Data Structure

Assignment no. 1

Saksham Khajuria
IT
11912058

Q1 T.C = $O(n^2)$
↳ It takes 5 seconds for $n = 10$

lets say $kn^2 = 5$

$k(100) = 5$

$k = \frac{5}{100}$

for $n = 50$

↳ Time $= k(50)(50)$

$= \frac{5}{\cancel{100}} \times 50 \times \cancel{50}^{25}$

$= 125$ seconds

∴ Approximately it will take 125 seconds

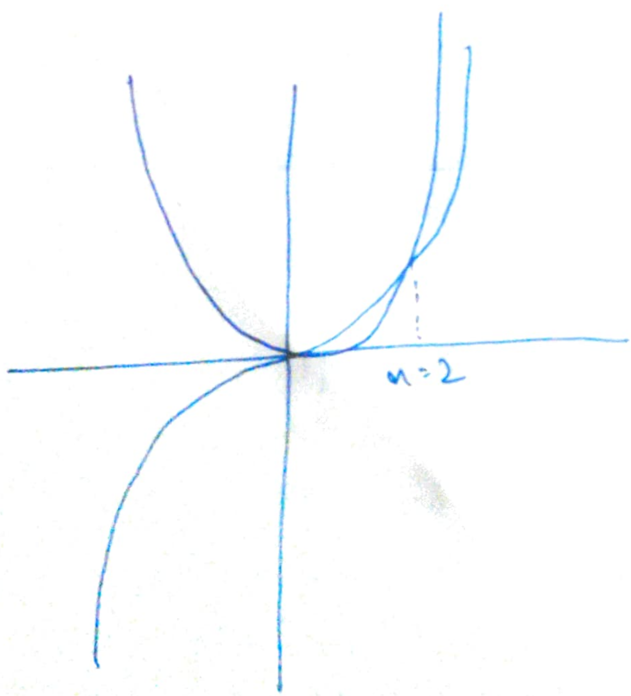
Q2 $T(A(n)) = n^3$

$TB(n) = 2n^2$

so, $n^3 = 2n^2$

$n^2(n-2) = 0$

$n = 2$



∴ after n=2, they will start to deviate

Q 3 Use of limit rule to check $n2^n$ is in $O(4^n)$

$$\lim_{n\to\infty} \frac{4^n}{n2^n}$$

$$\lim_{n\to\infty} \frac{2^{2n}}{n2^n}$$

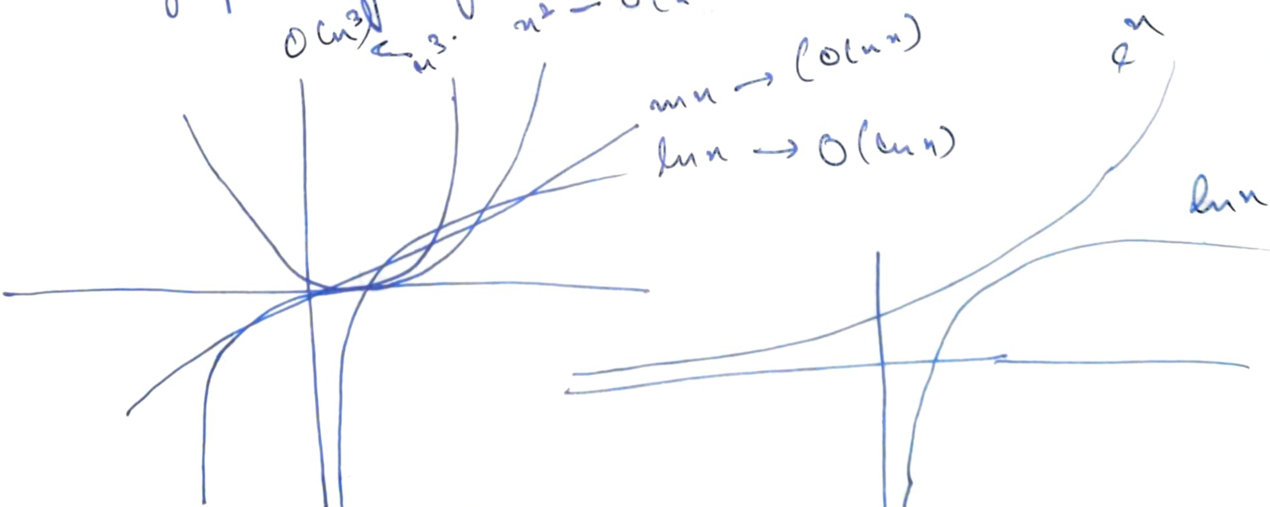
$$\lim_{n\to\infty} \frac{2^n}{n}$$

Applying L'Hospital rule as $\frac{\to\infty}{\to\infty}$ form, indeterminant

$$\lim_{n\to\infty} \frac{n2^{n-1}}{1} = \infty$$

$\therefore$ It means for large value of n 4^n is much bigger than $n2^n$ so $n2^n$ is in bounds of $O(4^n$

Q 4 graph of $\log n$, let us assume base to e



$\therefore$ here we can clearly see that for large value of n (input) log graph has slowest growth

we can also see by differentiating them

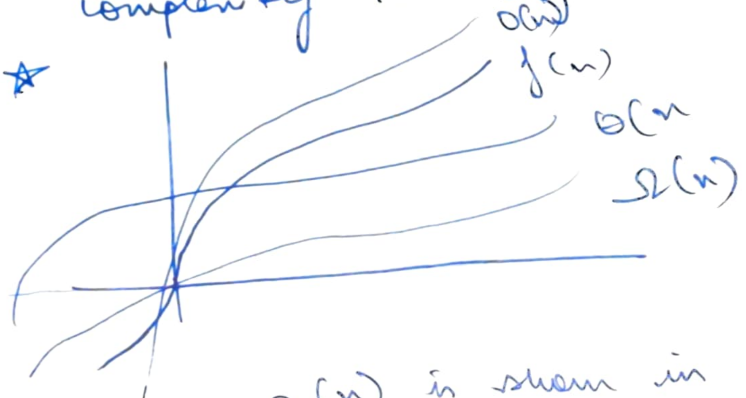
$n^3 \to 3n^2$, $n^2 \to 2n$, $mn \to m$, $e^n \to e^n$, $\log n \to 1/n$

Q5 a) Θ and O

average base time complexity (theta)

$\Theta \rightarrow$ average base time complexity that a f^n / it is the average of time complexity that a algorithm performs. eg $\rightarrow$ Quick sort has avg. T.C $= \Theta(n \log n$ but has worst case time complexity i.e Big $O = O(n^2$

☆



☆ $\Omega(n)$ is shown in sorting algorithm like bubble sort when array already sorted fully

Q6 a) $n^4 + \log n + 17$

$$\lim_{n\to\infty} n^4 + \log n + 17$$

$$\lim_{n\to\infty} n^4 \left(1 + \cancel{\frac{\log n}{n^4}}^{\,0} + \cancel{\frac{17}{n^4}}^{\,0}\right)$$

$$\lim_{n\to\infty} \frac{\log n \rightarrow \infty}{n^4 \rightarrow \infty}$$

$$\lim_{n\to\infty} \frac{(n^4)}{\cancel{\ }}$$

$\therefore$ applying L'Hospital rule

$$\lim_{n\to\infty} \frac{\frac{1}{n}}{4n^3} = \frac{1}{4n^4} = 0$$

so for large n f^n behaves like n^4 so, it has $O(n^4)$

Q 7

a) k = 1
while k ≤ n
k = k × 1
End while

T. C = $O(n)$

b) for i = 1 to n-1 do
for j = i + 1 to n do
Swap
End for
End for

$n + (n-1) + (n-2) \ldots\ldots 2 + 1 - 1$

$$\frac{n(n+1)}{2} \cong O(n^2)$$

Q 8 algorithm T.C ⇒ $O(n^2)$ it takes t_1 time for n input

for 2n inputs $(2n)^2 = 4n^2$
kn inputs $(kn)^2 = 4n^2$

for $k = \sqrt{2}$
if input size increase by 1.42 times
than that the time running algorithm
will become twice i.e $2t_1$

Q9 $T_A = 100^n$

$T_B = n^4$

$$\lim_{n\to\infty} \frac{100^n}{n^4} \xrightarrow{\infty}_{\infty} \Rightarrow \frac{n\,100^{n-1}}{4n^2} \Rightarrow (n-1)\frac{10^{n-2}}{8n}$$

apply L'Hospital rule!

$$\Rightarrow \frac{100^{n-2}}{8\left(\frac{1}{1-\frac{1}{n}}\right)} \Rightarrow \infty$$

So, T_A is much worse performing than T_B at larger values of n

Q10 show that $n \lg n \in (\lg(n!))$

$$\lim_{n\to\infty} \frac{\lg(n!)}{n \lg n} \Rightarrow \frac{\lg n}{n \log n} + \frac{\lg n-1}{n \lg n} + \ldots = 0$$

$$\lim_{n\to\infty} \frac{n \lg n}{\lg n!} \Rightarrow \frac{\lg n^n}{\lg n!} \Rightarrow \frac{n^n}{n!} \Rightarrow \frac{n \cdot n \cdot n \cdots}{n(n-1)\cdots 1}$$

$$= \frac{1}{\left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\cdots}$$

$= 1$

Hence we proved that

$n \log n \simeq \Theta(\lg(n!))$

Q11 a) $2^{n-1} + 4^{n+1}$

$$\frac{2^n}{2} + 2^{n+2}$$

$$2^n\left(\frac{1}{2} + 4 \cdot 2^n\right)$$

$\simeq 2^n$] for $n \to \infty$

$2^{2n} = 4^n \Rightarrow O(4^n$

b) $(n^2+b)^8$

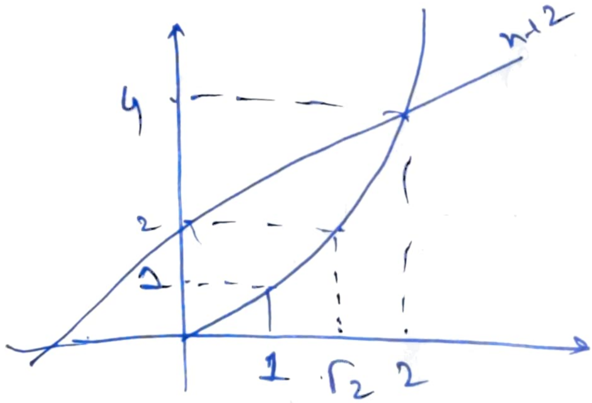
$$\lim_{n\to\infty} (n^2)^8 \left(1+\cancel{\frac{b}{n^2}}^{\to 0}\right)^8$$

$$= n^{16}$$

$$\Rightarrow \simeq O(n^{16}$$

Q $T_A = n^2$

$T_A = n+2$



breaking point is $n=2$ after $n=2$ n^2 grows much faster than $n+2$

Q i) $f(n) = 3n^3 = O(n^3)$

ii) $f(n) = n^3 + 2n^2 + n = O(n^3)$

iii) $f(n) = 2n^3 + 13\lg n / 7n^2 \Rightarrow \frac{2n}{7} + \frac{3\lg n}{n^2} = O(n$

for(i=1; i<=n, i++)

{

} $= O(n)$

☆ for (i = n-1, i ≥ 1 ; i /= 2

⇒ $O(\log_2 n)$

☆ for (i = 0 , i < n, i++) → $O(n)$

for (j = 0; j < n, j++) → $O(n)$

$O(n^2)$

☆ for (i = 0, i < n , i++) → $O(n)$

for (i = n-1, i ≥ 1 . i = $\frac{1}{2}$) → $O(\log_2 n)$

⇒ $O(n \log_2 n)$

☆ for (i = 0 ; i < n ; i ++) → $O(n)$

for (j = 0 ; j < n ; j ++) → $O(n)$

for (k = 0; k < n; k++) → $O(n)$

$O(n^3$

☆ for (i = $\frac{n}{2}$; i < n ; i ++)

for (j = 1 ; j < n/2 ; j ++)

for (k = 1; k <= n ; k = k*2)

$O(n^2 \log_2 n)$

☆ for (i = $\frac{n}{2}$, i <= n , i ++)

for (j = 1 ; j < n; j = 2*j)

for (k = 1 ; k ≤ n ; k = 2*k)

$O(n(\log_2 n)^2)$