$$a_0 t_n + a_1 t_{n-1} + \cdots + a_k t_{n-k} = b^n p(n)$$

* Change of Variable :-

T(n) is general recurrence
t_i is term of new obtained recurrence by this method

Ex:- Recurrence

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 3T(n/2) + n & \text{if } n \text{ is Power of } 2,\ n > 1 \end{cases}$$

→ replace n by 2^i
So, It achieves new recurrence t_i defined by $t_i = T(2^i)$

$$n/2 = \frac{2^i}{2} = 2^{i-1}$$

In other words, original recurrence
T(n) becomes t_i
T(n/2) " t_{i-1}

rewrite recurrence,

$$t_i = T(2^i) = 3T(2^{i-1}) + 2^i$$
$$= 3t_{i-1} + 2^i$$

$$t_i - 3t_{i-1} = 2^i$$

Compare it with eq^n (1)

The char. Poly. is,

$(x-3)(x-2)$

roots are $r_1 = 3$ & $r_2 = 2$

General Form, $\boxed{t_i = c_1 3^i + c_2 2^i}$ ——(A)

$T(2^i) = t_i \Rightarrow T(n) = t_{\lg n}$ $\left\{ \begin{array}{l} n = 2^i \\ i = \log n \end{array} \right.$

$T(n) = c_1 3^{\log n} + c_2 2^{\log n}$

$\boxed{T(n) = c_1 n^{\log 3} + c_2 n}$ ——(B)

Initial Con^d,

if $n=1$ $c_1 + c_2 = 1$

if $n=2$ $3c_1 + 2c_2 = 5$

$c_1 2^{\log 3}$
$= c_1 3^{\log}$
$= 3c_1$

$c_1 = 3$ & $c_2 = -2$

$T(2) = 3T(1) + 2$
$= 3 + 2$
$= 5$

$T(n) = 3n^{\log 3} - 2n$

$\boxed{T(n) = O(3n^{\log 3})}$

Ex:- Consider recurrence

$T(n) = 4T(n/2) + n^2$ when n is Power of 2, $n \geq 2$

$\rightsquigarrow$ $t_i = T(2^i) = 4T(2^{i-1}) + (2^i)^2$

$= 4t_{i-1} + 4^i$

rewrite recurrence,

$t_i - 4t_{i-1} = 4^i$

Compare it with eq^n (1)

$(x-4)(x-4)$

roots are $r_1 = 4$ $m_1 = 2$

General Form,

$t_i = c_1 4^i + c_2 i 4^i$ —— (A)

In n term,

$T(n) = c_1 4^{\log_2 n} + c_2 (\log n) 4^{\log_2 n}$

$\boxed{n = 2^i \\ i = \log_2 n}$

$T(n) = c_1 n^2 + c_2 n^2 \log n$ —— (B)

original recurrence,

Date: / /
Page No.:

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

Put value of $T(n)$ from eqn (3)

$$= c_1 n^2 + c_2 n^2 \log n - 4\left(c_1\left(\frac{n}{2}\right)^2 + c_2\left(\frac{n}{2}\right)^2 \log n/2\right)$$

$$= \cancel{c_1 n^2} + c_2 n^2 \log n - \cancel{c_1 n^2} - c_2 n^2 \log n/2$$

$$n^2 = c_2 n^2 \log n - c_2 n^2 \log n/2$$

$$1 = c_2 \log n - c_2 \log n/2 = c_2\left[\log_2 n - \log_2 n/2\right]$$

$$= c_2\left[\log_2\left(\frac{n}{n/2}\right)\right]$$

$$= c_2 * 1$$

$$\boxed{c_2 = 1}$$

$$\boxed{T(n) = \Theta(n^2 \log n \mid n \text{ is Power of } 2)}$$

Ex:- $T(n) = 2T\left(\frac{n}{2}\right) + n\log_2 n$

we take $n = 2^i$

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

$$t_i = 2t_{i-1} + i\,2^i$$

$$t_i - 2t_{i-1} = i\,2^i \quad \text{——— (1)}$$

$$(x-2)\ (x-2)^2$$

$$(x-2)^3$$

$t_i = c_1 2^i + c_2 i 2^i + c_3 i^2 2^i$

(B) $T(n) = c_1 n + c_2 n \log_2 n + c_3 n \log_2^2 n$ — (2)

Put eq^n (2) in original recurrence

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

$= c_1 n + c_2 n \log_2 n + c_3 n \log^2 n - 2[c_1 \frac{n}{2} + c_2 \frac{n}{2} \log_2 n/2 + c_3 \frac{n}{2} \log_2^2 n/2]$

$= c_2 n [\log n - \log \frac{n}{2}] + c_3 n [\log^2 n - \log^2 \frac{n}{2}]$

ex:- $a^2 - b^2$

$= c_2 n + c_3 n [(\log n - \log \frac{n}{2})(\log n + \log n/2)]$

$= c_2 n + c_3 n [\log (n \cdot \frac{n}{2})]$

$= c_2 n + c_3 n \cdot \log_2 n^2/2$

$= c_2 n + c_3 n [\log n^2 - \log 2]$

$= c_2 n + c_3 n [\log n^2 - 1]$

$n \log n = (c_2 - c_3) n + 2 c_3 n \log n$

$\Rightarrow c_2 - c_3 = 0$ & $2c_3 = 1 \Rightarrow c_3 = \frac{1}{2}$

$c_2 = \frac{1}{2}$

$T(n) \in \Theta(n \log^2 n)$

Date: / /
Page No.:

MASTER METHOD

master th^m

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

→ $a \geq 1$, $b > 1$ are constants
→ $f(n)$ is asympto. pos. fu^n
→ It describes the runtime of algo. that divides a Prob. of size n into a subproblems, each of size n/b
→ a SubProb. are solved recursively in time $T(n/b)$
→ The cost of diving prob. & Combing sol^n $= f(n)$
⇒ $T(n)$ can be bounded asym as follows:

① If $f(n) = O(n^{\log_b a-\varepsilon})$, for some Const $\varepsilon > 0$ then, $\boxed{T(n) = \Theta(n^{\log_b a})}$
here, we try to proove, $f(n) \leq n^{\log_b a-\varepsilon}$

② If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n$
here, $f(n) = n^{\log_b a}$

③ If $f(n) = \Omega(n^{\log_b a+\varepsilon})$, for some const. ε
and if $af(n/b) \leq c.f(n)$, for const $c < 1$
then $T(n) = \Theta(f(n))$
we try to proove, $f(n) \geq n^{\log_b a+\varepsilon}$

⇒ Some Examples :-

Ex-1 $T(n) = 9T(n/3) + n$

here $a = 9$, $b = 3$, $f(n) = n$

$$\therefore n^{\log_b a} = n^{\log_3 9}$$

$$= n^{\log_3 3^2}$$

$$= n^2$$

here, $n^{\log_b a} = n^2 > f(n)$

if we take $\varepsilon = 1$, then also $n^{\log_b a - \varepsilon} = n \geq f(n)$

we can apply case 1

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

Ex:- 2 $T(n) = T\left(\frac{2n}{3}\right) + 1$

here, $a = 1$, $b = 3/2$, $f(n) = 1$

$$\therefore n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$$?

now, $f(n)$ is also 1

So, we have $n^{\log_b a} = f(n)$

$$\therefore T(n) = \Theta(\log_2 n)$$

Ex:-3 $T(n) = 3T\left(\frac{n}{4}\right) + n \lg n$

here $a = 3$, $b = 4$ $f(n) = n \lg n$

$$\frac{\log_2 3}{\log_2 4}$$

$$n^{\log_b a} = n^{\log_4 3} = n^{0.793}$$

here, $n^{\log_b a} \leq f(n)$, if we take $\varepsilon = 0.2$

then also, $n^{\log_4 3 + \varepsilon} \leq f(n)$

here case 3 applies, so check regularity Con^d

we have $a \cdot f(n/b) = 3 \cdot f(n/b)$

$$= 3\left(\frac{n}{4}\right) \lg\left(\frac{n}{4}\right)$$

$$\leq \frac{3}{4} n \lg n$$

$$= c f(n), \quad c = \frac{3}{4} < 1$$

$\therefore T(n) = \Theta(n \lg n)$

Ex:-4 $T(n) = 2T(n/2) + n \lg n$

$a = 2 \quad b = 2 \quad f(n) = n \lg n$

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

here, $n^{\log_b a} < f(n)$, case 3 can be applied.

→ but $f(n)$ is not polynomialy larger

b'cz ratio $\frac{f(n)}{n^{\log_b a}} = \frac{n \lg n}{n} = \lg n$

is asymp. less than n^{ε} for +ve const ε.

⇒ Thus, it can't be solved by master's th^m

How to find log with base 2 ?

$\rightarrow$ In Calc : log $fu^n = \log_{10}$ & ln $fu^n = \log_e$

eg. $\log_2 3 = \frac{\log_{10} 3}{\log_{10} 2} \cong \frac{\ln 3}{\ln 2} = 1.59$

Ex:-5 $T(n) = 2T(n/2) + n^3$

here $a=2$ $b=2$ $f(n) = n^3$

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

so, $f(n) \geq n^{\log_b a + \varepsilon}$, where $\varepsilon = 2$

Case 3 applies

$a \cdot f\left(\frac{n}{b}\right) = 2\left(\frac{n}{2}\right)^3 = \frac{2n^3}{8} = \frac{1}{4} n^3$

$= cn^3$

where $c = \frac{1}{4} < 1$

Thus, $T(n) \in \Theta(n^3)$

Ex:-6 $T(n) = 16T\left(\frac{n}{4}\right) + n^2$

$a = 16$
$b = 4$
$f(n) = n^2$

$n^{\log_b a} = n^{\log_4 16} = n^2$

Case 2

here, $n^{\log_b a} = f(n)$, so $T(n) = \Theta(n^2 \cdot \lg n)$

Ex:-7 $T(n) = 7T\left(\frac{n}{2}\right) + n^2$

$a = 7$
$b = 2$
$f(n) = n^2$

$\therefore n^{\log_b a} = n^{\log_2 7} = n^{2.808}$

Since $2 < \lg 7 < 3$,

we have ~~Case 1~~

$\boxed{\varepsilon = 0.808}$

$\therefore n^2 = O(n^{\log_2 7 - \varepsilon})$, $\varepsilon > 0$

Case 1

$\therefore T(n) = \Theta(n^{\lg 7})$

Date:
Page No:

Ex:-8 $T(n) = T\left(\frac{9n}{10}\right) + n$

$= \theta(n)$

Ex:-9 $T(n) = 7T\left(\frac{n}{3}\right) + n^2$

$= \theta(n^2)$

Ex:-10 $f(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$ $\quad \left(\sqrt{n} = n^{\frac{1}{2}}\right)$

$= \theta(\sqrt{n}\, lg\, n)$

→ master method provides a method for solving recurrences of the form

$$T(n) = aT(n/b) + f(n) \quad \text{——— (1)}$$

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

where, a & b are constants
$a \geq 1$, $b > 1$
$f(n)$ is an asymptotically positive fuⁿ

eqⁿ (1) describes the running time of an algo. that divides a problem size n into subproblems, each size n/b

→ The a subproblems are solved recursively, each in Time $T(n/b)$

→ Cost of dividing the problem & combining the results of subproblems is described by fuⁿ $f(n)$

$$f(n) = C(n) + D(n)$$

Combining   dividing

Date: / /
Page No.:

⇒ Master thm :-

master method depends on this thm.

Let $a \geq 1$ & $b > 1$ be constants, let $f(n)$ be a fun and $T(n)$ be defined on the nonnegative integers by the recurrence

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

where we interpret n/b is size of each subproblem.
Then $T(n)$ can be bounded asymptotically as follows :

Case 1 : If $f(n) = O(n^{\log_b a - \varepsilon})$ for some consta[nt] $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$

case 2 : If $f(n) = \Theta(n^{\log_b a})$, then
$T(n) = \Theta(n^{\log_b a} \log n)$

case 3 : If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ ~~for Constant~~ ~~$\varepsilon > 0$~~ ~~&~~ ~~if~~
i) for Constant $\varepsilon > 0$
ii) if $af(n/b) \leq c f(n)$ for constant $c < 1$ & all sufficiently large n, then

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

$n^{\log_b a}$ → Sol^n of recursion

why base b ? → every subproblem is divided by b

<u>Explination</u> :-

→ In 3 cases, we are comparing fu^n $f(n)$ with the fu^n $n^{\log_b a}$

→ In case 1, fu^n $n^{\log_b a}$ is larger, then Sol^n is $T(n) = \Theta(n^{\log_b a})$

→ In Case 3, fu^n $f(n)$ is larger, then Sol^n is $T(n) = \Theta(f(n))$

→ In Case 2, two fu^{ns} are the same size, we multiply by a logarithmic factor & Sol^n is $T(n) = \Theta(n^{\log_b a} \lg n)$

$= \Theta(f(n) \lg n)$

In 1st Case → $f(n)$ be smaller than $n^{\log_b a}$, it must be polynomially smaller.
That is, $f(n)$ must be asymptotically smaller than $n^{\log_b a}$ by a factor of n^{ϵ}, $\epsilon > 0$

In 3rd Case → $f(n)$ be larger than $n^{\log_b a}$, it must be polynomially larger & satisfy the regularity Con^d that $af(n/b) \leq cf(n)$.
This Con^d is satisfied by most of the polynomially bounded fu^{ns} that we shall encounter.

⇒ Steps of master method to solve reecurrence :-

① find out a, b & f(n)

② ~~Put values in~~ ~~$n^{\log_b a}$~~ & Calculate $n^{\log_b a}$

③ Compare f(n) & $n^{\log_b a}$

↓↓ possibility

Case 1 — find out
Case 2 ⟶ T(n)
Case 3

④ Case 1 :- ε ?
If ε > 0 then find out T(n)

⑤ Case 3 :-
① ε ? ε > 0
② Regularity Con^d

then find out T(n)

⇒ Polynomialy ? (ε → ?)

ex:- ① g(n) = n+5
f(n) = n

by showing I can say g(n) is larger

but ⇒ $\frac{g(n)}{f(n)} = \frac{n+5}{n}$ (both are equal polynom~~ ~~ ε = 0

for that I can't apply M.T

(2) $\left.\begin{array}{l} g(n) = n^{3/2} \\ f(n) = n \end{array}\right\} \Rightarrow \dfrac{g(n)}{f(n)} = \dfrac{n^{1.5}}{n}$

$= n^{0.5}$

$= n^{\varepsilon}$

$\boxed{\varepsilon > 0}$

(3) $\dfrac{g(n)}{f(n)} = \dfrac{\cancel{n} \log n}{\cancel{n}} = n^{0} \cdot \log n$

$= n^{\varepsilon} \log n$

for that, $\boxed{\varepsilon = 0}$

master th^m can not applied

In R.T. - each node represents the cost ... somewhere ... each level of tree to obtain a set of per level costs & the cost of all levels of recursion.

* Recursion Tree :-

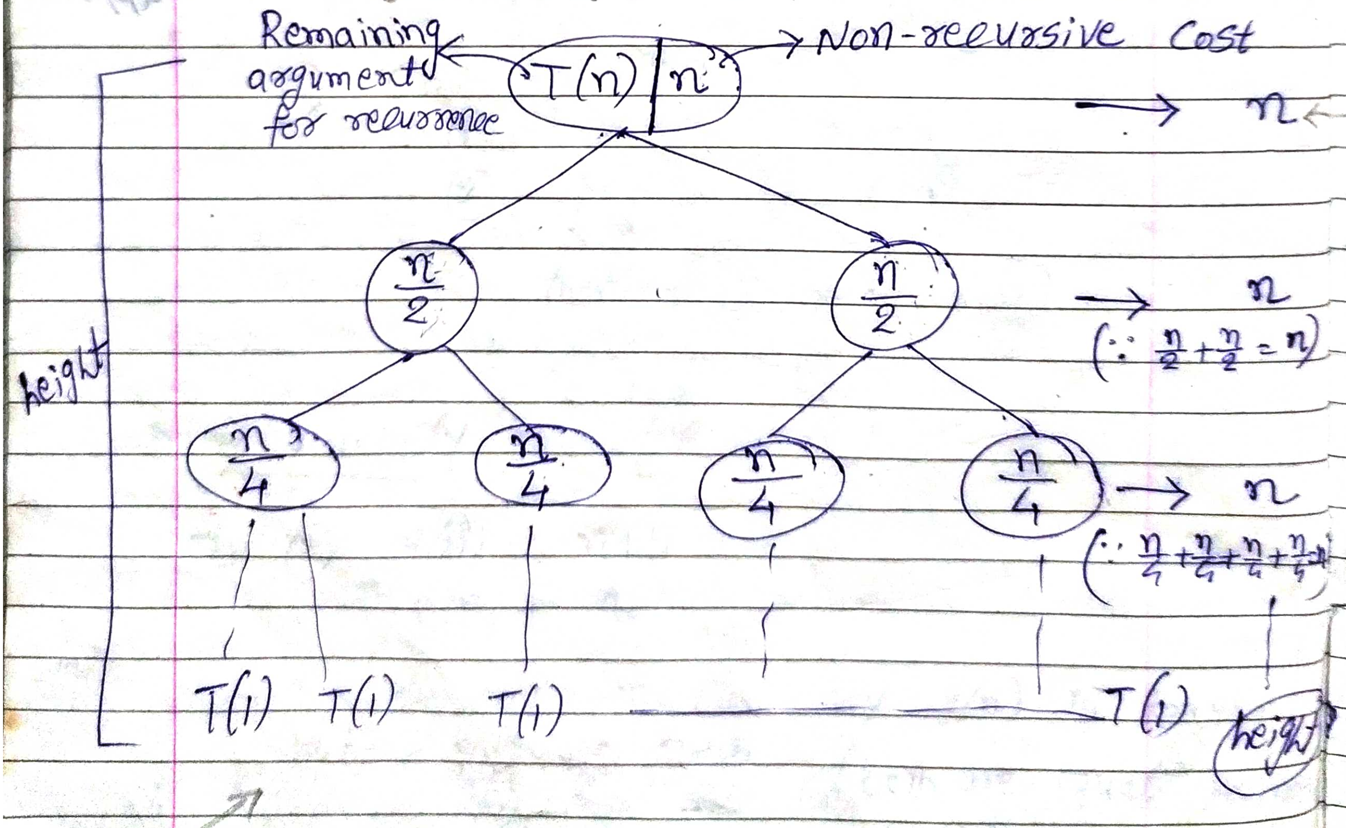
→ It is a Pictorial representation of recurrence which is in form of tree where each level nodes are expanded.

→ It is used to keep track of the size of remaining arguments in the recurrence & non-recursive costs.

ex:- Recurrence

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

Constructing recursion tree for recurrence



fully expanded tree

... the set of recursive ... invocations, we sum the costs within ... sum all the Pre level costs to determine the total

To Calculate height

From 1^{st} tree take

subprob. size for node at depth k.

$n = 2 \Rightarrow \frac{2}{2} = 1 \Rightarrow k = 1$

$n = 4 \Rightarrow$

each time sub prob. size decrease by factor 2. So it's time to find height means how far from the root do we reach one?

$n = 2 \Rightarrow 2^1 \Rightarrow k = 1$

$n = 4 = 2^2 \Rightarrow k = 2$

$n = 8 = 2^3 \Rightarrow k = 3$

In general $n = 2^k \Rightarrow \boxed{k = \log_2 n}$

cost of top level of recursion

$$\boxed{height = \log_2 n}$$

further expanding node with $T(n/2)$

$T(n) = n + n + n + \cdots + \log n$ times

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

$= n(1+2+4+ \cdots \log$