Anshuman Tomar ML Roll No - 18

Design & Analysis of Algorithms

Assignment - 1

Q.1.

Ans.1. Asymptotic notation is a shorthand used to describe the running time of an algorithm.

- Big O : Used to represent the upper bound of an algorithm's running time. It measures the worst case time complexity of an algorithm.

eg - $O(n^2)$ represents the upper bound of bubble sort algo.

- Omega (Ω) : It represents the lower bound of the running time of an algorithm. It gives the best case time complexity.

eg - $\Omega(n)$ represents lower bound for bubble sort when the array is already sorted.

- Theta (Θ) : It gives both the upper & lower bound of an algorithm. It is used for analyzing the average time - complexity of an algo.

eg - $\Theta(n) \leq f(n) \leq \Theta(n^2)$ for bubble sort.

Ans: 2.

for (i = 1 to n) { i = i * 2 3

⇒ The time complexity for ~~an algorithm~~ a loop is the amount of time the loop works.

i = 1, 2, 4, 8, 16, ...

The seq. is a G.P

common multiplier (r) = 2
first entry (a) = 1
last entry = n

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

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

$\log_2 n = (n-1) \log_2 2$

$\log n = (n-1)$

$n = \log n + 1$

$O(n) = \log(n)$ for the above problem.

Ans-3.

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

$\therefore T(n) = 3T(n-1)$ - (I)

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

$\therefore T(n-1) = 3T(n-2)$ - (II)

put $n = n-2$ in eq - (I),

$\therefore T(n-2) = 3T(n-3)$ - (III)

~~from (I) (II) (III) we can generalise.~~

~~$T(n+1) = 3T(n-n)$ for $n \to 0$.~~

put eq - (II) in (I),

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

$= 9T(n-2)$

put (III) in (IV).

$T(n) = 9\,3T(n-3)$

$= 27T(n-3)$

$= 3+3\times3\,T(n-3)$ - (V)

∴ We can generalize,

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

now,

but $n - k = 1$

$k = n+1$

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

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

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

$$T_n = 3^{n+1}$$

$$T_n = 3^n$$

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

Ans-4.

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

2). $T(n) = 2T(n-1) - 1$ — (I)

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

$T(n-1) = 2T(n-2) - 1$ — (II)

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

$T(n-2) = 2T(n-3) - 1$ — (III)

put (II) in (I).

2). $T(n) = 2 \times 2 T(n-2) - 1 - 1$ — (IV)

put (III) in (IV).

$T(n) = 2 \times 2 \times 2 T(n-3) - 1 - 1 - 1$ — (V)

We can generalise,

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

put $n-k = 1$

$k = n-1$.

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

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

$O(T_n) = (2^n)$

Ans-5.

$i = 1$, $s = 1$

$i = 2$, $s = 1+2$

$i = 3$, $s = 1+2+3$

$i = k-1$, $s = \frac{(k)(k+1)}{2}$

in $(k+1)^{th}$ iteration,

$s = \frac{(k+1)(k+2)}{2}$ → cond. for when the loop terminates.

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

$k^2 + 2k + k + 2 \geq 2n$

$k^2 + 3k + 2 \geq 2n$

$k > \sqrt{2n}$ for the loop to terminate

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

Ans - 6:

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

i = 1 , i*i = 1
i = 2 , i*i = 4
i = 3 , i*i = 9
i = 4 , i*i = 16

$i*i = 1^2 + 2^2 + 3^2 + 4^2 \dots$

$\frac{[n(n+1)(2n+1)]}{6}$ → terminating cond.

$\frac{[(n+1)(n+2)(2n+3)]}{6} > n$

$(n^2 + 3n + 2)(2n+3) > 6n$

$2n^3 + 3n^2 + 6n^2 + 6n + 4n + 6 > 6n$

$\Rightarrow 2n^3 + 9n^2 + 10n + 6 > 6n$

$f(n) = (\sqrt[3]{n})$

Ans 7.

```
void funct. (int n)
{
    int i, j, k, count = 0;
    for (i = n/2 ; i <= n ; i++)          (n/2)
    {
        for (j = 1 ; j <= n ; j = j * 2)   (n/2) * log(n)
            for (k = 1 ; k <= n ; k = k * 2)
            {  count ++ ;
            }
    }
}
```

(n/2) * log(n) * log(n)

=). the loop of i will work n/2 times.

the loop of j will work (n/2) * log (n) times as it is a geo. progression in a nested loop.

Similarly, loop of k will work (n/2) * log(n) * log(n) times.

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

$= \frac{n}{2} \log n \quad = O(n \log n))$

Ans. 8 :

```
{ if (n == 1) return ;
  for (i = 1 to n)        (n)
  {
      for j = 1 to n)     (n^2)
      { printf("*"); }
  }
  funct. (n-3) ;          (n-3)
}
```

=> ~~f(n) = (n-3) (n^2)~~

More time comp. for inner loops will be $O(n^2)$ as they are nested loops.

the funct. call will happen approximately $(n - n/3)$ times i.e. $O(n)$.

So the Time complexity for the overall function would be,

$$O(fn) = O(n * n^2) = O(n^3)$$