TUTORIAL - 1

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

PAGE No.
DATE

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 func^n only from above.

ii) **Omega Notation (Ω)** :-

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

iii) **Theta Notation (Θ)** :

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

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

→ $T(n) = \Omega(2^n)$
→ $T(n) = O(n2^n)$
→ $T(n) = \Theta(n2^n)$

PAGE No
DATE

Q2. What should be Time complexity of -

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

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

$t_k = ar^{k-1}$

$n = 2^{k-1}$

$\log_2 n = k-1$

$k = \log_2 n + 1$

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

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

Q3. $T(n) = \{3T(n-1)$ if $n>0$, otherwise $1\}$

$T(0) = 1$

$T(n) = 3T(n-1)$ —— (1)

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

$T(n-1) = 3T(n-2)$ —— (2)

Put in eq (1)

$T(n) = 3^2 T(n-2)$ —— (3)

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

$T(n-2) = 3T(n-3)$ —— (4)

Put in eq (3)

$T(n) = 3^3 T(n-3)$ —— (5)

for some constant k,

$T(n) = 3^k T(n-k)$ —— (6)

Put $n-k = 0$

$k = n$

PAGE No.
DATE

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

$\boxed{T(n) = O(3^{n})}$

$\Downarrow$

Ans $\boxed{T(n) = O(3^{n})}$

Q4. $T(n) = \{2T(n-1) - 1$ if $n > 0$, otherwise $1\}$

$T(n) = 2T(n-1) - 1$ —— (1)

Put $n = n-1$

$T(n-1) = 2T(n-2) - 1$ —— (2)

Put in eq (1)

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

$T(n) = 4T(n-2) - 2 - 1$ —— (3)

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

$T(n-2) = 2T(n-3) - 1$ —— (4)

Put in eq (3)

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

$T(n) = 8T(n-3) - 4 - 2 - 1$ —— (5)

for some constant k,

$T(n) = 2^{k}T(n-k) - 2^{k-1} - 2^{k-2} - \ldots - 1$ —— (6)

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

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

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

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

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

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

$= 2^{n}[2 - 2^{-n}]$

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

Q5.

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

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

$s = 1+3+6+10+15+\ldots + T_n$ — ①

$s = 1+3+6+10+\ldots + T_{n-1} + T_n$ — ②

Sub eq ② from eq ①

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

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

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

for k iterations,

$1+2+3+\ldots + k <= n$

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

$\frac{k^2+k}{2} <= n$

$O(k^2) <= n$

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

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

PAGE No.
DATE

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

$\because \; i^2 <= n$

$i <= \sqrt{n}$

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

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

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

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

$\therefore \; \boxed{T(n) = O(n)}$

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++;
}
```

for $k = k^2$

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

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

i	j	k
$n/2$	$\log n$	$\log n * \log n$
$n/2+1$	$\log n$	
⋮	⋮	⋮
n	$\log n$	$\lg n * \log n$

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

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

Q8.

```
function (int n) {
        if (n==1) return;                    // O(1)
        for (i=1 to n){                      ——— n time
            for (j=1 to n){
                printf(" * ");               ——— n time
        }
    }
    function (n-3);                          // T(n/3)
}
```

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

Using Master's Method,

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

$c = \log_3 1 = 0$

$n^c = 1 > n^2$

$$\therefore \boxed{T(n) = \Theta(n^2)}$$

Q. void function (int n) {

```
    for (i=1 to n) {
        for (j=1; j<=n; j=j+i)
            printf("*");
    }
}
```

for $i=1$, $j \Rightarrow 1, 2, 3, 4, \ldots\ldots n = n$

for $i=2$, $j = 1, 3, 5, \ldots\ldots n = n/2$

for $i=3$, $j = 1, 4, 7, \ldots\ldots n = n/3$

$\vdots$

for $i=n$, $j=1$ $= 1$

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

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

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

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

PAGE No.
DATE

Q10. For functions, n^k & c^n, what is asymptotic relationship b/w these func?
Assume that $k >= 1$ & $c > 1$ are constants.
find the value of c & n_0 for which relation holds.

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

$$\boxed{n^k = O(c^n)}$$

as $n^k \leq ac^n$

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

for $n_0 = 1$

$c = 2$.

$\Rightarrow 1^k \leq a2^1$

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