Osheen Langer
Sec - A
Roll No - 30

2014425
classmate
Date
Page

Assignment - 1

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

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.

$O(g(n)) = \{f(n)$: there exists +ve constants c_1, c_2 and no such that $0 <= c_1 * g(n) <= f(n) <= c * g(n)$ for all $n >= n_0\}$

- **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.

$O(n^2)$ → worst case
$O(n)$ → best case
$O(n^2)$ is overall case.

$O(g(n)) = \{ f(n)$: there exists positive constants c & n_0 such that $0 <= c * g(n) <= f(n)$ for all $n > n_0 \}$.

- **Ω notation**:
provides lower bound,

(least used)

$\Omega(g(n)) = \{ f(n)$: there exists +ve const c and n_0 such that $0 \leq g(n) \leq f(n)$ for all $n >= n_0 \}$

ex: for fib(n):

fib(n-1) fib(n-2) => $O(2^n)$
=> $O(1.6^n)$
=> $O(1.5^n)$

classmate
Date
Page

- O (notion): loose upper bound of f^n

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

$f(n) = O(g(n))$
$f(n) < c\ (g(n))$
$\forall\ n > n_0$ &
$\forall\ c > 0$

ex - Merge Sort

- Ω notation : Strict lower bound - Small omega

$f(n) = \Omega(g(n))$
$f(n) > c\ (g(n))$
$\forall\ n > n_0$ & $\forall\ c > 0$

ex - Merge Sort

Q.

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

Sol.

i	
1	2^0
2	2^1
:	:
n	2^k $(k+1)^{th}$ iteration

$2^k = n$

$\log 2^k = \log n$

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

$k = \log(n)$

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

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

Sol:

Using Substitution

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

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

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

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

$= \cdots$

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

$= 3^n(T(0))$

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

$= O(3^n)$

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

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

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

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

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

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

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

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

$= 1$

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

(5) T.C ⇒

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

Sol.

i	s
1	1
2	3
3	6
4	10
5	15

$1 + 2 + 3 + \cdots K$ at K^{th}

$$\frac{K(K+1)}{2} >= n$$

$$\frac{K^2 + K}{2} >= n$$

$$K^2 >= n$$

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

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

(6)
```
void function (int n)
{
int i, c=0;
for (i=1; i*i<=n; i++)
  c++;
}
```

i

1
2^2
3^3
.
.
.
K^2

$K^2 >= n$

$K >= \sqrt{n}$

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

(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

$n/2$ → 2 → $\log(n)$

k
2^2
2
$\vdots$
2^{p-1}
1

j
2
2^2
2^{p-1}
$\log(n)$

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

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

$$\Rightarrow \boxed{O(n \log^2 n)}$$

(a)

```
fun ( int n)
{ if (n == 1)
    return;

  for (i = 1 to n)
  {
     for (j = 1 to n)
     { printf("*");
     }
  }
  fun (n-3);
}
```

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

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

$T(1) = 1$

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

put $n = n-3$

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

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

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

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

put $n-k=1$, $k=n-1$

$$T(1) + (-2)^2 + (-5)^2 + (-8)^2 + \cdots n^2$$
$$= 1 + 4 + 25 \cdots n^2$$
$$= 1 + \Sigma (3n-1)^2$$
$$= \Sigma (3n-1)^2$$
$$= \Sigma (9n^2 + 1 - 6n)$$
$$= \frac{9(n)(n+1)(2n+1)}{6}$$
$$\approx n^3$$

$O(n^3)$

(4)
```
void fun (int n)
{
  for (i = 1 to n)
  {
  for (j = 1; j <= n; j = j + i)
      { printf ("*");
  }
```

i	j
1	n - times
2	n/2 - times
:	:
n	1 time

$O(n \log n)$

Date
Page

(1) $n^k = O(a^n)$

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

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

for $k \geq 1$

$a > 1$

let $k = 2$

$a = 2$

$c * 2^n \leq n^2$

$n = 2$

$c = 1$

let $k = 1$

$a = 2$

$c * 2^n \leq n$

$n = 0, c = 0$

let $k = 2, a = 2$

$f(n) = n^2$

$g(n) = 2^k$

for $k >= 2$ & $a >= 2$

$f(n) >= c * g(n)$

$n^2 >= c * 2^k$

log:

$O(\log n)$