11912074

AYUSHI GUPTA _ IT

PAGE NO.
DATE : / /

Assignment -

Qu(1) $O(n^2)$ for $n = 10 \longrightarrow 5$ sec

$100 \longrightarrow 5$

$2500 \longrightarrow \frac{5}{100} \times 2500 = 125$ sec

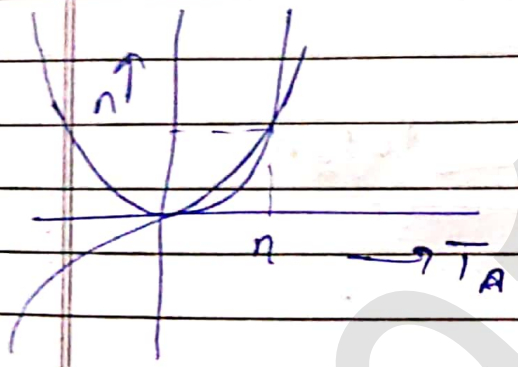
Qu.(2) $T_A(n) = n^3$

$T_B(n) = 2n^2$

$n^3 - 2n^2 = 0$

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

$n = 2$



Qu (3) $f(n) = n2^n$ $O(4^n)$

$g(n) = 4^n$

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

$$\lim_{n \to \infty} \frac{n2^n}{4^n} = n\left(\frac{1}{2}\right)^n$$

$$\lim_{n \to \infty} \frac{n}{2^n} \Rightarrow \lim_{n \to \infty} = 0$$

$O(f(n)) < O(g(n))$

so the $n2^n$ is in $O(4^n)$

PAGE NO.
DATE : / /

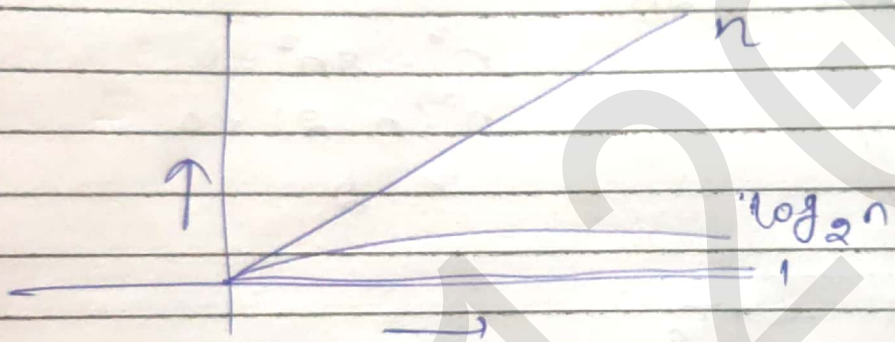
Qu (4) $O(\log_3 n)$

an algorithm's running time depends on its input size(n).

ex.

$\log_3 27$ is 3

Logarithm function gets slightly slower as n grows. when n doubles the running time increased by constant



Qu (5) Worst case (O) is the function which performs the maximum numbers of steps on input data of size n.

ex. if we want to sort an array of integers in ascending order then the worst case would be that array is already sorted in descending order.

Average case (O) is used when we refer to average number of input to

PAGE NO.
DATE : / /

test the algorithm. It is the average resources used taking into all possible inputs

Qu (5) (b) O() → big-oh notation
is an asymptotic notation notation for the worst case. It provides us with an asymptotic upper bound for the growth rate of the running of an algorithm.

$T(n)$ is $O(gn)$

Ω - is an asymptotic notation for the best case. It provides us with an asymptotic lower bound for the growth rate of the running of an algorithm.

$$T(n) \geq cg(n) \text{ for } (n > n_0)$$

Qu (6) $n^4 + 10g(n) + 17$
growth rate of constant

$O(n^4) > O(\log n) > O(1)$

so

$n^4 + 10g(n)$
$= n^4$ $\Phi O(n^4)$

PAGE NO.
DATE : / /

as we increase the input size the growth rate of logarithm function lesser than the polynomial function.

Qu. (a)

```
k = 1          → O(1)
while (k <= n) → O(n)
{                          let assume
    k = k + 1 ; → (O(n)    N = 5
}
```

N = 5 — number of time of looping
5 — 5

→ O(1) + O(n) + (O(n)
→ O(n) time complexity

```
for (i = 1 to n-1) → O(n)
{
    for (j = i+1 to n) → O(n-1)
    {
        swap → c(n-1)
    }
}
```

for n = 5

Time = O(n) + O(n-1) + c(n-1)
= O(n)

PAGE NO.
DATE : / /

Qu(8) quardatic algorithm

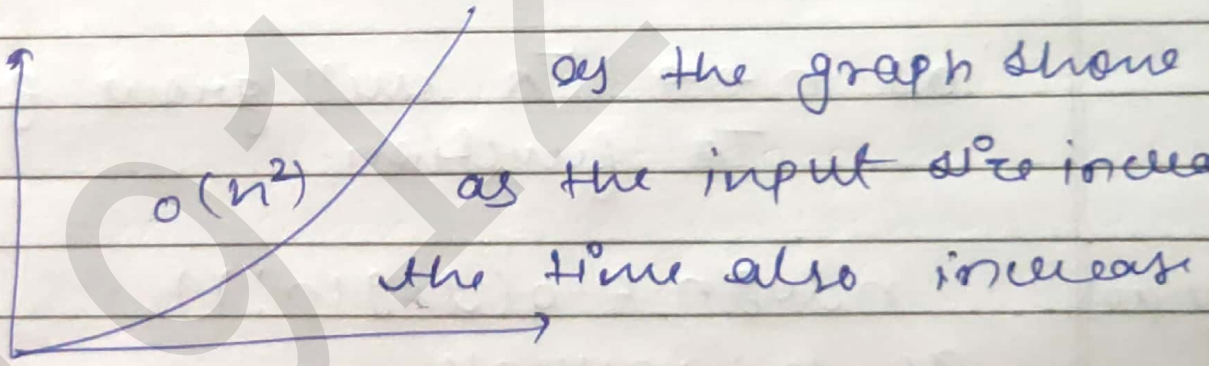
$f(n) = n^2 + 2n + 4$

as we see the growth of rate

$O(n^2) > O(n) > O(1)$

$f(n) \approx O(n^2)$

as increase the input size double then the running time will become larger



Qu(9) $T_A = 100^n$
$T_B = n^4$

$$\lim_{n \to \infty} \frac{T_A}{T_B} = \frac{100^n}{n^4}$$

$$= \frac{(10)^{2n}}{n^4}$$

PAGE NO.
DATE : / /

$100^n < n^4$

$T_B(n) < T_A(n)$

$n^4 < 100^n$

$\lim_{n \to \infty} \frac{100^n}{n^4}$ (hospital rule

$\frac{n(100)^{n-1}}{4n^3} = \infty$

Thus $f(n)$ is in $O(g(n))$

Thus n^4 is a upper bound of $(100)^n$

as $n \to \infty$ T_A will grow faster than T_B function.

Qu. (10) $n \log n \in O(\log(n!))$

as n = increase

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

$\log(n!) = \log(n \times (n-1)!)$

$= \log n + \log((n-1)!)$

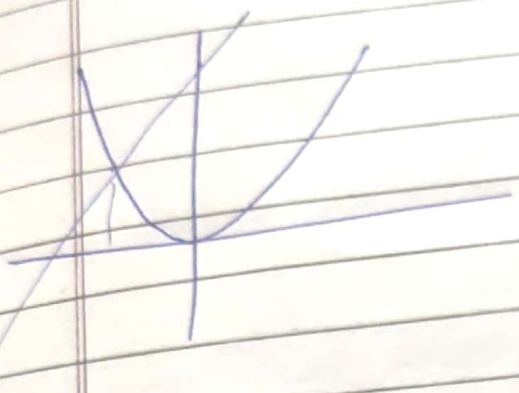
$n \log n \to \underbrace{\log 1 + 2\log 2 + 3\log 3 \dots}_{n \log n}$

PAGE NO.
DATE : / /

Qu(12) $2^{n-1} + 4^{n-1}$

$T_A = n^2$

$T_B = n + 2$



$n^2 = n + 2$

$n^2 - n - 2 = 0$

$n^2 - 2n + n - 2 = 0$

$n = 2$

Breaking point $n = 2$

Qu(14) total = 0 → O(1)

for (i = 0 to n+1) → O(n+1)

{ sum = sum + A[i] → C(O(n))

return sum; → C(O(n))

}

Tsum = O(1) + O(n+1) + C(O(n+1)) + C(

→ O(n)

Qu(15) int count = 0; → O(1)

for (i = 0 — i < N; i++) → O(n)

{

for (j = 0 j < i j++) → ?(O(

{

count++ → C(O(n)

}