AA : Mod 1

- **Recurrance**

It refers to a way of describing how time complexity of an algo. depends on the size of its input.
Solving Recurrances : Substitution, Iteration, Master methods.

- **Substitution Method** [guessing]

Make an educated guess about form of soln. and prove it correct through mathematical induction. Can prove both upper bounds $O()$ and lower bounds $\Omega(\cdot)$ Examples:-

1 Solve the equation by Substitution Method $T(n) = T(n/2) + 1$

We have to show that it is asymptotically bound by $O(\log n)$

→ For $T(n) = O(\log n)$, we have to show that for some constant c

$T(n) \leq c \log n$: Put this in given recurrance eqn.

$T(n) \leq c \log(n/2) + 1$

$T(n) \leq c \log(n) - c \log_2 2 + 1$

$T(n) \leq c \log n$ for $c \geq 1$

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

→ $T(n) = T(n/2) + c$ — ①

$T(n/2) = T(n/4) + c$ — ②

$T(n/4) = T(n/8) + c$ — ③

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

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

$= T(n/2^3) + 3c$

k times

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

$n = 2^k \longrightarrow \log n = k \log 2$ → $k = \log n$

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

$1 + kc$: Time complexity

$= 1 + \log n.c = O(\log n)$

3 $T(n) = \begin{cases} 1 & \text{if } n = 1 \\ n\, T(n-1) & \text{if } n > 1 \end{cases}$

→ $T(n) = n * T(n-1)$ — ①

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

$= (n-1) T(n-2)$ — ②

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

$= (n-2) T(n-3)$ — ③

$T(n) = n(n-1) T(n-2)$: ② in ①

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

Page No.
Date / /

eliminate this

$T(n) = n(n-1)(n-2)\,T(n-3)$

(n-1) steps

$= n(n-1)(n-2)(n-3)\ldots T(n-(n-1))$

$= n(n-1)(n-2)(n-3)\ldots T(1)$

$= n(n-1)(n-2) \ldots 3 \times 2 \times 1$

$= n \times n(1-1/n) \times n(1-2/n) \times \ldots n(3/n) \cdot n(2/n) \times n(1/n)$

$= O(n^n)$

$n \times n \times n \ldots = n^n$

4 $T(n) = \begin{cases} 1 & \text{if } n=1 \\ 2T(n/2)+n & \text{otherwise} \end{cases}$ — ①

$T(n/2) = 2T(n/4) + n/2$ — ②

$T(n/4) = 2T(n/8) + n/4$ — ③

$T(n/8) = 2T(n/16) + n/8$

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

$T(n) = 2^2\left[T(n/2^2)\right] + 2n$ substitute ③ in this

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

$= 2^3\,T(n/2^3) + 4n \ldots$

k times

$= 2^k\,T(n/2^k) + kn$ ∵ end this / terminate

to $T(1)$ $n/2^k = 1 \rightarrow n = 2^k$

$= 2^k\,T(1) + n\log n$ $\log n = \log_2 2^k = k$

$= n + n\log n$ dominating

$= O(n\log n)$

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

$T(n-1) = T(n-2) + \log(n-1)$ — ②

$T(n-2) = T(n-3) + \log(n-2)$ — ③

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

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

k times

$= T(n-k) + \log(n-(k-1)) + \log(n-(k-2)) + \ldots \log n$

end this

Page No.
Date / /

$n-k=1 \rightarrow k=n$

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

$T(n) = 1 + \log(n-(n-1)) + \log(n-(n-2)) + \dots \log n$

$= 1 + \log 1 + \log 2 + \dots \log(n)$

$= 1 + \log(1.2.3.4 \dots n) = 1 + \log(n!) = 1 + \log n^n$

$= O(n \log n)$

$n(n-1)(n-2)(n-3)\dots$

$= n \times n \times n \times n = n^n$

Cookbook approach

- Master Theorem / Method [solve subproblems, combine their soln.]

Used to analyze time complexity of divide and conquer algorithms (break down problem into subproblems).

$T(n) = aT(n/b) + f(n)$: form.

a: no. of sub problems

n/b: size of each subproblem

From yt

$a \geq 1$, $b > 1$

Soln: $T(n) = n^{\log_b a}[u(n)]$

$R(n) = \dfrac{f(n)}{n^{\log_b a}}$

$R(n)$	$u(n)$
$n^r, r>0$	$O(n^r)$
$n^r, r<0$	$O(1)$
$(\log n)^i, i \geq 0$	$\dfrac{(\log_2 n)^{i+1}}{i+1}$

to get these ← check these

From notes

$$T(n) = \begin{cases} \Theta(n^{\log_b a}) & f(n) = O(n^{\log_b a-\varepsilon}) \\ \Theta(n^{\log_b a}\log n) & f(n) = \Theta(n^{\log_b a}) \\ \Theta(f(n)) & f(n) = \Omega(n^{\log_b a+\varepsilon}) \\ & \text{AND } af(n/b) < cf(n) \text{ for large } n \end{cases} \quad \varepsilon > 0,\ c < 1$$

Answers

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

→ $a=9$ $b=3$ $f(n)=n$

$n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$

Since $f(n) = O(n^{\log_3 9-\varepsilon})$ where $\varepsilon=1$, case 1 applies

$T(n) = \Theta(n^{\log_b a})$ when $f(n) = O(n^{\log_b a-\varepsilon})$

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

yt

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

→ $a=8$ $b=2$ $f(n)=n^2$

$T(n) = n^{\log_b a}u(n) = n^3 u(n)$

$= n^3 \times 1 = O(n^3)$

Page No.
Date

3 $T(n) = T(n/2) + c$

→ $a=1 \quad b=2 \quad f(n)=c \qquad T(n) = n^{\log_b a} u(n)$

$T(n) = n^{\log_2 1} u(n) = n^0 u(n) = u(n)$

$R(n) = \dfrac{f(n)}{n^{\log_b a}} = (\log_2 n)^0 \cdot c$

y' $\dfrac{(\log_2 n)^{0+1}}{0+1} = (\log_2 n) \cdot c = \log_2 n \cdot c = O(\log_2 n)$

4 $T(n) = 8T(n/2) + 1000n^2$ apply master theorem on it.

→ $T(n) = 8T(n/2) + 1000n^2 \qquad a=8 \quad b=2 \quad f(n) = 1000n^2$

$a \geq 1 \quad b > 1 \quad \log_b a = \log_2 8 = 3$

$f(n) = O(n^{\log_b a - \varepsilon})$ — 1st case applied

$1000n^2 = O(n^{3-1}) = O(n^2)$

$T(n) = \Theta(n^3)$ as $T(n) = \Theta(n^{\log_b a})$

5 $T(n) = 2T(n/2) + 10n$ solve the recurrance using master method

→ $a=2 \quad b=2 \quad f(n) = 10n \qquad \log_b a = \log_2 2 = 1$

$f(n) = \Theta(n^{\log_b a} \log^k n)$ — Case 2 applied

$10n = \Theta(n \log^k n) = \Theta(n) \quad T(n) = \Theta(n^{\log_b a} \log^{k+1} n) = \Theta(n \log n)$

6 Solve the recurrance relation :-

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

→ Compare with $T(n) = aT(n/b) + f(n)$ with $a \geq 1$ and $b > 1$

$a=2 \quad b=2 \quad f(n) = n^2 \quad \log_b a = \log_2 2 = 1$

Put all the values in $f(n) = \Omega(n^{\log_b a + \varepsilon})$

$n^2 = \Omega(n^{1+\varepsilon}) \qquad \varepsilon = 1$ — 1st cond^n

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

$2 \times \left(\frac{n}{2}\right)^2 \leq cn^2 \longrightarrow \frac{1}{2} n^2 \leq cn^2$

$c = 1/2$ It follows $T(n) = \Theta(f(n)) = \Theta(n^2)$

- Iteration Method / Recursive Tree Method

It means to expand the recurrance and express it as ~~a summation~~ a summation of terms of n and initial condition.

Recursion Tree : iteration method which is in the form of a tree, where each level's nodes are expanded. 2 term = root
Each root and child represents cost of single subproblem
Each level costs' sum = pre-level costs. divide till you get 1.
Pre-level costs' sum = total cost.

1 Consider the Recurrance $T(n) = \begin{cases} 1 & \text{if } n=1 \\ 2T(n-1) & \text{if } n>1 \end{cases}$

$$T(n) = 2T(n-1) = 2[2T(n-2)] = 2^2 T(n-2)$$
$$= 4[2T(n-3)] = 2^3 T(n-3) = 2^4 T(n-4)$$

} From substitution

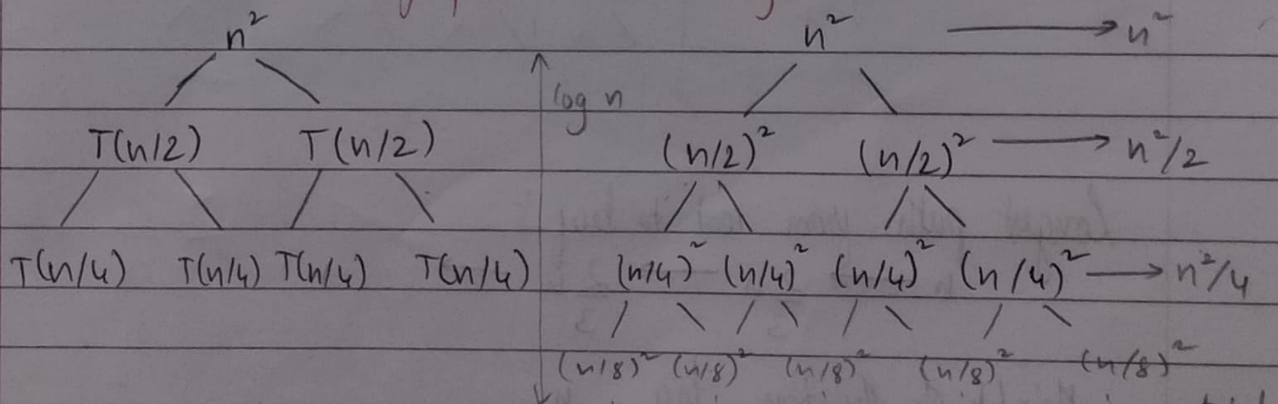
Repeat the procedure i times

$$T(n) = 2^i \overbrace{T(n-i)}^{\text{terminate}} \qquad n-i=1 \rightarrow i = n-1$$
$$T(n) = 2^{n-1} T(n-n+1) = 2^{n-1} \cdot 1 = 2^{n-1}$$

2 $T(n) = T(n-1)+1$ and $T(1) = \theta(1)$ Substitution

$$T(n) = T(n-1)+1 = T(n-2)+1+1 = T(n-2)+2$$
$$= T(n-3)+3 \qquad k = n-1$$
$$T(n-k) = T(1) = \theta(1) \rightarrow T(n) = \theta(1)+(n-1) = 1+n-1 = n = \theta(n)$$

3 Consider $T(n) = 2T(n/2) + n^2$
We have to obtain asymptotic bound using recursion tree method.



total $\theta(n^2)$

$$T(n) = n^2 + n^2/2 + n^2/4 + \ldots \quad \log n \text{ times}$$
$$\le n^2 \sum_{i=0}^{\infty} 1/2^i$$
$$\cancel{n^2/(1-1/2)}$$
$$T(n) \le n^2 (1/(1-1/2))$$
$$\le 2n^2$$
$$T(n) = \theta(n^2)$$

GP sum = $\frac{1}{1-r}$

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

$\frac{n}{2^i} = 1 \longrightarrow \log n = i \log_2 2$
$i = \log n$

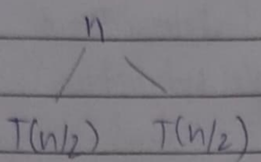
$\frac{1}{1-r}$ = GP sum $= a(1-r^n)/(1-r)$

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

Page No.
Date

4 $T(n) = 4T(n/2) + n$: obtain the asymptotic bound using recursion tree

→

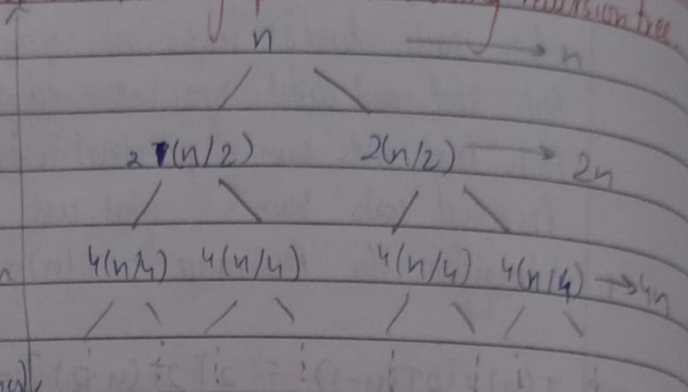


level :: $4^i \times \frac{n}{2^i} \rightarrow \frac{n}{2^i} = 1$

$n + 2n + 4n + \dots \log n$ times

$= n\,(1 + 2 + 4 + \dots + \log_2 n \text{ times})$

$= n\,\frac{2^{\log_2 n} - 1}{2-1} = \frac{n(n-1)}{1} = n^2 - n = \theta(n^2)$



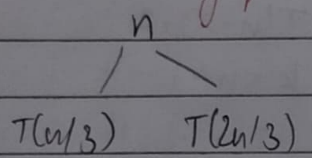
$\theta(n^2)$

5 Consider the following recurrance

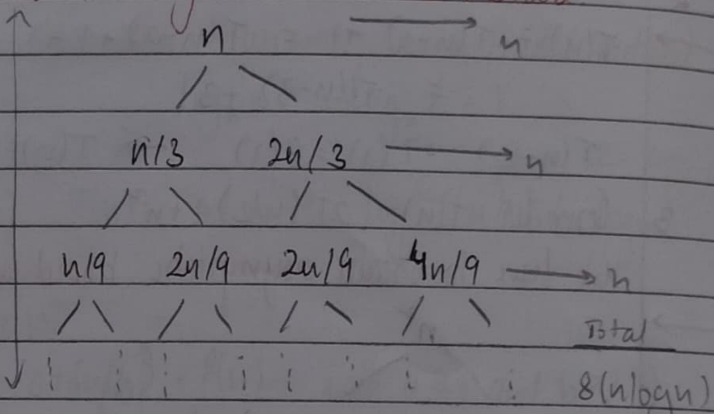
$T(n) = T(n/3) + T(2n/3) + n$

Obtain the asymptotic bound using recursion tree method

→



$\log_{3/2} n$



Longest path from root to leaf :-

$n \longrightarrow \frac{2}{3}n \longrightarrow \frac{2}{3}n \longrightarrow \dots 1$

$\frac{n}{3^i}$: level :

$\frac{n}{3^i} = 1$

Height of the tree $\log_{3/2} n$

$T(n) = n + n + n + \dots \log_{3/2} n$ times

$= \theta(n \log n)$