Name → Dhruva Bisht , Roll No → 25

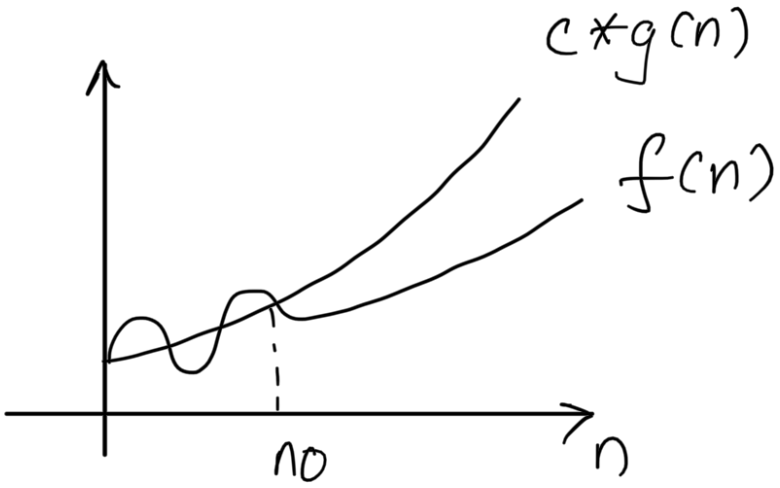
Section → DS

DAA Tutorial - 1

Q1) Asymptotic Notation → These are basically a language to express the required time & space used by an algorithm to solve a given problem.

① Big-O-Notation → It is a notation for the worst case analysis of an algorithm, (upper-bound). According to it $f(n) = O(g(n))$

if and only if n_0 & C are such that $0 \leq f(n) \leq c*g(n)$ for all $n \geq n_0$.



eg → $n+n^2 = O(n^2)$

here $f(n) = n+n^2$, $g(n) = n^2$

$n+n^2 \leq n^2+n^2$ (∵ $n < n^2$, $n^2 = n^2$)

$n+n^2 \leq 2n^2$ (here $c=2$) for $n_0=1$

∴ $f(n) = O(g(n)) \Rightarrow n+n^2 = O(n^2)$

② Big theta (θ) :→ For avg case,

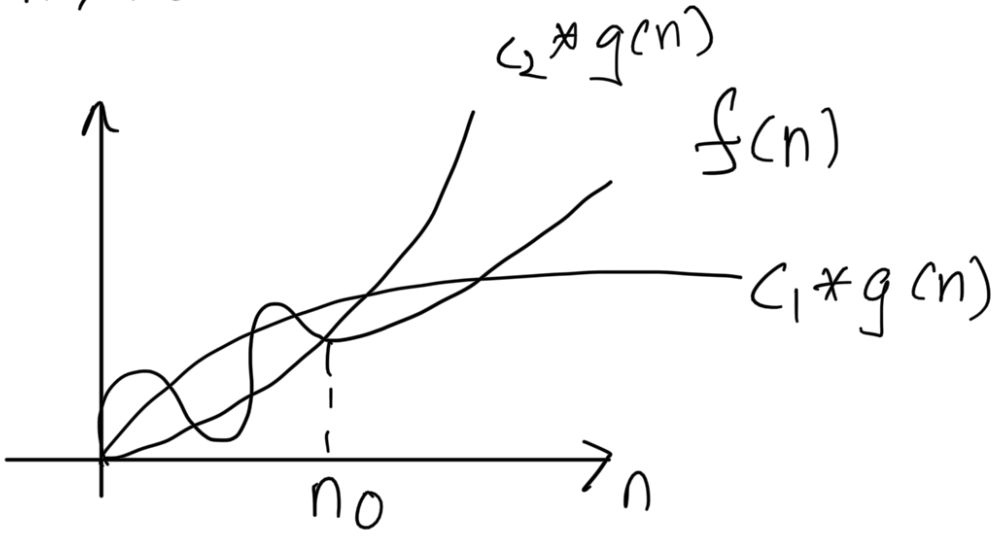
time complexity (tightly bound)
for any two functions f(n) & g(n)

$f(n) = \Theta(g(n))$ if → there exists

n_0, c_1, c_2 as ∃

$$0 \leq c_1 * g(n) \leq f(n) \leq c_2 * g(n)$$

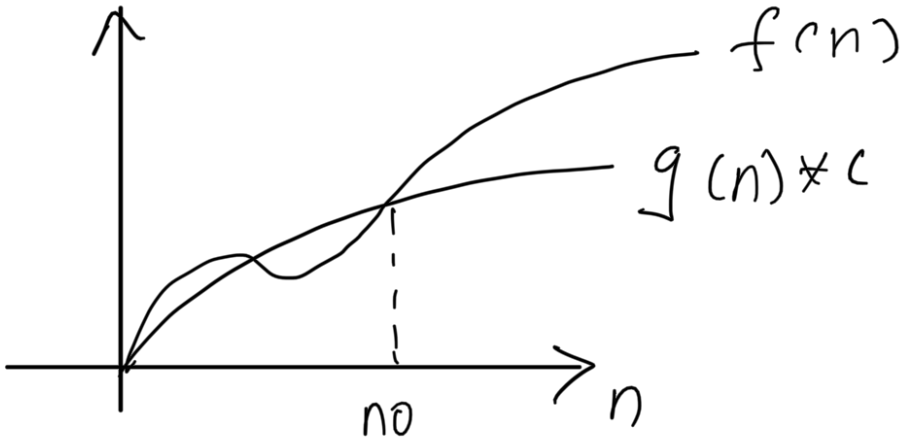
for $n \geq n_0$



③ Big Omega (Ω) → for best case complexity, (lower bound)

$f(n) = \Omega(g(n))$ if $\exists\ n_0, c_1$

$\exists\ 0 \leq c_1 * g(n) \leq f(n)\ \forall\ n \geq n_0$



Q2) TC for (i=1 to n) { i = i*2 }

Series → 1, 2, 4, 8 .... n (G.P)

$a = 1, \; r = 2$

$t_k = a r^{k-1} \Rightarrow n = a * 2^{k-1}$

$n = 2^{k-1}$

$2^k = 2n$

$k = 2 \log_2 n \quad \therefore TC = O(\log_2 n)$

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

$T(n) = 3T(n-1)$ ---①

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

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

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

$T(n) = 3^n T(0) = 3^n \Rightarrow TC = O(3^n)$

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

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

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

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

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

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

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

-2^2-2-1

or) $T(n) = 2^k T(n-k) - 2^{k-k} - 2^{k-2} \cdots - 2^0$

$T(0) = 1$, let $n-k=0$, $k=n$

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

$= 2^n - 2^{n-1} - 2^{n-2} \ldots\ldots 2^0$ {G.P}

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

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

$\therefore O(1)$

Q5) int i=1, s=1;

while (s <= n) {

c++; s = s+i;

printf ("#");

}

Seriese = 1, 3, 6, 10, 15 .... n

1st it => s = s+1

2nd => s = s+1+2 ... till n

$$\therefore \quad \frac{k*(k+1)}{2} <= n \Rightarrow k^2 <= n$$

$$\Rightarrow \quad k = O(\sqrt{n}) = TC$$

Q6)
```
for (i=1 ; i+ i <= n ; i++)
    count ++
```

let loop run till $i = k$

$$k^2 <= n \; ; \quad k <= \sqrt{n}$$

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

Q7)
```
for ( i = n/2 ; i <= n ; i++)
{
    for (j=1 ; j <=n; j=j*2)
        for (k=1 ; k <= n ; k=k*2)
```

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

Q8) Recurrance Relation $\Rightarrow T(n) = T(n-3) + n^2$

$$T(n) = T(n-6) + 2 + n^2$$
$$= T(n-9) + 3n^2$$
$$= T(n-3k) + kn^2$$

$T(1) = 0$; $n - 3K = 1$; $K = \frac{n-1}{3}$

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

$\Rightarrow TC = O(n^3)$

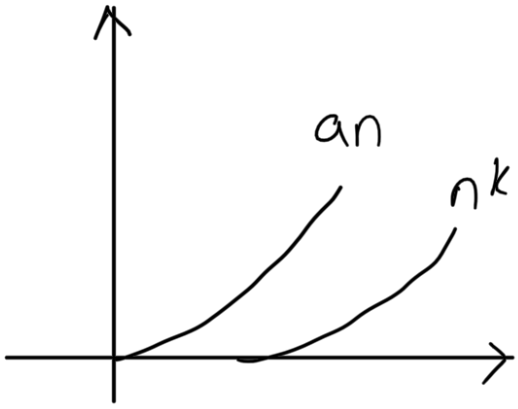
Q9)

i	j	times
1	$1 \to n$	n
2	$1 \to n$	$n/3$
⋮		
n	$1 \to n$	$n \log n$

$\therefore TC = O(n \log n)$

Q10) $n^k = O(a^n)$

$n^k \leq a^n$, $c \forall c > 0$ & $n \geq n_0$



let $n = n_0$

$n_0^k \leq c \cdot 2^{n_0}$

[so let $k = a = 3$; $n_0^3 \leq c \cdot 3^{n_0}$

so $c \geq 1$ & $n_0 \geq 1$]