SER-501 ASSIGNMENT-1

ASU ID: 1211274841
SNEHA VIDHYASHEKAR

1. Use mathematical induction to show that the solution to the recurrence.

$$T(n) = \begin{cases} 2 & \text{if } n = 2 \\ 2T\left(\frac{n}{2}\right) + n & n = 2^k,\ k > 1 \end{cases}$$

is $T(n) = n\log n$.

Step 1 : Base step

If $n = 2$,

then $T(2) = 2\log 2 = 2$

Thus, $T(2) = 2\log 2 = 2$ $\therefore$ $T(n)$ is true for $n = 2$.

Step 2 : Hypothesis step

Assume,

$T(n) = n\log n$ is true if $n = 2^k$ for some integer $k > 1$

Step 3 : Induction step

If $n = 2^{k+1}$, then

$T\left(2^{k+1}\right)$

$= 2T\left(2^{k+1}/2\right) + 2^{k+1}$

$= 2T\left(\frac{2^k \cdot 2^1}{2}\right) + 2^{k+1}$

$$= 2T\left(\frac{2^K \cdot 2}{2}\right) + 2^{K+1}$$

$$= 2T(2^K) + 2^{K+1}$$

$$= 2\left[2^K \log 2^K\right] + 2^{K+1}$$

W.K.T, $2(T(2^K)) = 2^K \log 2^K$

$$= 2^{K+1} \log 2^K + 2^{K+1}$$

$$= 2^{K+1}\left[\log 2^K + 1\right]$$

$$= 2^{K+1}\left[\log 2^K + \log_2 2\right] \quad (\because \log_2 2 = 1)$$

$$T(2^{K+1}) = 2^{K+1}(\log 2^{K+1}) \quad \left[\log(ab) = \log a + \log b\right]$$

Hence proved that the solution is

$T(n) = n \log n$ for the given recurrence.

2. Indicate for each pair of expressions (A, B) whether A is big O, little o, big Ω, little omega or Θ of B.

Assume, $K >= 1$, $\epsilon > 0$ and $C > 1$ are constants.

Justify your answer with proper reasoning.

2
(a)

$A = f(n) = \log^k n$

$B = g(n) = n^{\epsilon}$

Given,

$k >= 1$, $\epsilon > 0$ and $c > 1$ are constants.

Now,

$f(n) = \log^k n \Rightarrow \lim_{n\to\infty} f(n) = \infty$

$g(n) = n^{\epsilon} \Rightarrow \lim_{n\to\infty} (n^{\epsilon}) = \infty$

Hence, by using L'Hopital's rule, which is

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{\infty}{\infty} \text{ then,}$$

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

$$\Rightarrow \lim_{n\to\infty} \frac{\log^k n}{n^{\epsilon}} = \lim_{n\to\infty} \frac{\frac{d}{dn}(\log^k n)}{\frac{d}{dn}(n^{\epsilon})}$$

$$= \lim_{n\to\infty} \frac{k \log^{k-1}(n)(1/n)}{\epsilon . n^{(\epsilon-1)}}$$

$$= \lim_{n\to\infty} \frac{k \log^{k-1} . n}{\epsilon . n^{\epsilon} . n^{-1} . n^{1}}$$

$$= \lim_{n\to\infty} \frac{k \log^{k-1}(n)}{\epsilon . n^{\epsilon}}$$

By differentiating again,

$$\Rightarrow \lim_{n\to\infty} \frac{K \log^{K-1} n}{\epsilon . n^{\epsilon}} = \lim_{n\to\infty} \frac{K(K-1)\log^{K-2}(n)(1/n)}{\epsilon.\epsilon.n^{\epsilon-1}}$$

$$= \lim_{n\to\infty} \frac{K(K-1)\log^{(K-2)}(n)}{\epsilon^2 . n^{\epsilon}}$$

By differentiating the expression K-times,

$$\Rightarrow \lim_{n\to\infty} \frac{\log^K . n}{n^{\epsilon}} = \lim_{n\to\infty} \frac{[K(K-1)(K-2)\ldots\ldots 1]\log^{(K-K)} n}{\epsilon^K . n^{\epsilon}}$$

$$= \lim_{n\to\infty}\left[\frac{K!}{\epsilon^K . n^{\epsilon}}\right]$$

$$\left[\log^{(K-K)}(n) = \log^0(n) = 1.\right]$$

By substituting $n = \infty$

$$\lim_{n\to\infty} \frac{\log^K n}{n^{\epsilon}} = \frac{K!}{\epsilon^K . \infty^{\epsilon}} = \frac{K!}{\infty} = \underline{\underline{0}}$$

$\therefore$ Rate of growth of $g(n) = (n^{\epsilon})$ $>$ Rate of growth of $f(n) = \log^K n$

Hence, $g(n)$ is the asymptotic upperbound of $f(n)$.

$$\Rightarrow f(n) = O\, g(n)$$

$$\Rightarrow \underline{\underline{\log^K n = O(n^{\epsilon})}}$$

Now, to check for big O

Let, $g(n) = c.f(n)$ where, $c \rightarrow$ constant.

$$n^{\epsilon} = c.\log^{k} n$$

when $n = 1$, $n^{\epsilon} = 1^{\epsilon} = 1 \Rightarrow g(n) = g(1) = 1$

when $n = 4$, $\epsilon = 1$

$$\Rightarrow g(n) = g(4) = 4^{1} = 4$$

Let, $c = 2$, $k = 1$

$$\Rightarrow f(n) = \log^{k} n = \log^{1} 4 = \log(2)^{2} = 2\log 2 = 2$$

L.H.S $= g(n) = \underline{\underline{4}}$

R.H.S $= c.f(n) = c.\log^{k} n$

$= 2.2 = \underline{\underline{4}}$

$\therefore$ $\underline{\underline{\text{LHS} = \text{RHS}}}$

Hence, there is a point where $f(n)$ & $g(n)$ meet.

$$\therefore f(n) = O(g(n))$$

$\therefore$ $A = O(B)$

$A = o(B)$

A is big O of B &

A is little o of B

2(b) $A = f(n) = n^k$

$B = g(n) = c^n$

Given,

$k >= 1$, $\epsilon > 0$, $c > 1$ are constants

since,

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \left(\frac{n^k}{c^n}\right) = \frac{\infty}{\infty}$$

we use L'Hopital's rule,

$$\lim_{n \to \infty} \left(\frac{n^k}{c^n}\right) = \lim_{n \to \infty} \frac{k \cdot n^{k-1}}{c^n \log c} \qquad \left\{\frac{d(c^x)}{dx} = c^x \cdot \log c\right\}$$

By differentiating again,

$$\lim_{n \to \infty} \left(\frac{n^k}{c^n}\right) = \lim_{n \to \infty} \frac{k(k-1)\, n^{k-2}}{c^n (\log c)(\log c)}$$

$$= \lim_{n \to \infty} \frac{k(k-1)\, n^{k-2}}{c^n (\log c)^2}$$

By differentiating the expression k-times,

$$\lim_{n \to \infty} \left(\frac{n^k}{c^n}\right) = \lim_{n \to \infty} \frac{k!\, n^{k-k}}{c^n (\log c)^k} \qquad \left\{n^{k-k} = n^0 = 1\right\}$$

$$= \lim_{n \to \infty} \frac{k!}{c^n (\log c)^k}$$

By substituting $n = \infty$

$$\lim_{n\to\infty}\left(\frac{n^k}{c^n}\right) = \frac{k!}{c^{\infty}(\log c)^k} = \frac{k!}{\infty} = 0$$

Hence,

Rate of growth of $g(n) = c^n$ $>$ Rate of growth of $f(n) = n^k$

$\therefore$ $g(n)$ is the asymptotic upperbound of $f(n)$

$$\Rightarrow f(n) = o\, g(n)$$

$$n^k = o\, g(n)$$

$$\underline{\underline{n^k = o(c^n)}}$$

To check for $g(n) = m.f(n)$ where $m \to$ constant

$$c^n = m.n^k$$

LHS when $c = 2$, $n = 1 \Rightarrow c^n = 2^1 = \underline{\underline{2}}$

RHS when $x = 2$, $n = 1 \Rightarrow x.n^k = 2 \times 1^k = \underline{\underline{2}}$ for $k >= 1$

$\therefore$ $\underline{\underline{\text{LHS} = \text{RHS}}}$

Hence, $f(n)$ and $g(n)$ meets at a point

$\therefore$ $f(n) = O(g(n))$

$$\boxed{\begin{array}{l} \therefore\ A = O(B)\ \& \\ A = o(B) \\ \text{A is big O of B \&} \\ \text{A is little o of B} \end{array}}$$

2(c) $A = f(n) = \sqrt{n}$

$B = g(n) = n\sin n$

$n\sin n$ is an oscillating function.

$\sin n$ oscillates between +1 and -1

→ Hence it does not have a strictly increasing or strictly decreasing values.

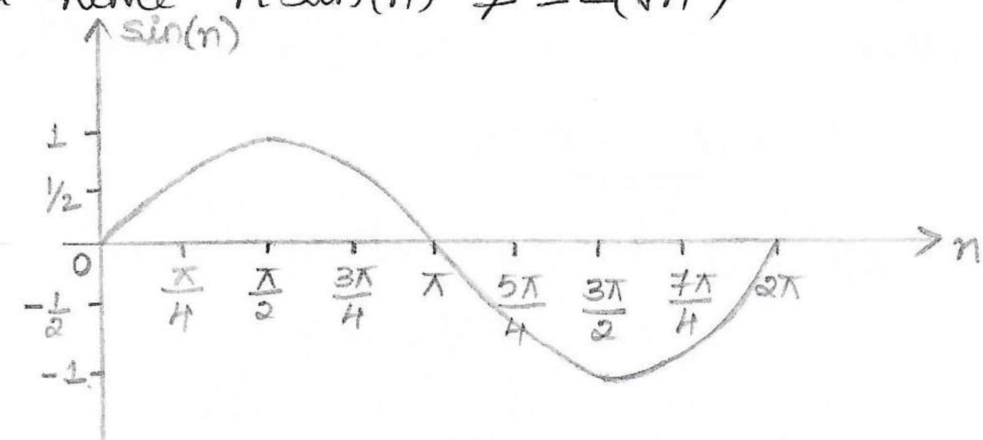
→ $\sin n$ has both positive and negative values.

→ According to definitions of O, o, Ω, ω and Θ, $f(n)$ and $g(n)$ must be greater than zero (cannot have negative values).

So, $\sin n$ function is invalid here.

→ when at its maximum value, $n\sin n > c\sqrt{n}$ and hence $n\sin(n) \neq O(\sqrt{n})$

→ when $\sin n$ is at its minimum, $n\sin n < c\sqrt{n}$ and hence $n\sin(n) \neq \Omega(\sqrt{n})$



→ since $\sin n$ oscillates between positive and negative, <u>$g(n)$ is not a valid function to compare with $f(n)$</u> in terms of Big O, little o, little omega, big Ω and Θ.

2(d) $A = f(n) = 2^n$

$B = g(n) = 2^{n/2}$

Given,

$K >= 1$, $\epsilon > 0$, $C > 1$ are constants.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{\infty}{\infty}$$

By using L' Hopital's rule,

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

$$\Rightarrow \lim_{n\to\infty} \left(\frac{2^n}{2^{n/2}}\right) = \lim_{n\to\infty} \left(2^n \cdot 2^{-\frac{n}{2}}\right) = \lim_{n\to\infty} \left[2^{\left(n-\frac{n}{2}\right)}\right]$$

$$= \lim_{n\to\infty} \left(2^{\frac{n}{2}}\right)$$

From the above expression, when n tends to ∞,

$$\lim_{n\to\infty} \left(2^{n/2}\right) = \frac{2^{\infty/2}}{1} = \underline{\underline{\infty}}$$

Hence,

Rate of growth of $f(n) = 2^n$ (numerator) $>$ Rate of growth of $g(n) = 2^{n/2}$ (denominator)

$$\therefore \boxed{f(n) = \omega(g(n))}$$

Now, to check for $f(n) = x \cdot g(n)$

where $x \rightarrow$ constant

$f(n) = x \cdot g(n)$ where $x \rightarrow$ constant.

i.e, $2^n = x \cdot 2^{n/2}$

when, $n = 0$ & $x = 1$,

L.H.S $= 2^n = 2^0 = \underline{\underline{1}}$

R.H.S $= x \cdot 2^{n/2} = 1 \cdot 2^{0/2} = 1 \cdot 2^0 = \underline{\underline{1}}$

$\underline{\underline{\text{LHS} = \text{RHS}}}$

$\therefore$ There is a point where $f(n)$ & $g(n)$ will meet.

$\therefore f(n) = \Omega(g(n))$

$$\boxed{\begin{array}{c} \therefore A = \Omega(B) \text{ \& } \\ A = \omega(B) \\ A \text{ is big } \Omega \text{ of } B \text{ \& } \\ A \text{ is little } \omega \text{ of } B. \end{array}}$$

2(e) $A = f(n) = n^{\lg c}$

$B = g(n) = c^{\lg n}$

Given,

$K >= 1$, $\epsilon > 0$, $C > 1$ are constants.

By taking log on both A & B,

$A = f(n)$
$= n^{\log c}$
By taking log,
$A = \log c . \log n$

$B = g(n)$
$= c^{\log n}$
By taking log.
$B = \log n . \log c$.

using L'Hopital's rule,

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{f'(n)}{g'(n)}$$

$$= \lim_{n \to \infty} \frac{\lg c . \lg n}{\lg n . \lg c} = \frac{1}{1} = \underline{\underline{1}}$$

since $f(n)$ and $g(n)$ are equal,

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

Hence,

$f(n) = O(g(n))$ &

$f(n) = \Omega(g(n))$ according to O & Ω defn.

(1) $f(n) = \Theta(g(n))$

There exists positive constants c_1, c_2 and n_0 such that

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$$\forall \quad n \geq n_0$$

(2) $f(n) = O(g(n))$

There exists positive constants c and n_0 such that

$$0 \leq f(n) \leq c g(n)$$

$$\forall \ n \geq n_0$$

(3) $f(n) = \Omega(g(n))$

There exists positive constant c and n_0 such that

$$0 \leq c(g(n)) \leq f(n)$$

$$\forall \ n \geq n_0$$

$$\therefore \quad A = O(B) \quad A \text{ is big O of } B$$
$$A = \Omega(B) \quad \& \quad A \text{ is big } \Omega \text{ of } B$$
$$A = \Theta(B) \quad A \text{ is } \Theta \text{ of } B$$

2(f) $A = f(n) = \lg n!$

$B = g(n) = \lg n^n$

W.K.T,

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

$$\Rightarrow \lg(n!) = \lg\left[\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right]$$

By using L'Hopital's rule,

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{f'(n)}{g'(n)}$$

$$\Rightarrow \lim_{n\to\infty}\left[\frac{\lg\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\right)}{\lg n^n}\right] = \lim_{n\to\infty}\frac{\lg(2\pi n)^{0.5} + \lg\left(\frac{n}{e}\right)^n}{n\lg n}$$

$$\Rightarrow \lim_{n\to\infty}\left[\frac{0.5\lg(2\pi n) + n(\lg n - \lg e)}{n\lg n}\right]$$

By differentiating again,

$$\Rightarrow \lim_{n\to\infty}\left[\frac{\frac{\lg(2\pi n)}{2n} + (\lg n - \lg e)}{1 + \lg(n)}\right]$$

By differentiating again,

$$\Rightarrow \lim_{n\to\infty}\left[\frac{\frac{1}{2}\left\{\frac{1}{2\pi n}(2\pi)\,n^{-1}+\lg(2\pi n)(-n^{-2})\right\}+\left(\frac{1}{n}\right)}{1+\frac{1}{n}}\right]$$

$$= \lim_{n\to\infty}\left[\frac{\frac{1-\lg(2\pi n)}{2n^2}+\frac{1}{n}}{\left(\frac{1}{n}\right)}\right] = \lim_{n\to\infty}\left[\frac{1-\lg(2\pi n)}{2n}\right]$$

By differentiating again,

$$\Rightarrow \lim_{n\to\infty}\left[\frac{0-\frac{1}{2\pi n}(2\pi)}{2}+1\right] = \lim_{n\to\infty}\left[\frac{(1/n)}{2}+1\right]$$

$$= \lim_{n\to\infty}\left[\frac{1}{2n}+1\right]$$

By substituting $n=\infty$

$$= \frac{1}{2(\infty)}+1 = \frac{1(0)}{2\times 1}+1 = 0+1 = \underline{\underline{1}}$$

$\therefore$ Rate of growth of $f(n)$ (numerator) $=$ Rate of growth of $g(n)$ (denominator)

i.e, $\lg n! = \Theta \lg n^n$

$$\Rightarrow f(n) = \Theta\, g(n)$$

$$f(n) = O\, g(n)$$

$$f(n) = \Omega\, g(n)$$

$$\boxed{\begin{aligned} \lg(n!) &= \Theta \lg(n^n) \\ \lg(n!) &= O \lg(n^n) \quad \& \\ \lg(n!) &= \Omega \lg(n^n) \end{aligned}}$$

By using Sterlings theorem,

$A = \lg(n!)$ $\qquad$ $B = \lg(n^n)$

According to Sterling's theorem, $\boxed{\lg n! \approx n \lg n}$

we have,

$$B = \lg n^n = n \lg n$$

$\therefore \quad A = B \qquad$ ($\because$ of above theorem)

$$\boxed{\begin{aligned} \therefore \quad A &= \Theta(B) \\ A &= O(B) \\ A &= \Omega(B) \end{aligned}}$$

A	B	O	o	Ω	ω	Θ
$\lg^k n$	n^ϵ	✓	✓	×	×	×
n^k	c^n	✓	✓	×	×	×
$\sqrt{n}$	$n^{\sin n}$	×	×	×	×	×
2^n	$2^{n/2}$	×	×	✓	✓	×
$n^{\lg c}$	$c^{\lg n}$	✓	×	✓	×	✓
$\lg(n!)$	$\lg(n^n)$	✓	×	✓	×	✓

(3) we can express insertion sort as a recursive procedure as follows.
In order to sort $A[1...n]$, we recursively sort $A[1....(n-1)]$ and then insert $A[n]$ into the sorted array $A[1.....(n-1)]$.
write a recurrence for the running time of this recursive version of insertion sort and also solve it.

Let the time taken to sort 'n' numbers be $T(n)$

where $n \rightarrow$ problem size.

considering 2 cases

(i) for $n=1$ (ii) for $n>1$

case (i) :- for $n=1$ (base case)

If $n=1$ $\rightarrow$ problem size $=1$

$\rightarrow$ Hence sorting is not required.

$\rightarrow$ The sub-array for $n=1$, does not exist.

$\therefore$ Sorting takes a constant time. i.e $\Theta(1)$ time

case (ii) :-

If $n>1$ $\rightarrow$ The array is divided into sub-arrays

$\rightarrow$ sorting is done on the array $A[1....(n-1)]$ recursively using insertion sort.

$\rightarrow$ After sorting, $A[n]$ element is inserted into the right position in the above sorted array $A[1.....(n-1)]$

$\rightarrow$ Running time $= T(n-1) + O(n)$

Running time is given by,

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

By substitution,

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

Assume, $T(1) = C$,

$T(n) \leq c_1 n^2$

Proving for $T(n) \leq c_1 n^2$

By using induction,

$$T(n) = T(n-1) + O(n)$$
$$= c_1(n-1)^2 + c_2 n$$
$$= c_1(n^2 - 2n + 1) + c_2 n$$
$$= c_1 n^2 - 2c_1 n + c_1 + c_2 n$$
$$= c_1 n^2 - c_1 - c_2 n - 2c_1 n$$

$\underline{\underline{T(n) \leq c_1 n^2}}$ for $c_1 > c_2$ &

$2c_1 n - c_1 - c_2 n > 0$

$\therefore$ Recurrence for the running time of this recursive version of insertion sort is done by using substitution method.

4. Non-Recursive, linear time algorithm for the max-subarray problem.

Algorithm :-

```
function max_sub_array(A, n)
 // A -> array.
 // n -> size of the array.

begin
    max_sum = 0
    max_end = 0
    start = low   // to start from left end of the
                  // array.

    for i from 1 to n
        max_end = max ( max_end + A[i] , 0 )
        // set current sum = 0, if sum < 0

        if max_end == 0
            start = i+1  // setting the next element index
                         // as a fresh start for subarray
        end if

        max_sum = max ( max_sum, max_end )
        // To find the sum of maximum subarray so far.

    // To set the indices values of the subarray that
    // is maximum.

    if (max_sum == max_end)
        high = i
        low = start
    end if
    return low, high, max_sum
end
```

5) Use a recursion tree to determine a good asymptotic upper bound on the recurrence,

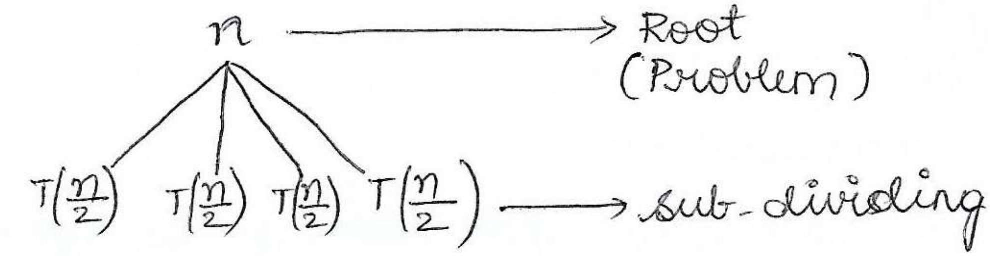
$T(n) = 4T(\frac{n}{2}) + n$. Use substitution method to verify

considering the recurrence function as a tree,

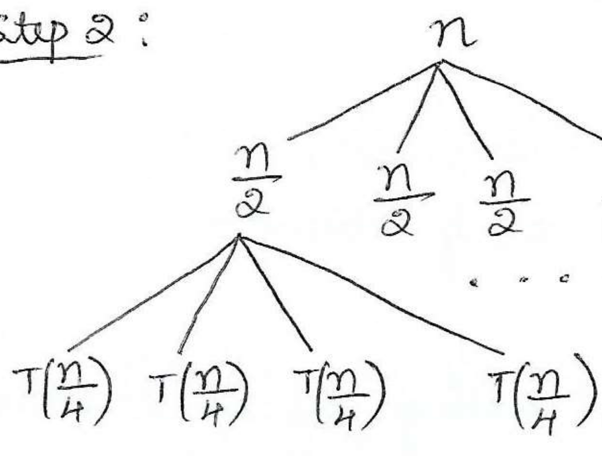
First level of the tree contains $\rightarrow \frac{n}{2}$ nodes.

Recursion tree method :

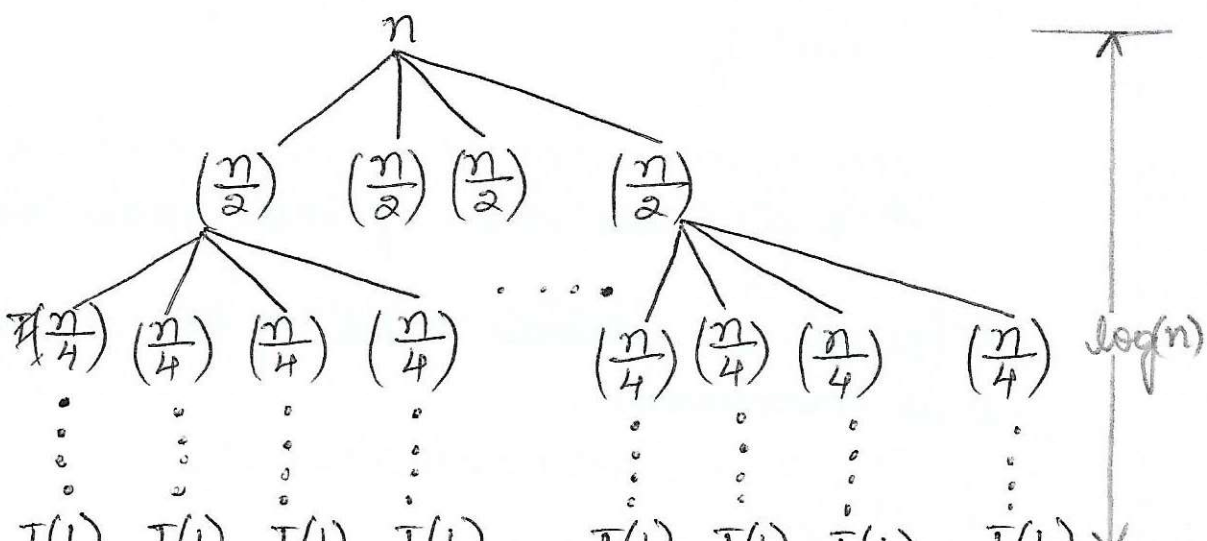
Step 1 :



Step 2 :



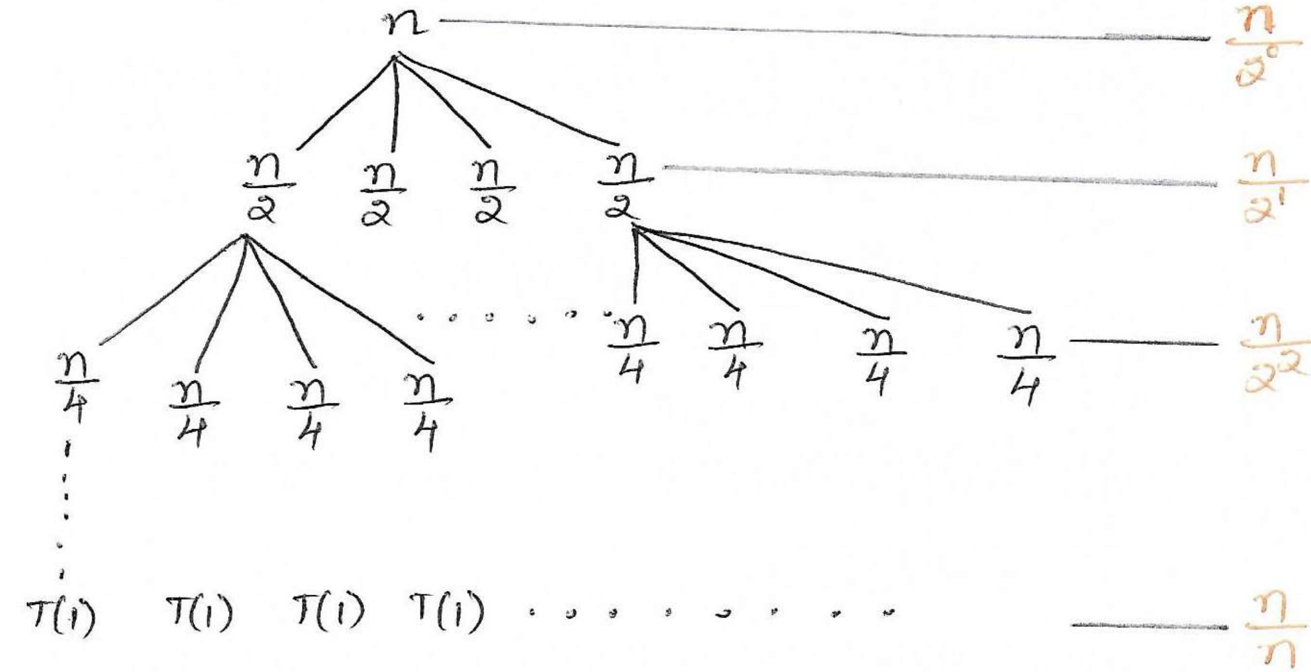
Step 3 :



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

$T(n) \rightarrow$ Total work done @ each level

To calculate the depth of the tree,



$n = 2^k$

From the leaf node,

i.e, $\frac{n}{n} = 1$

The denominator is in the power of 2.

$\Rightarrow n = 2^k$ where $k \rightarrow$ depth of the tree

$$\boxed{k = \log n}$$

$$T(n) = n + 4\left(\frac{n}{2}\right) + 4^2\left(\frac{n}{2^2}\right) + 4^3\left(\frac{n}{2^3}\right) + \dots\dots \log_2 n \text{ terms}$$

$$= n + 2n + 4n + 8n + \dots - + \log_2 n \text{ terms}$$

$$= n\,[1 + 2 + 4 + 8 + \dots\dots]$$

$\hookrightarrow$ $= \log_2 n$ terms.

The above equation represents geometric progression form of

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

$$= n\left[2^{\log_2 n} \cdot 2 - 1\right]$$

$$= n\left[n \times 2 - 1\right]$$

$$= n\left[2n - 1\right]$$

$$= 2n^2 - n$$

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

By substitution method,

Assume, $T(K) \leq cK^2 \quad \forall\ K < n$

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

$$\leq 4c\left(\frac{n}{2}\right)^2 + n$$

$$\leq 4c\left(\frac{n^2}{4}\right) + n$$

$$\leq cn^2 + n$$

$$\underline{\underline{cn^2 + n}} > cn^2$$

Hence, this is failed.

By assuming,

$$T(k) \leq c_1 k^2 - c_2 k \quad \forall \; k < n \;\&\; n > 0$$

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

$$\leq 4\left[c_1\left(\frac{n}{2}\right)^2 - c_2\left(\frac{n}{2}\right)\right] + n$$

$$\leq c_1 n^2 - 2c_2 n + n$$

$$\leq c_1 n^2 - c_2 n - [c_2 n - n]$$

$$T(n) \leq c_1 n^2 - c_2 n \quad \forall \; c_2 > 1 \; (+ve)$$

Hence proved. i.e,

$\therefore \; T(n) = O(n^2)$ where c_1 should be substantially large.

(6) For each of the recurrences,

(i) Give an expression for run time $T(n)$ if the recurrence can be solved with the Master Theorem.

(ii) Otherwise, indicate that the master Theorem does not apply.

Justify the answer with reasoning.

6(a) Master Theorem :-

Let $a \geq 1$ & $b > 1$ be constants

Let $f(n)$ be a function &

$T(n)$ be defined as non-negative integers by the recurrence.

if $T(n) = aT\left(\frac{n}{b}\right) + f(n)$ then,

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & f(n) = O\, n^{\log_b a - \epsilon} \quad \text{case-1} \\ \Theta\left(n^{\log_b a} \log n\right) & f(n) = \Theta\left(n^{\log_b a}\right) \quad \text{case-2} \\ \Theta\left(f(n)\right) & f(n) = \Omega\left(n^{\log_b a + \epsilon}\right) \quad \text{case-3} \\ & \& \\ & a\, f\left(\frac{n}{b}\right) \leq c\, f(n) \ \& \\ & \text{all sufficiently large } n \end{cases}$$

$\epsilon > 0$

$c < 1$

6(a) $T(n) = 2T\left(\frac{n}{2}\right) + n^4$

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

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

$\Rightarrow n^4 = \Omega(n)$

By applying case (3) :

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

$n^4 = \Omega\left(n^{1+\epsilon}\right)$ where $\epsilon = 3$

$\Rightarrow$ $a f\left(\frac{n}{b}\right) \leq c f(n)$

$2 f\left(\frac{n}{2}\right) \leq c f(n)$

$2\left(\frac{n^4}{16}\right) \leq c(n^4)$ $\quad [\because f(n) = n^4]$

$\frac{1}{8} \leq c < 1$

$$\boxed{\therefore \quad T(n) = \theta(n^4)}$$

5(b) $T(n) = T\left(\frac{7n}{10}\right) + n$

$a = 1, \; b = \frac{10}{7}, \; f(n) = n$

$\Rightarrow n^{\log_b a} = n^{\log_{(10/7)} 1} = n^0 = 1$ $\quad [\because \log 1 = 0]$

$\Rightarrow n = \Omega\left(n^{0+\epsilon}\right)$ where $\epsilon = 1$ $\quad (\epsilon > 0)$

$n = \Omega(n^1)$

case 3 applies.

By applying case-3 :

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

$n = \Omega(n^{\epsilon})$; $\epsilon = 1$

$\Rightarrow$ $a f\left(\frac{n}{b}\right) \leq c f(n)$

$1 \cdot f\left(\frac{7n}{10}\right) \leq c f(n)$

$\frac{7}{10} n \leq c(n)$ $[\because f(n) = n]$

$\frac{7}{10} \leq c < 1$ $[c < 1]$

$\Longrightarrow c \geq \frac{7}{10}$ & $c < 1$

$$\boxed{\therefore \quad T(n) = \Theta(n)}$$

5(c) $T(n) = 16 T\left(\frac{n}{4}\right) + n^2$

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

$\Longrightarrow n^{\log_b a} = n^{\log_4 4^2} = n^2$

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

case 2 :- $f(n) = \Theta\left(n^{\log_b a}\right)$

$n^2 = \Theta\left(n^{\log_4 4^2}\right)$

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

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

6(d) $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$

$a=2,\ b=4,\ f(n)=\sqrt{n}=n^{1/2}$

$\Rightarrow n^{\log_b^a} = n^{\log_4^2} = n^{1/2} = \sqrt{n}$

$\Rightarrow \sqrt{n} = \Theta(\sqrt{n})$

<u>case 2</u> applies :

$\rightarrow f(n) = \Theta\left(n^{\log_b^a}\right)$

$\sqrt{n} = \Theta\left(n^{\log_4^2}\right)$

$\sqrt{n} = \Theta\left(n^{\log_{2^2}^2}\right)$

$= \Theta\left(n^{\frac{1}{2}\log_2^2}\right)$

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

$\sqrt{n} = \Theta(\sqrt{n})$

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

6(e) $T(n) = \sqrt{2}\ T\left(\frac{n}{2}\right) + \log n$

$a=\sqrt{2},\ b=2,\ f(n)=\log n$

$\Rightarrow n^{\log_b^a} = n^{\log_2^{\sqrt{2}}} = n^{\log_2^{(2)^{1/2}}} = n^{1/2} = \sqrt{n}$

$\Rightarrow \log(n) = O\left(n^{1/2-\epsilon}\right)$

case 1 applies :

$\rightarrow f(n) = O\left(n^{\log_b a - \epsilon}\right)$

$\log n = O\left(n^{1/2 - \epsilon}\right)$

$$\boxed{\begin{aligned} \therefore\ T(n) &= \Theta(\sqrt{n}) \\ T(n) &= \Theta(n^{1/2}) \end{aligned}}$$

---

6 (f) $T(n) = 64\,T\left(\frac{n}{8}\right) - n^2 \log(n)$

$a = 64$, $b = 8$, $f(n) = -n^2 \log(n)$

According to master theorem,

to apply master theorem,

(i) $a \geq 1$

(ii) $b > 1$

(iii) $f(n)$ – must be positive (non-negative)

If any of the above cases are not satisfied, then master theorem cannot be applied. to that recursion.

Hence, master theorem cannot be applied ~~to~~ for the given recursion as $f(n)$ is negative, although $a \geq 1$ & $b > 1$.

6(g) $T(n) = 2T(\frac{n}{4}) + n^{0.51}$

$$a = 2,\ b = 4,\ f(n) = n^{0.51}$$
$$\approx n^{0.5}$$

$$\Rightarrow n^{\log_b a} = n^{\log_4 2} = n^{\log_{2^2} 2} = n^{1/2} = \sqrt{n} = n^{0.5}$$

$$\Rightarrow n^{0.5} = \Omega(n^{0.5})$$

Case 3 applies:

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

$$n^{0.51} = \Omega\left(n^{1/2 + \epsilon}\right) \quad \text{where } \epsilon = 0.01$$

$$\Rightarrow a f\left(\frac{n}{b}\right) \le c f(n)$$

$$2 f\left(\frac{n}{4}\right) \le c f(n)$$

$$2\left(\frac{n^{0.51}}{4^{0.51}}\right) \le c\left(n^{0.51}\right)$$

$$\frac{2}{4^{0.51}} \le c$$

$$\frac{2}{2^{1.51}} \le c$$

$$\frac{1}{2^{0.02}} \le c < 1$$

$$\boxed{\therefore\ T(n) = \Theta(n^{0.51})}$$

5(h) $T(n) = 16\,T\left(\frac{n}{4}\right) + n!$

$a = 16$, $b = 4$, $f(n) = n!$

$\Rightarrow n^{\log_b a} = n^{\log_4 4^2} = n^2$

$\Rightarrow n! = \Omega(n^{2+\epsilon})$

<u>case 3</u> applies :

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

$n! = \Omega\left(n^{2+\epsilon}\right)$

$\forall\ \epsilon > 0\ \&\ n \geq 4$

[$\because \epsilon > 0$ for which $\Omega(n^{2+\epsilon}) = n!$ $= n(n-1)(n-2)\ldots\ldots 1$]

$\rightarrow a f\left(\frac{n}{b}\right) \leq c f(n)$

$16 f\left(\frac{n}{4}\right) \leq c.f(n)$

$16.\left(\frac{n}{4}\right)! \leq c(n!)$ $\quad \left[c = \frac{1}{2} \quad (c<1)\right]$

For very large value of n, $cn! > 16\left(\frac{n}{4}\right)!$

$\boxed{\therefore\ T(n) = \Theta(n!)}$

5(i) $T(n) = 0.5\,T\left(\frac{n}{2}\right) + \frac{1}{n}$

$a = 0.5$, $b = 2$, $f(n) = \frac{1}{n}$

To apply master Theorem,

(i) $a \geq 1$ (ii) $b > 1$

(iii) $f(n)$ – non negative (positive) [asymptotically positive]

If any of the above cases are not satisfied, then master theorem cannot be applied to that recursion.

Hence, master theorem cannot be applied for the given recursion as $a < 1$.

6(j) $T(n) = 2^n T\left(\frac{n}{2}\right) + n^n$

$a = 2^n, \ b = 2, \ f(n) = n^n$

To apply master theorem,

(i) $a \geq 1$
(ii) $b > 1$
} must be constant

(iii) $f(n)$ — asymptotically positive.

If any of the above cases are not satisfied, then the master Theorem cannot be applied to that recurrence.

$\therefore$ Master Theorem cannot be applied for this recurrence as 'a' is not constant and keeps changing as the value of n changes.