Date.

Page No.

Assignment-1

Q1 What do you understand by Asymptotic notations with examples

Ans It is a mathematical notation that describes the behaviour of a function as its input size approaches infinity. It is used to analyze the time and space Complexity of algorithms

Different types of Asymptotic notations are

(1) Big oh(O)- It is used to describe the upper bound of the running time or space Complexity of an algo. It's worst case senerio of algo.

$f(n) = O(g(n))$

Iff $f(n) \leq cg(n)$

$\forall n \geq n_0$, some constant $(c > 0)$

(2) Big omega (Ω). It is used to describe the lower bound of the running time or space Complexity of an algorithm. It is the best Case senerio.

$f(n) = \Omega(g(n))$

Iff $f(n) \geq cg(n)$

$\forall n \geq n_0$, some Constant $(c > 0)$

iii) Theta(Θ): It is used to describe the tight bound of the running time or space Complexity of an algo. Its Avg case Senario.

$$f(n)=\Theta(g(n))$$

If, $c_1 g(n) \le f(n) \le c_2 g(n)$

$\forall n > \max(n_1, n_2)$ some constant $(c_1>0,$ $c_2>0)$

(iv) Smal oh(o): used to describe the strip upper bound of running time space Complexity of an algorithm. It is a more strict Version of Big-O notation.

v) small omega(ω) - describe the strict lower bound of running time or space complexity of an algorithm. It is a more strict version of Big-Omega notati

$$f(n) > c[g(n)]$$

Iff $f(n) > c[g(n)]$

$\forall n > n_0; \forall c > 0$

Que 2 What should be the Time complexity

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

Ans

$$i = \underbrace{1, 2, 4, 8, \ldots n}_{k\ terms}$$

here it is a GP

$a_n = a r^{n-1}$

$n = 1 \cdot 2^{k-1}$ $[\because a_n = n, r = 2, n = k]$

$2n = 2^k$

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

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

~~k log~~ $k = \log(2) + \log_2(n)$

$k = 1 + \log_2(n)$

$$\boxed{\therefore O(\log_2(n))}$$

Que 3 $T(n) = \{3T(n-1)$ if $n>0$, otherwise$\}$.
using forward subs.

Ans

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

$T(0) = 1$

$T(1) = 3T(1-1) = 3T(0) = 3 = 3$

$T(2) = 3T(2-1) = 3T(1) = 3 \cdot 3 = 3^2$

$T(3) = 3T(3-1) = 3T(2) = 3 \cdot 3 \cdot 3 = 3^3$

$\Rightarrow O(3^n)$

Que 4 $T(n) = 2T(n-1) - 1$ if $(n>0)$ otherwise

As here $a = 2$

$b = 1$

$C = \log_2(2) = 1$

$f(n) = 1$

$\therefore n^c = f(n)$

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

Que 5 Time complexity

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

Ans $s = \underbrace{1+2+3+ --- n}_{k \text{ terms}}$

$n = [k+1] \times \frac{1}{2}$

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

$2n = k^2 + k$

Date.

Page No.

$K = \left(-\frac{1}{2} \pm \sqrt{1+4*n}\right)/2$

$\therefore \{O(\sqrt{n})$

Q6 Time complexity of:

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

As

sol $i = 1, 2^2, 3^2, 4^2 \ldots - - K^k$

K^{th} term $= K*K$

k^{th} term $<= n$

$K+K <= n$

$K^2 <= n$

$\boxed{K = \sqrt{n}}$

$\Rightarrow O(\sqrt{n})$

Que] time complexity of 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++;
  }
```

Sol^n Inner most loops

K=1 to n , K=K*2

1, 2, 4, 8, 16 - - - - K term

K th term $= 2^{K-1}$

$2n = 2^K$ -

$\boxed{K = 1 + \log_2 n}$

it means for each value of j this loops runs $1 + \log_2 n$ times

Complexity of middle loop.

j=1 to n; j=j*2;

1, 2, 4, 8, 16, - - - K

$= \boxed{(1 + \log_2 n)}$

for each value of i,

* Outermost loop -

$n/2, n/2+1, n/2+2,$ --- k term

k^{th} term $= \frac{n}{2} + k$

$n = n/2 + k$

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

total complexity $= n/2 + (1 + \log_2 n) + (1 + \log_2 n)$

$= \boxed{O(n(\log_2 n)^2)}$

Que3 T-C

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

As

i	j
1	1→n
2	1→n
3	1→n
⋮	
n	

n * n times

ntimes   ntimes

for functions (n-3)

$n, n-3, n-6, n-9, ---$ kth term.

$n, n-3, n-2\times3, n-3\times3, ----$ kth term

$$k^{th}\text{ term} = n(-k-1)\times 3 = n-3k-3$$

$$1 = n-3k-3$$

$$n-3k-4=0$$

$$\boxed{k = \frac{n-4}{3}}$$

inner most loop will execute $= n^{*}\ \frac{n^{*}(n-4)}{3}$

$$\therefore \frac{n^3-4n^2}{3}$$

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

Date.

Page No.

Q9 Time C.

```
void function (int)
  {
    for (i=1 to n)
   {
      for(j=1; j<=n; j=j+1)
          print("*");
  }}
```

As

Outer loop will return n times (i)

for i=1, j will return n times

i=2; j will return n/2 times

i=n; j will return n/n times

$$\text{innerloop} = \left(n + \frac{n}{2} + \frac{n}{3} + \cdots \frac{n}{n-1} + \frac{n}{n}\right) \text{times}$$

$$n \cdot \left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \cdots \frac{1}{n}\right)$$

$$\boxed{n \cdot \log n}$$

Complexity O(n.logn).

Que10 For the func n^k and c^k what is asymptotic relationship b/w these func?
Assume that $K>=1$ and $c>1$ are constants
Find value of c and no. of n for which relation holds

As

$n^k = O(c^n)$ as n approaches infinity
n^k is bounded above by c^n.

bubble

selection

insertion

merge

quick

Randomized quick

heap sort

◦ Count sort.

bubble

```
for(i=0;i<n-1;i++)
{ for (j=0; j<n-i-1; i++)
    { if (A[j] > A(j+1))
        swap(A[j], A(j+1));
```