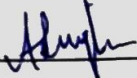


TUTORIAL - 1

NAME	:	Avantika Singh
ROLL NO.	:	01
SECTION	:	CST - SPL 1
STUDENT ID	:	20022854
DATE	:	10 March '22
SIGNATURE	:	

TUTORIAL-1

Q1. ASYMPTOTIC NOTATIONS:-

Asymptotic Notation give us an idea about how good a given algorithm is, as compared to some other algorithm.

There are 3 types of widely used Asymptotic Notations:-

- i) Big O (O)
- ii) Big Omega (Ω)
- iii) Big Theta (Θ)

i) Big O Notation:-

This notation defines an upper bound of an algorithm, it bounds a funcⁿ only from above.

ii) Omega Notation (Ω):-

Just as Big O notation provides an asymptotic upper bound on a funcⁿ, Ω notation provides an asymptotic lower bound.

iii) Theta Notation (Θ):

This notation bounds a funcⁿ from above & below, so it defines exact asymptotic behavior.

eg:- $f(n) = \sum_{i=1}^n i \times 2^i$

→ ~~$T(n)$~~ $T(n) = \Omega(2^n)$

→ $T(n) = O(n2^n)$

→ $T(n) = \Theta(n2^n)$

Q2. What should be Time complexity of -

```
for (i = 1 to n) {
    i = i * 2;
}
```

$$i = 1, 2, 4, 8, \dots, n$$

$$t_k = 2^{k-1}$$

$$n = 2^{k-1}$$

$$\log_2 n = k-1$$

$$k = \log_2 n + 1$$

$$O(k) = O(\log_2 n + 1)$$

$$\boxed{\text{Ans } T(n) = O(\log_2 n)}$$

Q3. $T(n) = \begin{cases} 3T(n-1) & \text{if } n > 0 \\ 1 & \text{otherwise} \end{cases}$

$$T(0) = 1$$

$$T(n) = 3T(n-1) \quad \text{--- (1)}$$

Put $n = n-1$ in eq (1)

$$T(n-1) = 3T(n-2) \quad \text{--- (2)}$$

Put in eq (1)

$$T(n) = 3^2 T(n-2) \quad \text{--- (3)}$$

Put $n = n-2$ in eq (1)

$$T(n-2) = 3T(n-3) \quad \text{--- (4)}$$

Put in eq (3)

$$T(n) = 3^3 T(n-3) \quad \text{--- (5)}$$

for some constant k ,

$$T(n) = 3^k T(n-k) \quad \text{--- (6)}$$

Put $n = k \cdot 0$

$$k = n/3$$

$$T(n) = 3^{n-1} \cdot T(0)$$

$$T(n) = O(3^{n-1})$$



Ans $T(n) = O(3^n)$

Q4. $T(n) = \begin{cases} 2T(n-1) - 1 & \text{if } n > 0 \\ 1 & \text{otherwise} \end{cases}$

$$T(n) = 2T(n-1) - 1 \quad \text{--- (1)}$$

Put $n = n-1$

$$T(n-1) = 2T(n-2) - 1 \quad \text{--- (2)}$$

Put in eq (1)

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

$$T(n) = 4T(n-2) - 2 - 1 \quad \text{--- (3)}$$

Put $n = n-2$ in eq (1)

$$T(n-2) = 2T(n-3) - 1 \quad \text{--- (4)}$$

Put in eq (3)

$$T(n) = 4[2T(n-3) - 1] - 2 - 1$$

$$T(n) = 8T(n-3) - 4 - 2 - 1 \quad \text{--- (5)}$$

for some constant k ,

$$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} - \dots - 1 \quad \text{--- (6)}$$

Put $n = k = 0 \Rightarrow n = k$

$$T(n) = 2^n T(0) - 2^{n-1} - 2^{n-2} - \dots - 1$$

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

$a = 2^{n-1}$ $x = +1/2$

$$S = \frac{2^n [(+1/2)^n - 1]}{+1/2 - 1} = 2^n [2^{-n} - 1]$$

$$T(n) = 2^n - 2^n [2^{-n} - 1] = 2^n [1 - 2^{-n} + 1] = 2^n [2 - 2^{-n}]$$

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

Q5.

```
int i = 1, s = 1;
while (s <= n) {
    i++;
    s += i;
    printf("#");
}
```

$i = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad \dots$

$$s = 1 + 3 + 6 + 10 + 15 + \dots + T_{n-1} + T_n \quad \text{--- (1)}$$

$$s = 1 + 3 + 6 + 10 + \dots + T_{n-1} + T_n \quad \text{--- (2)}$$

Sub eq (2) from eq (1)

$$0 = 1 + 2 + 3 + 4 + \dots + n - T_n$$

$$T_k = 1 + 2 + 3 + 4 + \dots + k$$

$$T_k = \frac{k(k+1)}{2}$$

for k iterations,

$$1 + 2 + 3 + \dots + k \leq n$$

$$\frac{k(k+1)}{2} \leq n$$

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

$$O(k^2) \leq n$$

$$k = O(\sqrt{n})$$

$$\boxed{T(n) = O(\sqrt{n})}$$

Q6. void function(int n) {
 int i, count=0;
 for (i=1; i*i ≤ n; i++)
 count++;

3

$$\therefore i^2 \leq n$$

$$i \leq \sqrt{n}$$

$$i = 1, 2, 3, 4, \dots, \sqrt{n}$$

$$\sum_{i=1}^{\sqrt{n}} 1+2+3+4+\dots+\sqrt{n}$$

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

$$T(n) = \frac{n\sqrt{n}}{2}$$

$$\text{du } T(n) = O(n)$$

Q7. void function(int n) {
 int i, j, k, count=0;
 for (i=n/2; i ≤ n; i++)
 for (j=1; j ≤ n; j=j*2)
 for (k=1; k ≤ n; k=k*2)
 count++;

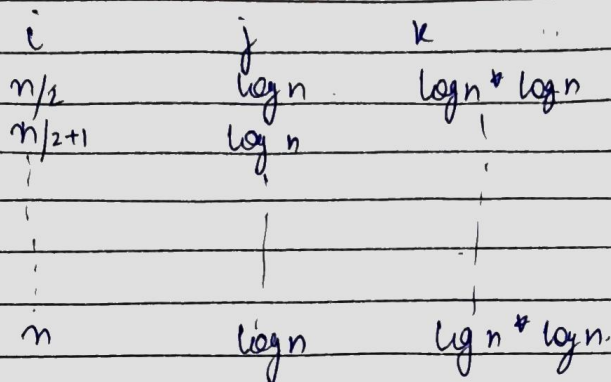
}

$$\text{for } k=k^2$$

$$k = 1, 2, 4, 8, \dots, n$$

$$a=1, x=2 \Rightarrow \boxed{k = \log_2 n}$$

2



$$T(n) = O\left(\frac{n}{2} * \log n * \log n\right)$$

$$\text{Ans } \boxed{T(n) = O(n \log^2 n)}$$

Q8. `function (int n) {`

`if (n==1) return;` // O(1)

`for (i=1 to n) {` —— n time

`for (j=1 to n) {` —— n time

`print(" * ");`

`}`

`}`

`function (n/3);` // T(n/3)

`}`

$$\Rightarrow T(n) = T(n/3) + n^2$$

Using Master's Method,

$$a=1, b=3, f(n)=n^2$$

$$c = \log_3 1 = 0$$

$$n^c = 1 > n^2$$

$$\text{Ans } \boxed{T(n) = O(n^2)}$$

Q9. void function (int n) {
 for (i=1 to n) {
 for (j=1; j<=n; j=j+1)
 printf("%d * %d");
 }
}

for $i=1$, $j = 1, 2, 3, 4, \dots, n = n$
 for $i=2$, $j = 1, 3, 5, \dots, n = n/2$
 for $i=3$, $j = 1, 4, 7, \dots, n = n/3$
 ⋮

for $i=n$, $j = 1$ — (1)

$$= \sum_{j=n}^1 n + \frac{n}{2} + \frac{n}{3} + \dots + 1$$

$$= \sum_{j=n}^1 n \left(1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} \right)$$

$$= \sum_{j=n}^1 n (\log n)$$

$$\therefore T(n) = O(n \log n)$$

Q10. for functions n^k & c^n , what is asymptotic relationship b/w these func?

Assume that $k \geq 1$ & $c > 1$ are constants.

find the value of c & n_0 for which relation holds.

Ans.

Relation b/w n^k & c^n is

$$n^k = O(c^n)$$

$$\text{as } n^k \leq a c^n$$

$$\forall n \geq n_0 \text{ \& some constant } a > 0.$$

$$\text{for } n_0 = 1$$

$$c = 2.$$

$$\Rightarrow 1^k \leq a 2^1$$

$$n_0 = 1 \text{ \& } c = 2$$

Ans