Assignment - 1

① The efficiency of an algorithm depends on the amount of time, storage and other resources required to execute the algorithm. The efficiency is measured with the help of asymptotic notations

Types:

(i) **Big-O Notation:** It represents the upper bounds of the running time of an algorithm.

Eg:- Time complexity of selection sort - $O(n^2)$

Time complexity of merge sort - $O(n \log n)$

(ii) **Theta notation:** It encloses the function from above and below. Since it represents the upper and lower bound of the running algo., it is used for avg. case complexity of an algorithm.

Eg- bubble sort : $\Theta(n^2)$.

(iii) **Omega notation:** Omega notation represents the lower bound of the running algo. So, it provides the best case time complexity.

Eg - bubble sort: $\Omega(n)$.

② for (i=1 to n)

{

$i = i*2$

}

$i = 1, 2, 4, 8, 16, \ldots n$

$a = 1$

$r = 2$

③ $n = ar^{k-1}$

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

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

$2n = 2^k$

taking log

$\log_2 2n = \log_2 2^k$

$k = \log_2 2 + \log_2 n$

$k = \log_2 n + 2$

$\Rightarrow T.C = O(\log n)$

③ $T(n) = \begin{cases} 3T(n-1), & n > 0 \\ 1 & \end{cases}$

$T(0) = 1$

$T(n) = 3T(n-1)$ —①

put $n = n-1$ in ①

$\Rightarrow T(n-1) = 3T[(n-1)-1]$

put value of $T(n-1)$ in eqn ①

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

$T(n) = 9T(n-2)$ —②

put $n = n-2$ in ①

$T(n-2) = 3T(n-2-1)$

put $T(n-2)$ in (2)

$\Rightarrow T(n) = 9[3T(n-3)]$

$\Rightarrow T(n) = 27[T(n-3)]$

$T(n) = 3^k T(n-k)$ — (3)

$n-k = 1$

$\Rightarrow k = n-1$

put value of k in (3)

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

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

$\Rightarrow O(3^n)$

(4) $T(n) = \begin{cases} 2T(n-1)-1, & n>0 \\ 1 \end{cases}$

$T(0) = 1$

$T(n) = 2T(n-1)-1$ — (1)

put $n = n-1$ in (1)

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

put in (1)

$\Rightarrow T(n) = 4T(n-1-1)-1$. — (2)

put $n = n-2$ in (1)

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

$T(n) = 8\,T(n-2-1) - 1$ — 3

put $n-3$ in 1

$T(n-3) = 2\,T(n-3-1) - 1$

put in 3

$T(n) = 16\,T(n-3-1) - 1$ — 4

$T(n) = 2^K\,T(n-k) - 1$ — 5

$(n-k) = 1$

~~=> n =~~.

$\Rightarrow k = n-1$

put in 5

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

$O(2^n)$

5
```
int i=1, s=1
while(s<=n){
    i++
    s=s+i
    pf("#")
}
```

$s = 1 + 3 + 5 + 7 + \cdots\ n$

$a = 1$

$d = 2$

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

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

$\Rightarrow O(n)$

Ans

(6)

{

```
int i, count =0
for (i=1; i*i <= n; i++)
    count++;
}
```

~~i = 1, 2~~

$O(\sqrt{n})$

(7)

```
{ 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)
```

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

$n = ar^{k-1}$

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

$\Rightarrow n = \frac{2^k}{2}$

$\Rightarrow 2n = 2^k$

$\log_2(2n) = k \log_2 2$

$k = \log_2 2 + \log_2 n$

$k = \log_2 n$ $\quad n^k$ is $O(c^n)$

$O(\log n)$

Ans

(8)

```
if (n==1)
  return;
  for (i=1 to n) n
    { for (j=1 to n) (n+1)
      {        pf(" ")    n*(n+1) +1
      }
    }
    function (n-3)  (log n)
}
```

$\Rightarrow$ T.C $= O(n^2 \log n)$

Ans

(9)

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

$\Rightarrow n^2 + n + 1$

Ignoring the lower

order terms and constants

$O(n^2)$

Ans

~~10~~ $\cancel{n^k}$, $\cancel{k >= 1}$

$\cancel{c^n}$, $\cancel{c > 1}$

$\cancel{n^k = O(c}$

10 n^k, $k >= 1$

c^n, $c > 1$

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

Ans