Date. 23/02/24
Page No.

Assignment - 1 (DAA)

Q1. Asymptotic notations:

Asymptotic notations is a mathematical notation used to describe the limiting behavior of a function as its input approaches infinity or some other limit. There are 3 commonly used asymptotic notations:

1. Big O notation (O): It represent upper bound of a function's growth rate. It denotes the worst case scenario.
   Eg:- if fnc is $O(n^2)$: it means that the fnc 's growth rate is no worse than quadratic or its growth at most as fast as n^2.
   Eg:- $O(n^2)$ for Bubble sort.

2. Omega notation (Ω) it represents the lower bound of a fnc 's growth rate. It denotes the best case scenario.
   Eg:- if fnc is $\Omega(n)$: its means it's growth rate is atleast linear or as fast as n.
   Eg: $\Omega(n)$: best case time complexity of linear search.

3. Theta notation (Θ) It represent both the upper and lower bound of a fnc's growth rate, providing a tight bound.
   Eg: $\Theta n(\log n)$ for avg and worst case time complexity of Quick Sort.

Date.
Page No.

Q2. Time complexity of

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

Sol=> $i = 1, 2, 4 \ldots$ so on $(i < n)$

Let no. of iteration $= k$

Then $i = 2^k \geq n$

Taking log:

$k = \log_2(n)$

Time complexity is $O(\log n)$.

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

Sol=>

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

$T(0) = 1$

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

~~$= 3$~~

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

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

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

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

Date.
Page No.

general form

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

When $=0$, $T(0) = 1$

So,

$T(n) = 1$

$1 = 3^k (-k)$

$-k^{-1} = 3^k$

$-\log(-k) = k \log 3$

$3^k = n$

$k \log 3 = \log n$

Complexity $T(n)$ is $O(3^n)$

Q4. $T(n) = 2T(n-1) - 1$ if $n > 0$ otherwise 1.

Sol =>

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

when $n > 0$

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

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

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

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

Date.
Page No.

General form:

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

when $n = 0$, $T(0) = 1$; $T(n) = 1$

$$2^n - 1 = 0$$

$$2^n = 1$$

Time complexity = $O(2^n)$

Q5.

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

Sol ⇒ $i = 1, 2, 3$ and so on

let iteration = k

Sum of n ~~natural~~ natural no.

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

$$(k+1)(k+2) \leq 2n$$

$$k^2 + 3k + 2 \leq 2n$$
$$k^2 + 3k - 2n + 2 \leq 0$$

Date.
Page No.

$$k = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$k = \frac{-3 \pm \sqrt{1+8n}}{2}$$

$$k_1 = (-3 + \sqrt{1+8n})/2$$
$$k_2 = (-3 - \sqrt{1+8n})/2$$

Therefore Time complexity = $O\sqrt{n}$

Q6.

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

Sol⇒

iteration $i^2 \leq n$

$i \leq \sqrt{n}$

∴ Time complexity = $O(\sqrt{n})$

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++;
  }
}
}
```

Date.
Page No.

Sol=> iteration of $i = n/2$ to $i = n$

approx. $n/2$ times.

iteration of $j = 1$ to $j = n$
and $j = j * 2$
$\therefore$ $\log_2(n)$ times

Similarly k is $\log_2(n)$ times

Total iteration $\frac{n}{2} \times \log_2(n) \times \log_2(n)$

Time complexity: $O(n \log^2 n)$.

Q6)

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

$TC = O(n^2) + T(n/3)$

Date.
Page No.

9.
```
void func (int n){
  for(i = 1 to n){                    - O(n)
    for(j = 1; j<n; j = j+i)          - O(n)
    {
      printf("GG");
    }
}
```

$i = 1 \quad j = 1, 2, 3, 4 \ldots$
$i = 2 \quad j = 1, 3, 5, \ldots$
$i = 3 \quad j = 1, 4, 7$

$T(n) = n + n/2 + n/3 + n/4 + \ldots$

$T.C = \Theta(n \log n)$

10. Asymptotic relationship b/w n^K and c^n is that c^n grows faster than n^K. In other words c^n is an exponential growth function while n^K is a polynomial growth function.