DAA - Tutorial - 1

Q1. Asymptotic notations describes the algorithm efficiency & performance in a meaningfull way. It describes behaviour of time or space complexity for large instance characteristics.

There are mainly 3 asymptotic notation :-

a.] Big-O Notation

It gives the strict upper bound of an algorithm.

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

iff, $f(n) \leq c \cdot g(n) \quad \forall \quad n \geq n_0$, some constant $c > 0$.

b] Omega Notation (Ω)

It gives the strict lower bound of an algorithm.

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

iff, $f(n) > c \cdot g(n) \quad \forall \quad n \geq n_0, c > 0$

c.] Theta Notation (Θ)

It bounds the function from above & below. So it defines exact asymptotic behaviour.

ex: $3n^3 + 6n^2 + 6000 = \Theta(n^3)$.

Q2. $O(\log n)$.

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

$$T(n) = 3T(n-1)$$
$$= 3(3T(n-2))$$
$$= 3^2 T(n-2)$$
$$= 3^2 T(n-3)$$
$$\vdots$$
$$3^n T(n-n) = 3^n.$$

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

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

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

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

$\vdots$

$= 2^n T(n-n) - 2^{n-1} - 2^{n-2} \cdots - 2^0.$

$\because T(0) = 1.$

$\Rightarrow 2^n - 2^{n-1} - 2^{n-2} \cdots 2^0.$

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

T.C $\Rightarrow O(1).$

Q5. T.C = $O(\sqrt{n})$.

Q6. T.C = $O(\sqrt{n})$.

Q7. T.C $\Rightarrow n/2 \times \log n \times \log n$

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

Q8. T.C = $O(n^3)$.

Q9.

i	j
1	n times
2	n/2 times
3	n/3 times
⋮	
n	n/n-times

$\Rightarrow$ T.C = $O(n \log n)$.

Q10. Since polynomials grow slower than exponentials n^k has an asymptotic upper bound of $O(a^n)$.

for $a = 2$, $n_0 = 2$.