Pratik D. Suryawanshi
1213231238
ASURITE: psuryawa

Problem 1.

To solve given recurrance, we can test it for base condition $n=2$.

$\therefore$ According to given solution

$$T(2) = 2 \cdot \lg 2 = 2$$

Hence given solution holds true for base condition

Assume that this solution also holds true for $n=2^k$

$$\therefore T(n) = 2^k \lg 2^k = k\, 2^k.$$

We now show $T(n+1)$ which is next term of $T(n)$ i.e. for next power of 2.

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

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

$$= 2k\, 2^k + 2^{k+1}$$

$$= (k+1)(2^{k+1})$$

This expression is of required form to show solution holds true

Problem 2.

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

Justification

a. $A = \lg^k n$ $B = n^\epsilon$

Here $\lim_{n \to \infty} \frac{\lg^k n}{n^\epsilon} = 0$

Here n^ϵ is polynomial function and $\lg^k n$ is logarithmic function. Polynomial function grows at much higher rate compared to logarithmic function.

∴ when $n \to \infty$ n^ϵ will be much higher compared to $\lg^k n$. Hence $\lg^k n = o(n^\epsilon)$

∴ Also $\lg^k n = O(n^\epsilon)$ when $n = 4$, $k = 1$, $\epsilon = 1$ because with $c = 1/2$ $f(n) \leq c\, g(n)$ holds true.

b. $A = n^k$

$B = c^n$

Here $\lim_{n \to \infty} \frac{n^k}{c^n} = 0$

since n^k grows polynomially but c^n grows exponentially which is much larger when $n \to \infty$

$\therefore \quad n^k = o(c^n)$

Also for $n=2, k=1, c=2$ we have $2^1 = c\,2^2 \therefore c = 1/2$

hence. if $f(n) = n^k \quad g(n) = c^n$ then $f(n) \leq c\,g(n)$ with $c = 1/2$ holds true.

Hence. $n^k = O(c^n)$ holds true.

c. $A = \sqrt{n} \qquad B = n \sin n$

value of $\sin n$ is in range $[-1, 1]$.

$\therefore$ value of $n \sin n$ is in range $[-n, n]$.

$\therefore$ This range also covers value of $\sqrt{n}$ hence value of $n \sin n$ can be less or greater than $\sqrt{n}$.

$\therefore$ A and B can't be expressed in terms of asymptotic notation.

d. $A = 2^n \quad B = 2^{n/2}$.

Here $\lim_{n\to\infty} \frac{2^n}{2^{n/2}} = \lim_{n\to\infty} 2^{n/2} = \infty$.

Hence $2^n = \omega(2^{n/2})$. because A will always be ahead of B

Now for constant $c = 2$ and $n_0 = 2$ the condition

$f(n) \geq g(n) * c$ is satisfied.

i.e. $2^2 \geq 2^{2/2} * 2$.

$\therefore \quad 2^n = \Omega(2^{n/2})$.

e. $A = n^{\lg c} \quad B = c^{\lg n}$.

We know $a = e^{\log_e a}$

similarly $n^{\lg c} = 2^{\lg(n^{\lg c})} = 2^{\lg(n).\lg c}$.

$c^{\lg n} = 2^{\lg(c^{\lg n})} = 2^{\lg(c).\lg n}$.

Hence A and B are equal. or same functions.

$\therefore f(n) \leq c_1 g(n)$, where $c_1 > 1$

$f(n) \geq c_2 g(n)$, where $c_2 < 1$ and $c_2 > 0$.

Thus $c_2 g(n) \leq f(n) \leq c_1 g(n)$ holds true.

Hence $n^{\lg c} = O(c^{\lg n})$

$n^{\lg c} = \Theta(c^{\lg n})$

$n^{\lg c} = \Omega(c^{\lg n})$.

f. $A = \lg(n!)$ $B = \lg(n^n)$.

By Stirling's approximation we have,

$$\lg(n!) = n \lg n - n$$

$$\therefore \lim_{n\to\infty} \frac{\lg(n!)}{\lg(n^n)} = \lim_{n\to\infty} \frac{n\lg n - n}{n \lg n}$$

$$= \lim_{n\to\infty} 1 - \frac{1}{\lg n} \quad = 1 - \lim_{n\to\infty} \frac{1}{\lg n}$$

$\therefore$ using L'Hopital's rule $\lim_{n\to\infty} \frac{1}{\lg n} = 0$.

$$\therefore \lim_{n\to\infty} \frac{\lg(n!)}{\lg(n^n)} = 1 - 0 = 1.$$

$\therefore$ function A and B progress in same order

$$\therefore \lg(n!) = \Theta \lg(n^n).$$

$$\lg(n!) = O \lg(n^n)$$

$$\lg(n!) = \Omega \lg(n^n).$$

Problem 3.

In insertion sort, to sort array of size n, we sort array of n-1 size i.e. A(n-1). and A[n] is inserted into sorted array.

Hence effort to sort araray A[1...n-1] = T(n-1)

effort to insert A[n] into sorted array = n.

∴ Recurrance relation:

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

solving this using iteration method:

$$T(n) = T(n-1) + n$$
$$= T(n-2) + T(n-1) + n$$
$$= T(n-3) + T(n-2) + T(n-1) + n$$
$$= T(1) + T(2) + T(3) + \ldots + T(n-1) + n$$
$$= 1 + 2 + 3 + \ldots + n$$
$$= \frac{n(n-1)}{2}.$$

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

Problem 4.

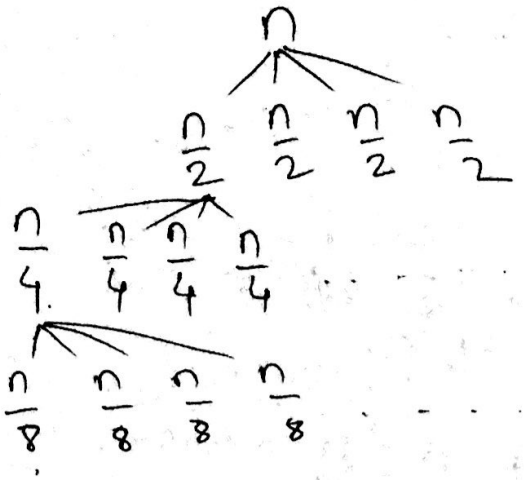
In this problem, algorithm can be developed to know maximum subarray seen so far till A[1...j]. for A[j+1], we can check if A[j+1] increases the existing sum till now. If it does then it can be part of max-sub array. If A[j+1] decreases the sum till now then temporary sum is stored and array is search to find next item greater than sum till now

pseudocode :-

```
max_sum_till_now = A[0].
temp_sum = 0
array_index = 0
while ( array_index is not last index of array ) {
    temp_sum = temp_sum + A[i].
  if ( max_sum_till_now < temp_sum ) {
     max_sum_till_now = temp_sum
  }
  if ( temp_sum < 0) {
       temp_sum = 0
  }
  array_index ++;
}.
return max_sum_till_now.
```

Problem 5

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



level	effort
0	n
1	$4(n/2) = 2n$
2	$16(n/4) = 4n$
3	$64(n/8) = 8n$
h-1	$2^{2h-2}(n/2^{h-1}) = 2^{h-1}n$

∴ At level k, total effort $= 2^{2k} \cdot \left(\frac{n}{2^k}\right) = 2^k n$.

At level $h-1$, total effort $= 2^{2(h-1)}\left[\frac{n}{2^{h-1}}\right] = 2^{h-1} n$.

At level h, total effort = No. of leaf nodes * $T(1)$.

Here in binary tree height = $\log_2$ (No. of nodes)

∴ for given tree, height = $\log_4$ (No. of nodes).

∴ No. of nodes = 4^h.

$$\therefore T(n) = n + 2n + 4n + 8n + \cdots + 2^{(h-1)} n + 4^h T(1).$$

$$= \sum_{i=0}^{h-1} 2^i n + 4^h T(1).$$

Here $h = \lg n$. ..... eqn (1).

$$\therefore T(n) = 4^{\lg n} T(1) + \sum_{i=0}^{h-1} 2^i . n.$$

$$= n^2 T(1) + n [1 + 2 + 4 + 8 + \cdots + 2^{h-1}]$$

Here we have geometric series in form
$S = a + ar + ar^2 + \cdots + ar^{k-1}$ with $a = 1$ and $r = 2$

$\therefore$ Using formula for G.S, $S = a \frac{(r^k - 1)}{r-1} = 1 \cdot \frac{(2^h - 1)}{2-1} = 2^h - 1.$

$$\therefore T(n) = n^2 T(1) + n(2^h - 1)$$
$$= n^2 T(1) + n(n-1) \quad \text{.. putting eqn (1)}$$
$$= c \cdot n^2 - n \quad \text{where } c = T(1) + 1.$$

$\therefore$ Asymptotic upper bound for $T(n)$ is $O(n^2)$.

where $T(n)$ is in $cn^2 - n$ form.

Now we can use substitution method to verify $T(n) = O(n^2)$. is upper bound.

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

$$\leq 4\left(c_1 (n/2)^2 - c_2 (n/2)\right) + n$$

$$= c_1 n^2 - 2c_2 n + n$$

$$= c_1 n^2 - c_2 n - (c_2 n - n)$$

$$T(n) \leq c_1 n^2 - c_2 n \quad \leftarrow \text{desired.}$$

we proved $T(n) = c n^2 - n$ $\therefore$ where $c_2 \geq 1$.

$\therefore$ we have $O(n^2)$ as upper bound.

Problem 6

a) $T(n) = 2T(n/2) + n^4$.

comparing this to master theorem equation,

$f(n) = n^4$

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

Here $f(n)$ grows faster polynomially by n^3 factor ($\epsilon = 3$)

$\therefore f(n) = \Omega(n^{1+\epsilon})$

Now to check regularity.

$a f(n/b) = 2 f(n/2) = 2(n/2)^4 = 1/8\, n^4$.

$\therefore a f(n/b) \leq 1/8\, f(n)$.

Thus regularity holds by checking $c = 1/8 < 1$.

$\therefore$ case 3 of master theorem, $T(n) = \Theta(f(n)) = \Theta(n^4)$.

b) $T(n) = T(7n/10) + n$

$f(n) = n$ $\quad n^{\log_b a} = n^{\log_{\frac{10}{7}} a} = n^{\log_{1.42} 1} = 1 = n^0$.

$f(n)$ polynomially grows faster than $n^{\log_b a}$ by n factor

$\therefore \epsilon = 1$.

checking regularity

$a f(n/b) = f(n/10/7) \leq \frac{7}{10} n$.

$\therefore$ for $c = 0.7 < 1$ regularity holds.

$\therefore$ case 3 of master theorem is satisfied

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

c) $T(n) = 16\,T(n/4) + n^2$

Comparing this to master theorem,

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

$a \geq 1$ and $b > 1$

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

Here $f(n) = n^2$

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

$\therefore$ From case 2 of master theorem,

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

because both $f(n)$ and $n^{\log_b a}$ grow at similar rate.

d) $T(n) = 2\,T(n/4) + \sqrt{n}$.

comparing this to master theorem,

$a = 2 \quad b = 4 \quad f(n) = \sqrt{n}$

since $a \geq 1 \quad b > 1$

$n^{\log_b a} = n^{\log_4 2} = n^{\log_4 4^{1/2}} = n^{1/2} = \sqrt{n}$.

$\therefore f(n) = n^{\log_b a}$.

$\therefore$ since $f(n)$ and $n^{\log_b a}$ grow at similar rate,

case 2 of master theorem is confirmed.

Hence $T(n) = \Theta(\sqrt{n} \log n)$.

e) $T(n) = \sqrt{2}\, T(n/2) + \log n$.

$f(n) = \log n$, $n^{\log_b a} = n^{\log 2^{1/2}} = \sqrt{n}$.

$\log n$ grows polynomially slower than $\sqrt{n}$.

∴ Case 1 of master theorem is confirmed

∴ $T(n) = \Theta(n^{\log_b a}) = \Theta(\sqrt{n})$.

f) $T(n) = 64\, T(n/8) - n^2 \log(n)$.

Here $f(n) = -n^2 \log n$.

$f(n)$ is not asymptotically positive.

Hence master theorem can't be applied.

g) $T(n) = 2\, T(n/4) + n^{0.51}$

$f(n) = n^{0.51}$

$n^{\log_b a} = n^{\log_4 2} = n^{1/2} = n^{0.50}$

∴ $f(n)$ polynomially grows faster than $n^{\log_b a}$ by n^{ϵ} factor $\epsilon = 0.01. > 0$

checking regularity,

$a f(n/b) = 2 \cdot (n/4)^{0.51}$

$\leq \frac{2}{4^{0.51}} n^{0.51} \leq c \cdot n^{0.51}$ where $c = \frac{2}{4^{0.51}} < 1$

∴ Regularity holds

∴ By case 3 of master theorem, $T(n) = \Theta(n^{0.51})$.

h) $T(n) = 16\,T(n/4) + n!$

comparing this with master theorem,

$a = 16 \quad b = 4 \quad f(n) = n!$

since $a \geq 1 \quad b > 1 \quad n^{\log_b a} = n^{\log_4 16} = n^2$

$\therefore f(n) > n^{\log_b a}$ i.e. $f(n)$ grows polynomially faster.

$\therefore$ case 3 of master theorem is confirmed.

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

i) $T(n) = 0.5\,T(n/2) + 1/n.$

comparing this with master theorem

$a = 0.5 \quad b = 2 \quad f(n) = 1/n$

since $a \geq 1$ condition does not satisfies,

master theorem does not apply for this equation

j) $T(n) = 2^n\,T(n/2) + n^n.$

comparing this with master theorem,

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

since $a \geq 1$, $b > 1$ condition is required to apply master theorem and $a = 2^n$ which is not constant, master theorem does not apply for this equation.