CSE 221 Assignment 01

① Time Complexity:-

a. Ascending order of growth:-

$\log(\log(n))$, $\log(n)$, $\sqrt{n}$, n, $n\log n$, $n^{3/2}$, n^2, $n^2\log n$, n^3, 2^n, $n!$, e^{n+1}

b

(i) $n^2 + 15n - 3 = \theta(n^2)$

let, $g(n) = n^2$;

Now,

$$c_1\, g(n) \le n^2 + 15n - 3 \le c_2\, g(n)$$

$$c_1 n^2 \le n^2 + 15n - 3 \le c_2 n^2 \quad \text{--- (i)}$$

Dividing all sides by n^2;

$$c_1 \le 1 + 15/n - 3/n^2 \le c_2 \quad \text{--- (ii)}$$

For $c_1 = 1$ and $c_2 \ge 13$ the inequity (ii) holds

$\therefore f(n) = n^2 + 15n - 3 \in \theta(n^2)$

(ii) $2n^3 + 5n^2 + 8n + 13 \in \theta(n^3)$

$$c_1 n^3 \le 2n^3 + 5n^2 + 8n + 13 \le c_2 n^3 \quad \text{--- (i)}$$

$$c_1 \le 2 + 5/n + 8/n^2 + 13/n^3 \le c_2 \quad \text{--- (ii)}$$

For $c_1 \le 2$ and $c_2 \ge 48$ the inequility (ii) holds,

$\therefore f(n) = 2n^3 + 5n^2 + 8n + 13 \in \theta(n^3)$

(iii) $5 + 2\sin(n) = \Theta(1)$

$$c_1 \leq 5 + 2\sin(n) \leq c_2 \quad \text{—(i)}$$

let, $f(n) = 5 + 2\sin(n)$

maxima of $f(n) = 7$

minima of $f(n) = 3$

thus, for $c_1 \leq 3$ and $c_2 \geq 7$, the inequility (i) holds

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

(iv) $5n^2 + 2n - 3 = \Omega(n^2)$

let, $f(n) = 5n^2 + 2n - 3$

and $g(n) = n^2$

for ~~the~~ $f(n) \in \Omega(n^2)$ to be true,

$$0 \leq c\,g(n) \leq f(n) \quad \text{—(i) needs to be true}$$

Now, $0 \leq c\,n^2 \leq 5n^2 + 2n - 3$

$$0 \leq c \leq 5 + 2/n - 3/n^2 \quad \text{—(ii)}$$

when n is ~~too~~ large enough. the ineauility always holds for $c \leq 5$

therefore, $f(n) \in \Omega(n^2)$

(v) Show that, $(n^2+8)(n+1) = O(n^3)$

let, $f(n) = (n^2+8)(n+1)$

$$= n^3 + n^2 + 8n + 8$$

and $g(n) = n^3$

For our claim to valid, the below inequility must hold

$$0 \le f(n) \le c\, g(n) \quad \text{— (I)}$$

$$0 \le n^3 + n^2 + 8n + 8 \le c\, n^3$$

$$0 \le 1 + \frac{1}{n} + \frac{8}{n^2} + \frac{8}{n^3} \le c \quad \text{— (II)}$$

For ~~the~~ ~~c≥18~~ $c \ge 18$, the above inequility will always hold.

$\therefore$ $f(n) \in O(n^3)$

~~(vi)~~ (vi) show that, x^3+5x is not $O(x^2)$

let, $f(x) = x^3+5x$

$g(x) = x^2$

~~For the~~ For $f(x) \in O(g(x))$ to be true,

$0 \leq x^3+5x \leq cx^2$ —(I) must hold

now,

$0 \leq x + 5/x \leq c$ —(II)

~~So~~ It's evident that, no matter how big 'c' is, Langer value of x will always surpass c.

therefore, $f(x) \notin O(x^2)$

(vii) Prove that, $2n^2+3n+6 \neq \theta(n^3)$

Here,

$0 \leq c_1 n^3 \leq 2n^2+3n+6 \leq c_2 n^3$

$0 \leq c_1 \leq 2/n + 3/n^2 + 6/n^3 \leq c_2$

Now, the right side inequality will hold for $c_2 \geq 11$. But the left part will not. when n reaches infinity. $2/n + 3/n^2 + 6/n^3$ will approach zero. in that case $0 \nleq c_1 \nleq 2/n + 3/n^2 + 6/n^3$. Beacuse c_1

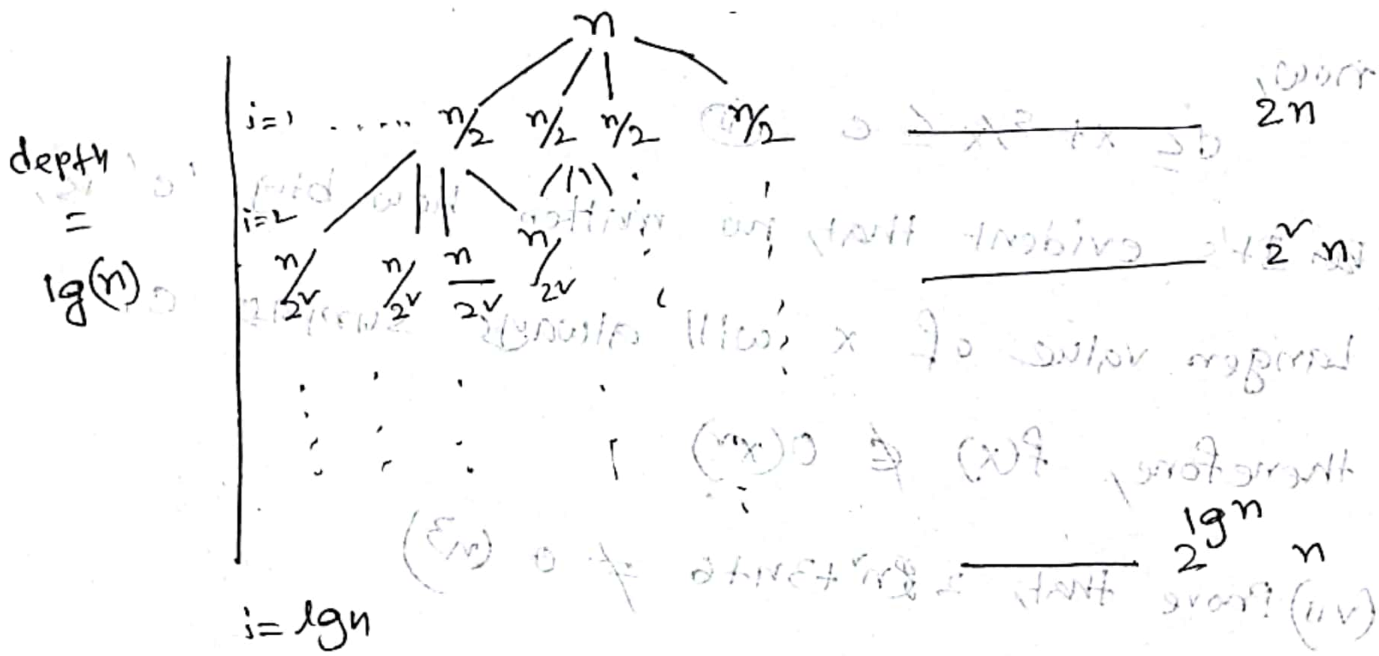
has to be a ~~positibe~~ positive integer.

Hence, $2n^2+3n+6 \notin \Theta(n^3)$

(viii)

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

Expanding the recursion tree:



$$T(n) = \sum_{i=0}^{\lg n} 2^i n$$

$$= n\left(\frac{2^{\lg n}-1}{2-1}\right) \quad \text{[Geometric Series formula]}$$

$$= n(n^{\lg 2} - 1)$$

$$= n(n-1)$$

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

(ix) $T(n) = 2T(n/2) + n^3 = O(n^3)$

Expanding the recursion tree ....

$i=0$ $\qquad n^3 \qquad\qquad n^3$

$i=1$ $\qquad \left(\frac{n}{2}\right)^3 \quad \left(\frac{n}{2}\right)^3 \qquad — \quad \frac{1}{4} n^3$

$i=2$ $\qquad \left(\frac{n}{4}\right)^3 \quad (n/4)^3 \quad (n/4)^3 \quad (n/4)^3 \qquad — \quad (1/4)^2 n^3$

$i=3$ $\qquad \left(\frac{n}{8}\right)^3 \quad \left(\frac{n}{8}\right)^3 \quad \cdots \qquad — \quad (1/4)^3 n^3$

$\vdots \qquad\qquad \vdots$

$i=\lg n \qquad\qquad (1/4)^{\lg n} n^3$

$$T(n) = \sum_{i=0}^{\lg n} n^3 (1/4)^{i}$$

$$= n^3 \sum_{i=0}^{\lg n} (1/4)^{i}$$

$$= n^3 \left[\frac{1-(1/4)^{\lg n}}{1-1/4}\right]$$

$$= \frac{4n^3}{3}\left[1-(1/4)^{\lg n}\right]$$

$$= \frac{4n^3}{3} - \frac{4n^3}{3}\left(\frac{n^{\lg 1}}{n^{\lg 4}}\right)$$

$$= \frac{4n^3}{3} - 0$$

$$= O(n^3)$$

(x) $T(n) = T(n/4) + T(5n/8) + n = O(n)$

~~Let's~~

the first term of the equation is $T(n/4)$

$a=1$

$b=4$

then, $n^{\log_b a} = n^{\log_4 1} = n^0 = 1$

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

so runtime $= \Theta(n^{\log_b a}) =$ ~~$\Theta(1)$~~ ~~$\Theta(1)$~~ ~~$\Theta(1)$~~ $\Theta(1)$

the second term $= T(5n/8) + n$

$a=1$

$b=8/5$

$n^{\log_b a} = n^0 = 1$

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

so runtime $= \Theta(f(n)) = \Theta(n)$

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

(xi) $T(n) = T(n/3) + T(4n/9) + n = O(n)$

For ~~the~~ term $T(n/3)$

$a = 1$

$b = 3$

$n^{\log_b a} = n^{\log_3 1} = 1$

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

$\therefore$ runtime $= \Theta(n^{\log_b a}) = \Theta(1)$

For, $T(4n/9) + n$

$a = 1,$

$b = 9/4$

$n^{\log_b a} = n^{\log_{9/4} 1} = n^0 = 1$

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

$\therefore$ runtime $= \Theta(f(n)) = \Theta(n)$

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

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

using Substitution,

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

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

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

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

$= T(n-5) + 5n - 10$

$= T(n-6) + 6n - 15$

$\vdots$

The constant values in this equation forms a triangular series, where the kth term $= k(k+1)/2$.

so in ith step,

$\cancel{T(n) = T(n-1) + i}$

$T(n) = T(n-i+1) + (i+1)n - \frac{i(i+1)}{2}$ — ①

when, n reaches the base case,

$T(n-i+1) = 1$ [constant time]

and let's say, $n-i+1 = 1$

$\therefore n = i$

plugging into equation ①

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

$= O(n^2)$

e.

1. the outer loop runs through $\lg n$ elements. the inner loop goes through n elements when $i=n$.

so time complexity = ~~$\theta(n\lg$~~ $\theta(n\lg n)$

2. Here,

Step	P

$i=0$ $\quad P=3^{2^0} = 3$

$i=1$ $\quad P=3^{2^1} = 9$

$i=2$ $\quad P=3^{2^2} = 81$

$i=3$ $\quad P=3^{2^3} = 6651$

$\cdots$

$i=k$ $\quad P=3^{2^k}$

Now,

$$3^{2^k} = n$$

$$2^k \lg(3) = \lg n$$

$$2^k = \frac{1}{\lg(3)} \times \lg n$$

$$k\lg(2) = \lg(1/\lg(3)) + \lg(\lg(n))$$

$$k = \text{constant} \times \lg\lg n$$

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

3. ~~time complexity~~ Run_time for the outer loop = n

and the inner loop = $\lg n$.

so time complexity = $\Theta(n \lg n)$

4. Here, i runs n^2 steps

and in the worst case, j runs n^2 steps

$\therefore T(n) = \Theta(n^2 \cdot n^2) = \Theta(n^4)$

Searching

a. Divide and conquer is an algorithmic techinque to recurisevly break a problem into smaller subproblems until the ~~reach~~ reach a base case ~~and~~ then ~~the~~ solving those subproblems and combine them to ~~get~~ solve the original problem,

b. ~~In~~ ~~Ternary~~ Binary Search is better than ternary search. Because, in the worst case binary search does $2\lg n$ comparisons as opposed to ternary search. which does $4 \log_3 n$ comparisons. Since, binary search does less comparsions, binary ~~seach~~ search is better than ternary search.

c. Recurrence relations for ternary search

$T(n) = \cancel{T(n/3)}\ T(n/3) + 1$, here $T(1) = 1$

Here, $a = 1$

$b = 4$

$n^{\log} \quad n^{\log_b a} = n^0 = 1$

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

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

$\therefore T(n) = \Theta$

c. ternary search takes $\log_3 n$ time to terminate the recursion and ~~θ(~~ constant time to combine. thus the time complexity $= \Theta(\log_3 n)$

c.

The recurrence relation for ternary search

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

expanding the recursion tree:

Depth $= \log_3 n$

$T(n)$ ——— 1

$T(n/3)$ ——— 1

$T(n/3^2)$ ——— 1

$T(n/3^3)$ ——— 1

⋮

~~at ith step~~

at final step 1

$$\therefore T(n) = \sum_{i=0}^{\log_3 n} (1)$$

$$= \log_3 n$$

$$\therefore T(n) = \Theta(\log_3 n)$$

d. Yes. the code is logically connect. Since the code divides the search space ~~in two~~ in half in every recursive call total steps to terminate the recursion would be $\log_2 n$.

thus time complexity = $O(\lg n)$