DAA-Assignment

Q1- Asymptotic Notations → They are used to analyze an algorithm when input is large. the efficiency of an algorithm depends on the amount of time, storage and other resources required to execute the algorithm. the efficiency is measured with help of asymptotic notations.

Three main asymptotic notations are:-

(i) Big-O Notation (O)

It represents the upper bound of the running time of an algorithm. Thus, it gives the worst case complexity of an algorithm.

$O(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for $n \ge n_0 \}$

$f(n) = O(g(n))$ $f(n) = 2n^2+3n+1 \Rightarrow f(n) = O(n^2)$ (highest order)

(ii) Big Omega (Ω)

It represents lower bound of an algorithm's time or space complexity.

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

$f(n) = 2n^2+3n+1$ $f(n) = \Omega(n^2)$ [lowest]

(iii) Theta Notation (Θ)

It is used to describe both the upper & lower bounds of an algorithm's time & space complexity. Thus, it gives exact estimate of complexity.

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

$f(n) = 2n^2+3n+1$ $f(n) = \Theta(n^2)$ [both highest & lowest]

Q2 - time complexity of
for(i=1 to n) i=i*2

i times

1 1 to n

$K =$ $i = i*2$

$K = 2^0 = 1$

$K = 2^1 = 2$

$K = 2^2 = 4$

$\vdots$

$K = 2^{p-1} = n$

$2^{p-1} = n$

$p-1 = \log_2 n$

$p = \log_2 n + 1$ → constant

$p = \log_2 n$

$\Rightarrow O(\log n)$ Ans

Q3 - $T(n) = \{3T(n-1)$ if $n>0$, otherwise $1\}$

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

$T(n) = 3T(n-1)$ — (i)

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

$T(n) = 3[3T(n-2)]$ — (ii) by subst. in eqn (i)

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

$T(n) = 3^3 T(n-3)$ — (iii)

$\vdots$ for k times

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

Assume $n-k=0$

$n=k$

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

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

$T(n) = 3^n$

$\boxed{O(3^n)}$

Ans

Q4- $T(n) = \{2T(n-1) - 1$ if $n > 0$ otherwise $1\}$

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

subst. $T(n-1)$ in eq (1)

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

$T(n) = 2^2 T(n-2) - 2 - 1$ — (II)

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

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

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

for k times

$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} - - - - 2^2 - 2^1 - 2^0$

Assume $n - k = 0$

$n = k$

$T(n) = 2^n T(0) - 2^{n-1} - 2^{n-2} - 2^{n-3} - - - - 2^2 - 2^1 - 2^0$

$\Rightarrow 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3} - - - 2^2 - 2 - 1$

$\Rightarrow 2^n - (1 + 2 + 2^2 - - - - 2^{n-2} + 2^{n-1})$

$\Rightarrow 2^n - (1(2^n - 1))$

$\Rightarrow \cancel{2^n} - \cancel{2^n} + 1$

$\Rightarrow 1$

$\boxed{O(1)}$

$S = \frac{a(r^n - 1)}{r - 1}$ $r > 1$

Q5-
```
int i=1 ; s=1;
while (s<=n) {
i++;
s = s+i;
printf("*");
}
```

~~while(i<=n)~~

i	
1	1
2	1+2 = 3
3	1+2+3 = 6
4	1+2+3+4 = 10
⋮	
m times	1+2+3 - - - +m

Assume $s > n$

$\therefore s = (1+2+3 - - m)$

$\Rightarrow \frac{m(m+1)}{2}$

$\therefore \frac{m(m+1)}{2} > n$

$m^2 > n$

$m > \sqrt{n}$

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

Q6-
```
void function (int n) {
int i, count=0;
for (i=1; i*i <=n; i++)
    count++;
}
```

i	
1^2	$i*i > n$
2^2	$i*i = K^2$
3^2	$k^2 > n$
⋮	$k > \sqrt{n}$
k^2	$\boxed{O(\sqrt{n})}$

Date
Page

Q7-

```
void function(int n){
    int i, j, K, count =0;  for(i=n/2; i<=n; I++) log n
    for(j=1; j<=n ; j=j*2); — log n x log n
        for(K=1;K<=n; K= K*2) —log n x log n x log n
            count++
}
```

$O(\log^3 n)$

Q8-

```
function (intn)
 if(n==1)        —— 1
   return;
 for (i=1 to n) { —— n+1
   for(j=1 to n){ —— n(n+1)
     printf("*");
   }
 }
function (n-3) ← T(n-3)
}
```

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

$T(n) = T(n-3) + \underbrace{n^2}_{\text{highest deg. term}}$

$$\left[\begin{array}{l} T(n) = aT(n-b) + f(n) \\ a>0 \ \& \ f(n) = O(n^k) \\ \quad \text{where } k \geq 0 \end{array}\right]$$ Master's theorem

$a=1$

$b=3$

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

$k = 2$

case 2: if $a=1$

$O(n^{k+1})$

$O(n^{2+1})$

$\boxed{O(n^3)}$

Teacher's Signature

Q9-

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

↓
$n \times \sqrt{n}$

i	j
1	1
2	$3 = 1+2$
3	$6 = 1+2+3$
4	$10 = 1+2+3+4$
|	
|	
|	
m times	$1+2+ --- m$

Assume $j > n$

$\therefore j = 1+2+3 --- +m$

$$j = \frac{m(m+1)}{2}$$

$$\therefore \frac{m(m+1)}{2} > n$$

$m^2 > n$ $O(n\sqrt{n})$

$m > \sqrt{n}$ $\boxed{O(n^{3/2})}$

Q10- If k is a constant & $c > 1$ then,

c^n grows faster than n^k as $n \to \infty$ because

rate of growth of c^n is exponential while rate of growth of n^k is

polynomial.
Or we can say $c^n = O(n^k)$.