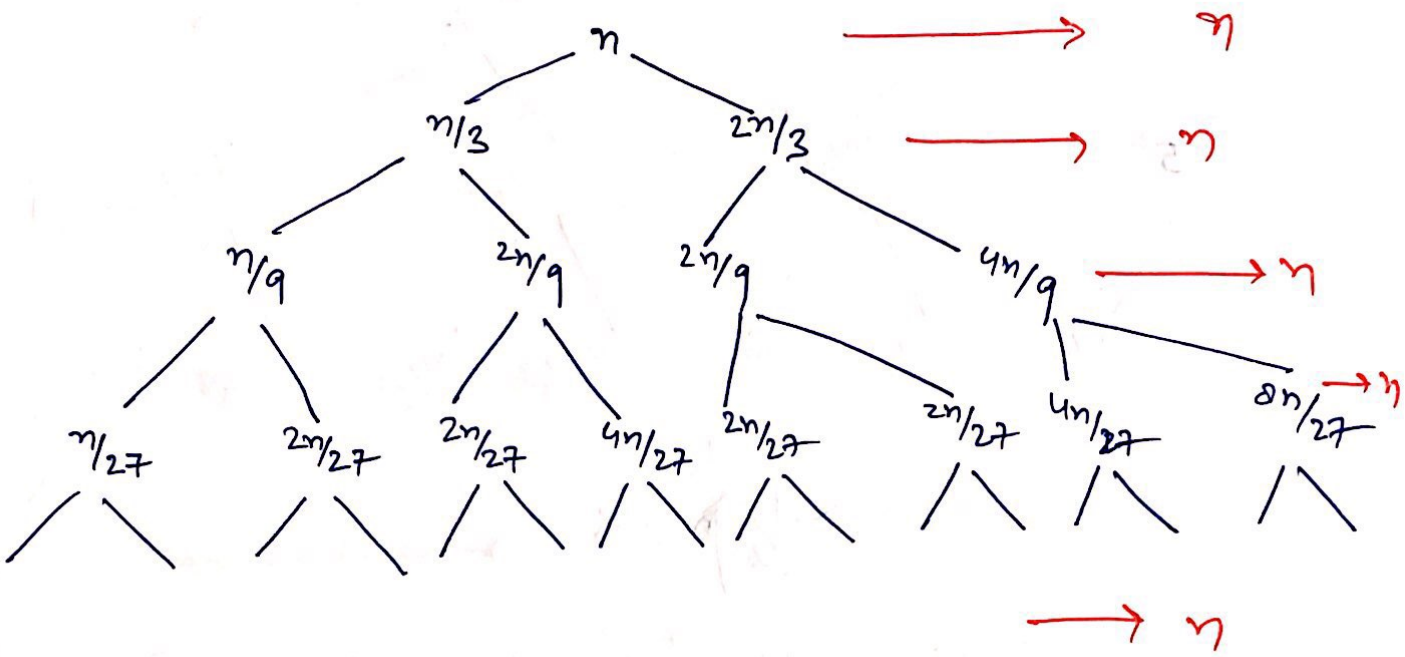
$$T(n) \leq c(c + 3c + 9c + \cdots 3^{\log_2 n} c)$$

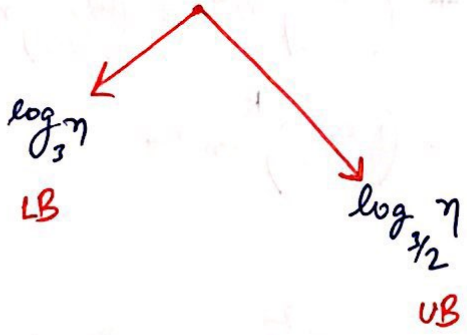
$$\leq c(1 + 3 + 3^2 + 3^3 + \cdots + 3^{\log_2 n})$$

$$\leq c\left(\frac{1(3^{\log_2 n} - 1)}{3-1}\right)$$

$T(n) = O(n^{\log_2 3})$ UB

Eg:- $T(n) = T(n/3) + T(2n/3) + n$





$T(n) = n \cdot K$

UB
$K = \log_{3/2} n$

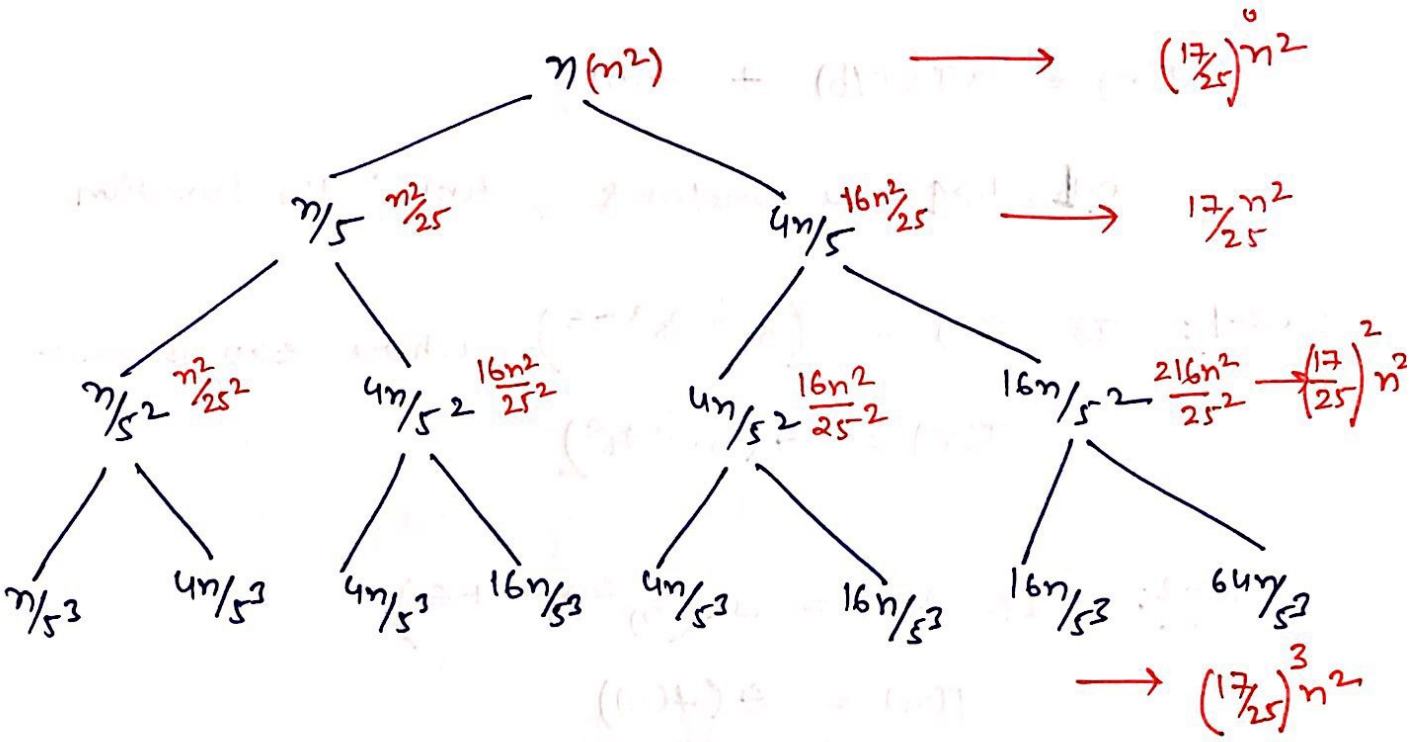
LB
$K = \log_3 n$

$T(n) = O(n \log_{3/2} n)$

$T(n) = \Omega(n \log_3 n)$

$T(n) = \Theta(n \log n)$ ← (UB ≈ LB)

Eg:- $T(n) = T(n/5) + T(4n/5) + n^2$



$$T(n) = \left(\frac{17}{25}\right)^0 n^2 + \left(\frac{17}{25}\right)^1 n^2 + \left(\frac{17}{25}\right)^2 n^2 + \cdots$$

$$\cdots\cdots + \left(\frac{17}{25}\right)^k n^2$$

$$= n^2 \left[\underbrace{\left(\frac{17}{25}\right)^0 + \left(\frac{17}{25}\right)^1 + \left(\frac{17}{25}\right)^2 + \cdots + \left(\frac{17}{25}\right)^k}_{O(1)\ \text{sum of decreasing GP}} \right]$$

$$T(n) = O(n^2)$$

$$= \Omega(n^2)$$

$$\boxed{T(n) = \Theta(n^2)}$$

(iii) Master Theorem :-

$$T(n) = aT(n/b) + f(n)$$

$a \geq 1$, $b > 1$ are constants, $f(n)$ is the function

case-1: If $f(n) = O(n^{\log_b a - \epsilon})$ where $\epsilon > 0$ is const

$$T(n) = \theta(n^{\log_b a})$$

case-2: If $f(n) = \Omega(n^{\log_b a + \epsilon})$ //

$$T(n) = \theta(f(n))$$

case-3: If $f(n) = \theta(n^{\log_b a} * (\log n)^k)$ where k is const $k \geq 0$

$$T(n) = \theta(n^{\log_b a} * (\log n)^{k+1})$$

- In case 1, comparing $f(n)$ and $n^{\log_b a}$, $n^{\log_b a}$ polynomial time greater than $f(n)$.

$\downarrow$

n^{ϵ}

$$f(n) \leq \frac{n^{\log_b a}}{n^{\epsilon}}$$

- In case 2, $n^{\log_b a}$ is smaller than $f(n)$ by n^{ϵ}.

$$f(n) \geq n^{\log_b a} * n^{\epsilon}$$

- In case 3, both $f(n)$ and $n^{\log_b a}$ are asymptotically equal.

Eg: $T(n) = 8T(n/2) + n^2$

a, b, $f(n)$

$f(n) = n^2$

$n^{\log_b a} = n^{\log_2 3}$

$= n^3$

so, $f(n) \leq n^{\log_b a}$

$f(n) = O(n^{\log_b a})$ CASE-1

so, $T(n) = O(n^{\log_b a}) = O(n^3)$

$$T(n) = \Theta(n^3)$$

ALTERNATIVE :- $T(n) = \max(f(n), n^{\log_b a})$

only in CASE-1 and CASE-2

eg:- $T(n) = 2T(n/2) + n^2$

a, b, $f(n)$

$f(n) = n^2$; $n^{\log_b a} = n^{\log_2 2} = n^1$

$f(n) \geq n^{\log_b a}$

$f(n) = \Omega(n^{\log_b a + \epsilon})$

$T(n) = \Omega(f(n))$

$T(n) = \Theta(n^2)$

eg:- $T(n) = 1 \cdot T(n/2) + c$

a, b, $f(n)$

$f(n) = c$ $n^{\log_b a} = n^{\log_2 1}$

$= n^0 = \text{const}$

$f(n) = \Theta(n^{\log_b a} \cdot (\log n)^k)$

put $k = 0$ to make them equal

so, $T(n) = \Theta(n^{\log_b a} \cdot (\log n)^{0+1})$

$T(n) = \Theta(c \cdot \log n)$

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

eg:- $T(n) = 2^n T(n/2) + n^n$

a ↙ b ↓ f(n) ↘

since $a = 2^n$ which is not constant

so, Master Theorem can't be applied.

eg:- $T(n) = 2T(n/2) + n\log n$

a=2 b=2 f(n)

$f(n) = n\log n$

$n^{\log_b a} = n^{\log_2 2}$

$= n$

$f(n) > n^{\log_b a}$

$> n$

f(n) is greater than $n^{\log_b a}$ by $\log n$

so, CASE-2 is applied

$$f(n) = \Theta\left(n^{\log_b a} \cdot (\log n)^k \right)$$

put $K = 1$ to make them equal

so, $$T(n) = \Theta\left(n^{\log_b a} \cdot (\log n)^{k+1} \right)$$

$$\boxed{T(n) = \Theta(n (\log n)^2)}$$

NOTE:- If difference b/w $f(n)$ and $n^{\log_b a}$ is:

- polynomial or more than β it → CASE-1, 2
- logarithmic → CASE-3

NOTE :- If $f(n) < n^{\log_b a}$ by $\log n$ then case-3 can't be applied,

case-3 can only be applied when –

$$f(n) > n^{\log_b a}$$

$$\text{i.e } f(n) = \Theta\left(n^{\log_b a} * (\log n)^k \right)$$

Apply substitution method if case-3 is also failed.

NOTE :-

$f(n)$ → n

$n^{\log_b a}$ → $n^2 \log n$

$T(n) = n^2 \log n$

polynomial + logarithm

(n) $(\log n)$

so, CASE - 1 can be applied.

eg:- $f(n) = n$ $n^{\log_b a} = \log n$

$f(n) > n^{\log_b a}$

by $n/\log n$ since n is present

apply case - 2 :

$T(n) = \max(f(n), n^{\log_b a})$

$T(n) = \theta(n)$

eg:- $f(n) = n$ $n^{\log_b a} = n^n$

since $n^n > n^c$ (polynomial) time

$T(n) = \max(f(n), n^{\log_b a})$ so, CASE 1, 2 can be applied

$T(n) = \theta(n^n)$

NOTE :- If recurrence relation contain $\sqrt{}$ operator, then convert it into master theorem problem

(i) $T(n) = T(\sqrt{n}) + c$ (const)

- assume $n = 2^k$

$$T(2^k) = T(2^{k/2}) + c$$

- assume $T(2^k) = S(k)$

$$S(k) = S(k/2) + c$$

Now apply master theorem CASE-3

we get $S(k) = \theta(\log k)$

$$T(2^k) = \theta(\log k)$$

Now, $\boxed{T(n) = \theta(\log(\log n))}$

(ii) $T(n) = 2T(\sqrt{n}) + \log_2 n$

assume

- assume $n = 2^k$

$$T(2^k) = 2T(2^{k/2}) + \log_2 2^k$$

$$T(2^k) = 2T(2^{k/2}) + k$$

- assume $S(k) = T(2^k)$

$$S(k) = 2S(k/2) + k$$

$$S(k) = \theta(k \cdot \log k) \qquad \text{CASE-3}$$

$$T(2^k) = \theta(k \cdot \log k)$$

since $n = 2^k$ so,

$$T(n) = \theta(\log n \cdot \log(\log n))$$

NOTE :-

$$T(n) = 2T(\lfloor n/2 \rfloor) + c \iff 2T(n/2) + c$$

$$T(n) = 2T(n/2 - 10) + c$$