

## Assignment - I

Tut 1 - b

1.

Ans: Asymptotic notations are mathematical tools to represent the time complexity of algorithm for asymptotic analysis.

• O Notation - it bounds a function from above & below, so it defines exact asymptotic behavior.

$$\text{Ex} - 3n^3 + 6n^2 + 6000$$

$$O(n^3)$$

Dropping lower order terms is always fine because there will always be no after which  $O(n^3)$  has higher values than  $O(n^2)$ , irrespective of constants involved.

$O(g(n)) = \{f(n) : \text{there exists some constants } c_1, c_2 \text{ and no such that } O^2 = c_1 * g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\}$

- Big O (Notation): it defines an upper bound of an algo; it bounds a function only from above. for ex - consider the case of insertion sort. It takes linear time in best Case and quadratic time in worst Case.

$O(n^2) \rightarrow$  worst Case

$O(n) \rightarrow$  best case

$O(n^2)$  is overall case.

$\Theta(g(n)) = \{ f(n) : \text{there exists positive constants } c \text{ & } n_0 \text{ such that } 0 \leq c * g(n) \leq f(n) \text{ for all } n \geq n_0 \}$

- $\Omega$  notation:

provides lower bound,

(least used)

$\Omega(g(n)) = \{ f(n) : \text{there exists a constant } c \text{ and } n_0 \text{ such that } 0 \leq g(n) \leq f(n) \text{ for all } n \geq n_0 \}$

ex: for  $\text{fib}(n)$ :

$$\begin{array}{ccc} \text{fib}(n-1) & \text{fib}(n-2) & \Rightarrow O(2^n) \\ \swarrow & \searrow & \\ & & \Rightarrow O(1.6^n) \\ & & \Rightarrow O(1.5^n) \end{array}$$

- $O(n)$ : loose upper bound of  $f(n)$

let  $f(n)$  and  $g(n)$  be functions

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

$$\forall f(n) \leq c(g(n)) \quad \text{ex - Merge Sort}$$

$$\forall n > n_0 \quad \&$$

$$\& c > 0$$

- $\Omega$  notation : Strict lower bound - Small Omega

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

$$f(n) \geq c(g(n))$$

$$\forall n > n_0 \quad \& \quad \forall c > 0$$

ex - Merge Sort

p2.

for ( $i=1$  to  $n$ )

$i$

$i = i + 2$

$2$

$1$

$2^0$

$2^1$

$\vdots$

$n$

$2^0$

$2^1$

$\vdots$

$2^K$

$(K+1)^{\text{th}}$  iteration

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

$$k \log_2 2 = \log n \quad (\log_2 2 = 1)$$

$$k = \log n$$

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

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

Sol.

Using Substitution

$$\begin{aligned} T(n) &= 3T(n-1) \\ &= 3(3T(n-2)) \\ &= 3^2(T(n-2)) \\ &= 3^3(T(n-3)) \\ &\quad \vdots \\ &= 3^n(T(n-n)) \end{aligned}$$

$$\Rightarrow 3^n(T(0))$$

$$= 3^n * 1 = 3^n$$

$$= O(3^n)$$

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

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

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

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

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

$$= 2^n(T(n-n)) + 2^{n-1} + 2^{n-2} + 2^{n-3}$$

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

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

$$= 1$$

$$T(n) = 1 \Rightarrow O(1)$$

⑤ T.C.  $\Rightarrow$  init  $i = 1, s = 1;$

while ( $s < n$ )

{  
     $i++;$

$s = s + i;$

}  $\text{printf}(“%d”);$

Step.

$i$        $s$

1	1
2	3

3	6
4	10

5	15
---	----

$1 + 2 + 3 + \dots + k$  at  $k^{\text{th}}$

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

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

$$k^2 \geq n$$

$$k \geq \sqrt{n}$$

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

(6)

void function (int n)

{ int i, c = 0;

for (i=1; i \* i <= n; i++)

{ c++;

}

1

2

3

.

.

$k^2$

$$k^2 \geq n$$

$$k \geq \sqrt{n}$$

$O(\sqrt{n})$

7) void fun (int n)

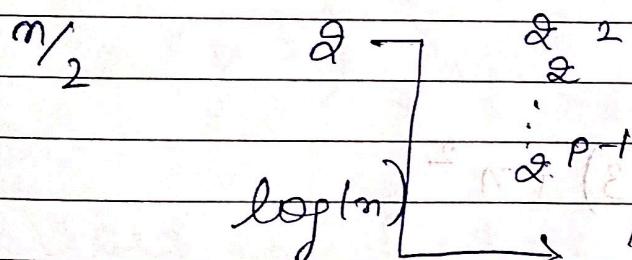
```

    int i, j, k, c = 0;
    for (i = n / 2; i <= n; i++)
        for (j = 1; j <= n; j = j * 2)
            for (k = 1; k <= n; k = k * 2)
                c++;
}

```

i      j      k

$n/2 \rightarrow 1 \rightarrow 1$



$\log(n)$

$O(n * \log(n) * \log(n))$

$= O(n * (\log n)^2)$

$\Rightarrow O(n \log^2 n)$

③  $\text{fun}(\text{int } n)$   
 { if ( $n = 1$ )  
   return;  
   for ( $i=1$  to  $n$ )  
     {  
       for ( $j=1$  to  $n$ )  
         {  
           printf("\*");  
         }  
       }  
       fun( $n-3$ );  
     }  
 }

Sol.  $T(n) = T(n-3) + n^2$

↓

$$T(n) = T(n-3) + n^2$$

$$T(1) = \uparrow$$

$$T(n) = T(n-3) + n^2$$

But  $n = n - 3$

$$T(n-3) = T(n-6) + (n-3)^2$$

$$T(n-6) = T(n-9) + (n-6)^2$$

$$T(n) = T(n-6) + (n-3)^2 + n^2$$

$$= T(n-k) + (n-(k-3))^2 + (n-(k-6))^2 + \dots + n^2$$

but  $n-k = 1$ ,  $k = n-1$   $\Rightarrow$   $n^2 - 1^2$

$$+ (1) + (-2)^2 + (-5)^2 + (-8)^2 + \dots n^2$$

$$= 1 + 4 + 25 \dots n^2$$

$$= 1 + \sum (3n-1)^2$$

$$= \sum (3n-1)^2$$

$$= \sum (9n^2 + 1 - 6n)$$

$$= 9(n)(n+1)(2n+1)$$

$\approx n^3$

$O(n^3)$

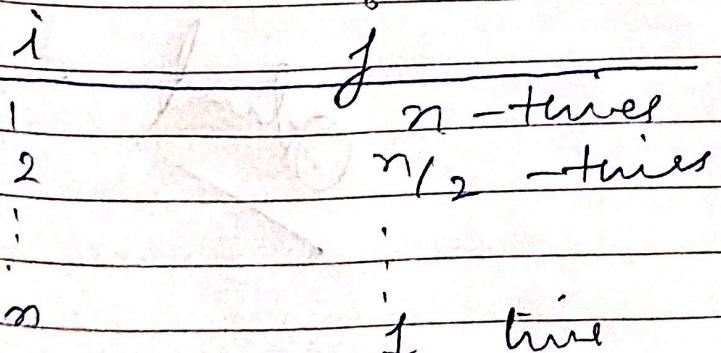
⑦ void fun (int n)

{ for ( $i = 1$  to  $n$ )

{ for ( $j = 1$ ;  $j \leq n$ ;  $j = j + i$ )

{ printf ("\*");

}



$O(n \log n)$

$$\textcircled{6} \quad n^k = O(a^n)$$

$$C * g(n) \leq f(n)$$

$$C * a^n \leq n^k$$

for  $k > 1$

$$a > 1$$

$$\text{let } k = 2$$

$$a = 2$$

$$C * 2^{2n} \leq n^2$$

$$n = 2$$

$$C = 1$$

$$\text{let } k = 1$$

$$a = 2$$

$$C * 2^n \leq n$$

$$n = 0, C = 0$$

$$\text{let } k = 2, a = 2$$

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

$$g(n) = 2^k$$

$$\text{for } k > 2 \text{ &} \\ a > 2$$

$$f(n) \geq C * g(n)$$

$$n^2 \geq C * 2^k$$

log:

$$O(\log n)$$

↓  
↓