4/1/24

# * Time Complex & Space Complex ⇒

Q. Obtain Time & Space Complexity for the above -

| | | TC | SC |
|---|---|---|---|
| If (n>5) | — | 1 | 1 |
| { | | | |
| Statement | — | 1 | |
| } | | | |
| Else | — | 1 | |
| Statement | — | 1 | |

$2 \simeq 1 \Leftarrow O(n^0)$   $1 \Rightarrow O(n^0)$

(either if or either else will work)

$O \Rightarrow$ order

TC & SC ⇒ both O are diff

Q. Add main(s)

```
{
  s = 0                      — 1
  for(i=0 ; i<n ; i++)       — n
  {
    s = s+1                  — n
  }
  Return s;                  — 1
}
```

$TC = 2+2n = 2n^0 + 2n^1$

$= O(n)$

$SC = 3 = 3n^0 = O(n^0)$

(s, i, n)

Date
Page

Q. Sum of two matrix

Algo Add(A, B, n) — 1
{
for(i=0 ; i<n; i++) — n
{
for(j=0 ; j<n ; j++) — $n^2$
{
stmt — $n^2$
}

SC = A → n   2n+3
B → n   = O(n)
n → 1
i → 1
j → 1

TC = $n+n^2+n^2$ ⇒ $O(n^2)$

Q. Algo Multiply (A, B, n)
{
for(i=0; i<n; i++) - n
{
for(j=0; j<n ; j++) - $n^2$
{
Stmt; - $n^2$
for(k=0; k<n ; k++) - $n^3$
{
Stmt; $n^3$
}
}

SC =
A - n
B - n
n - 1
i - 1
j - 1
k - 1

SC = 2n+4
= O(n)

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

Q. for (i = 1 ; i < n ; i = i * 2)

{

stmt;

}

TC = $O(\log_2 n)$

SC - i - 1 $\Rightarrow O(n^0)$

n - 1

| i | i*2 | |
|---|---|---|
| 1 | 2 | - $2^1$ |
| 2 | 4 | - $2^2$ |
| 4 | 8 | - $2^3$ |
| 8 | 16 | - $2^4$ (n) |
| ' | ' | |
| ' | ' | |

$\Rightarrow 2^k = n$

$k \log 2 = \log n$

$k = \dfrac{\log n}{\log 2}$

$k = \log_2 n$

$k \Rightarrow O(\log_2 n)$

5/4/24

eg- for (i = n ; i >= 1 , i = i/2)

{

sm ;

}

| i | i/2 | |
|---|---|---|
| n | n/2 | - $n/2$ |
| n/2 | n/4 | - $n/2^2$ |
| n/4 | n/8 | - $n/2^3$ |
| n/8 | n/16 | - $n/2^4$ |

$n/2^k = 1$

TC = $n = 2^k$

$k \log 2 = \log n$

$\boxed{k = \log_2 n}$

SC = ~~i -~~

~~n -~~

~~$\frac{k}{2} = n$ , $k = 2n$~~

| i | i/2 | |
|---|---|---|
| 1 | 1/2 | - |
| 2 | 1 | - |
| 3 | 3/2 | - |

SC = $O(n^0)$

$+ + \frac{n(n+1)}{2} \rightarrow 2^{\log 2} = 2$

Date
Page

eg
```
for(i=0; i<n; i=i++)
    for(j=0; j<i; j++)
    {
        statement
    }
```

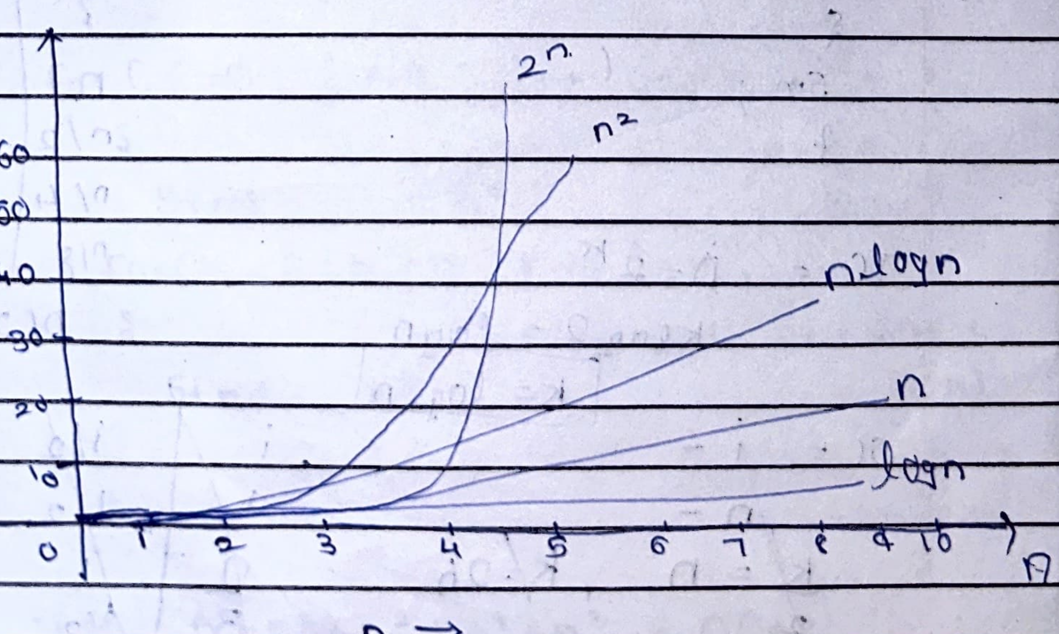
| i | j | outer | inner |
|---|---|---|---|
| 0 | | ✓ | × |
| 1 | 1 | ✓ | ✓ |
| 2 | 1, 1 | ✓ | ✓ |
| 3 | 1, 1, 1 | ✓ | ✓ |

$TC = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{n}{2} = O(n^2)$

$SC = O(n^0)$

* Growth of Funct$^n$ ⇒ (refer the table in ppt)

$1 < \log(n) < \sqrt{n} < n < n\log(n) < n^2 < n^3 - - 2^n < n^n$




Time complexity.

* Asymptotic Analysis ⇒ A method of describing limiting behavior

- Big O
- Big Omega
- Big Theta

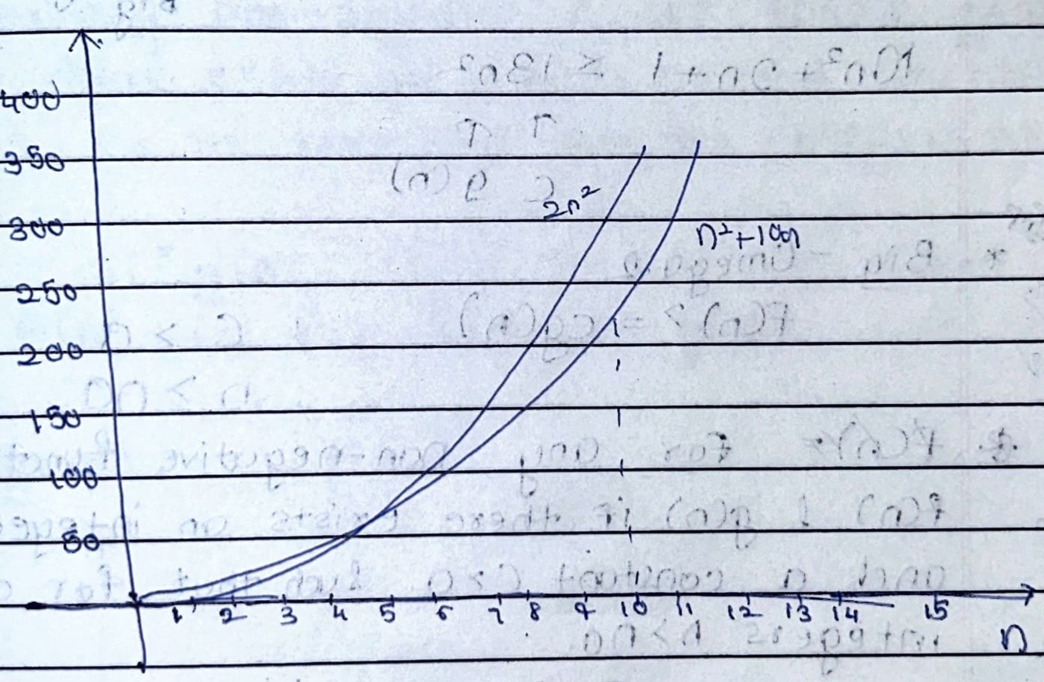
(Algo Book) - Write def of the above

⇒ Big O - for any non-negative funct$^n$, f(n) & g(n), if there exists an integer no and a constant $c > 0$ such that for all integers $n > n_0$,

(gives upper bound)

$$f(n) \le c\,g(n)$$

then f(n) is Big O of g(n).

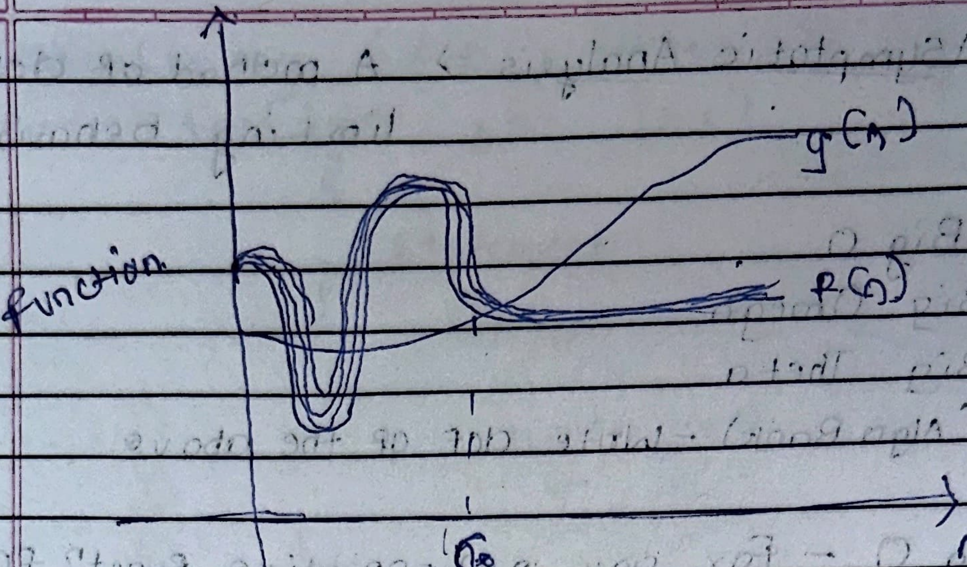



eg $F(n) = 3n+5$ $\quad f(n) \le c\,g(n)$

$(3n+5) \le 8n$ → (we need to chk for each cond$^n$ & get the most appropriate one)

$8(n) = n$

$c = 8$




10/1/24

Q. $f(n) = 10n^2 + 2n + 1$

→ $f(n) \leq cg(n)$

Big 'O'

$10n^2 + 2n + 1 \leq 13n^2$

↑ ↑

c g(n)

From ~~Imp~~

* Big - Omega Ω

$f(n) >= cg(n)$ ; $c > 0$

; $n > n0$

* f(n) >= For any non-negative funct$^n$ f(n) & g(n) if there exists an integer n0 and a constant c > 0 such that for all integers $n > n0$.

Write in algo bank

$f(n) >= cg(n)$

then f(n) is Big Ω of g(n).

Q. $f(n) = 3n + 5$

$f(n) >= cg(n)$

$3n + 5 >= n$

$\Omega(n) < (1)$ → any value below will always satisfy

* ~~Big - Thetha~~

Q: $10n^2 + 2n + 1 \geq c\,g(n)$

$10n^2 + 2n + 1 \geq 1\,n^2$

$\uparrow$ c $\quad$ $\uparrow$ g(n)

$10(1)^2 + 2(1)^2 + 1$

$10 + 2 + 1$

$=$

* Big - Thetha Θ →

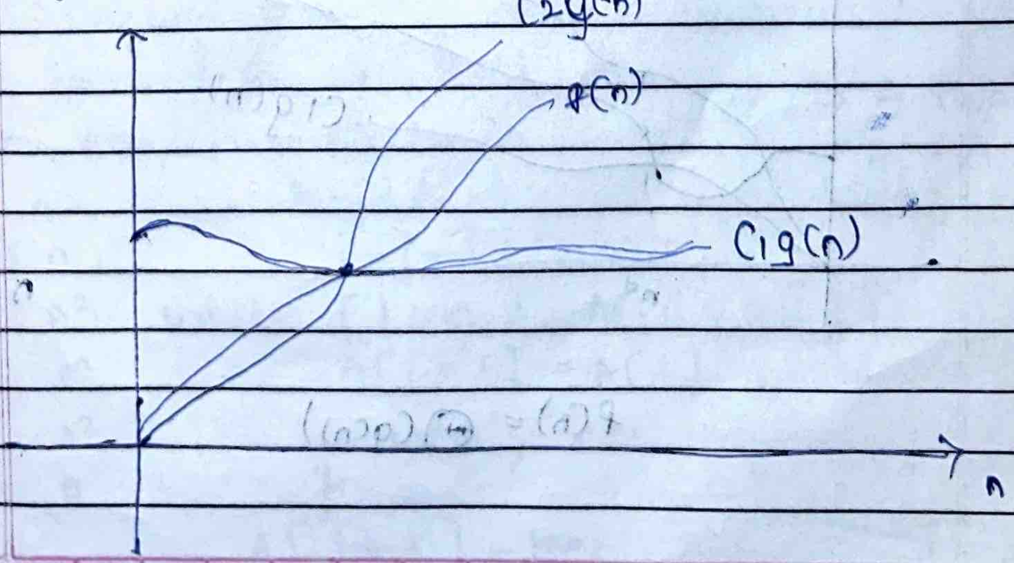
When big-Oh & Ω meet, we indicate this by using Θ notatⁿ.

→ Θ(f(n)), if it is in O(f(n)) and it is in Ω(f(n)).

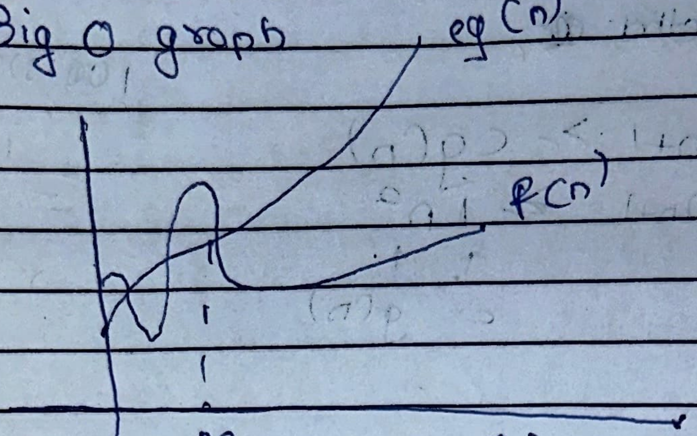
→ For any non-negative functⁿ, f(n) & g(n) if there exists an integer $n_0$ and constant $c_1, c_2 > 0$ such that all the integers $n \geq n_0$

$c_1 * g(n) <= f(n) <= c_2 * g(n)$ then f(n) is Big Thetha of g(n)

→ graph

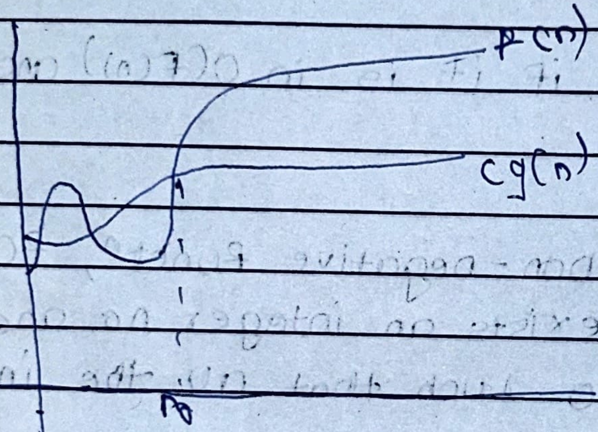


* Big O graph



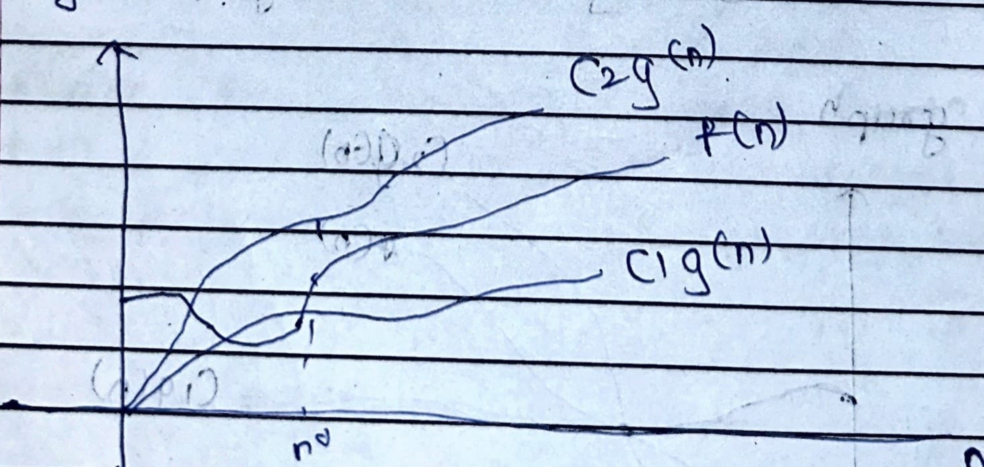

$f(n) = O(cg(n))$

* Big - Ω




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

* Big - Thetha




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

Q. $f(n) = 3n+5$ Big Theta ($\Theta$)

→ $3n+5$

$= 3(1)$

8

$c_1 g(n) \le f(n) \le c_2 g(n)$

$1n \le 3n+5 \le 8n$ — we can get from above two

$c_1 = 1, \quad c_2 = 8, \quad g(n) = n$

Q. $f(n) = 10n^2 + 2n + 1$

$c_1 g(n) \le f(n) \le c_2 g(n)$

$1n^2 \le 10n^2 + 2n + 1 \le 13n^2$

Q. $f(n) = n!$

$c_1 g(n) \le n! \le c_2 g(n)$ — $n^1$

$\le n! \le ?$ — $n^2$

Can't ~~predict~~ (Because ~~that~~ nothing can be less then one so can't be able to find $c_1$ & $c_2$ & $g(n)$. Draw $n!$ graph)

* Insertion Sorting ⇒

eg - 5 2 4 6 1 3

→ B $O(n)$
→ A $O(n)$
→ W $O(n^2)$

```
n    for j = 2 to length
n        key = A[j]
n        i = j - 1;
n^2  while [i > 0 & A[i] > key] {
n^2      A[i+1] = A[i]
n^2      i = i - 1;
n    }
     A[i+1] = key
```

$f(n) = 4n + 3n^2$

$= O(n^2)$

$\Omega$

11/1/24

* ~~Relation~~ ⇒ Recurssion

$Fac(n) = n * Fac(n-1)$

→ Funct$^n$ must call itself & it must terminate properly

* Fibonacci Series ⇒

* 3 methods to identify recurssion ⇒

① Iterative Method
② Binary Tree Method
③ Master Method

```
void Test(n)
{
  for(i=0; i<n; i++)
  {
     statements
  }
  Test(n-1)
}
```

Funct$^n$ ⇒ $f(n) = n + f(n-1)$

$(n>0)$ - ①

delay

$f(n-1) = f(n-2) + (n-1)$

A] Iterative Method

Delay the eq$^n$ ① by 1

$f(n) = f(n-2) + f(n-1)$ - ②

put ② in ①

$f(n) = f(n-2) + 2n - 1n^0$

$f(n) = f(n-2) + 2n$ - ③

delay eq$^n$ ② by ①

$f(n-2) = f(n-3) + (n-2)$ - ④

$f(n) = f(n-1) + n$ $\quad f(0) = 1$

put eqn (4) in (3)

$f(n) = f(n-3) + 3n$

$f(n) = f(n-k) + kn$ $\quad (k = n)$

$= f(0) + n^2$

$f(n) = 1 + n^2 \rightarrow \underline{O(n^2)}$

B] Binary Tree

```
            f(n)
          /      \
         n      f(n-1)
                /     \
             n-1      f(n-2)
                      /     \
                   n-2      f(n-3)
                              |
                           f(n-k)
                           /     \
                          1      f(0)
```

$F(n) = 1 + 2 + 3 + \cdots (n-2) + (n(-1)) + n$

$f(n) = \frac{n(n+1)}{2} = \frac{n^2+n}{2} \rightarrow \underline{O(n^2)}$

n(n

eg- void Test (n)

```
{
  for (i=0; i>n; i=i*2)
  {
     statement
  }
  Test (n-1)
}
```

$f(n) = f(n-1) + f(n) \cdot \log_2(n)$ — ①

delay $eq^n$ ① by 1

$f(n-1) = f(n-2) + \log_2(n-1)$ — ②

put ② in ①

$f(n) = f(n-2) + \log(n-1) + \log(n)$ — ③

~~n-k~~

~~f(n) = f~~

$f(n-2) = f(n-3) + \log(n-2)$

$f(n) = f(n-3) + \log_2(n-2) + \log(n-1) + \log_2(n)$ $\quad n-k$

$f(n) = f(n-k) + \log(n-(k-1)) + \dots + \log(n-2) + \log(n-1) + \log(n)$

$n = k$

$f(n) = f(0) + \log(1) + \dots \log_2(n-2) + \log_2(n-1) + \log_2 n$

$= \log(1 \cdot 2 \cdot 3 \dots (n-2)(n-1)n)$

$= \log_2(n!)$

ok

f(n)

$\log_2(n)$ $\quad f(n-1)$

$\log_2(n-1)$ $\quad f(n-2)$

$\log_2(n-1)$ $\quad f(n-3)$

$\log_2(n!)$

# DAC

* Master Method in Bank Book

ex- $T(n) = 2T\left(\frac{n}{2}\right) + n^2$

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ $\quad a=2$

$\quad b=2$

$= aT\left(\frac{n}{b}\right) + O(n^2)$ $\quad c=2$

~~Case~~ III

$\log_b a = \frac{\log_2 a}{\log_2 b} = \frac{\log_2 2}{\log_2 2} = 1$

III

$c = 2 \qquad c > \log_a b \quad \therefore T(n) = O(f(n))$

$= O(n^2)$

eq- $T(n) = 8T\left(\frac{n}{4}\right) + n^2$

$\frac{\log 2^3}{\log 2^2}$

$a=8 \quad b=4 \quad c=2$

III

$\log_a b = \log_8 4 = \frac{\log_2 8}{\log_2 4} = 1.5$ $= \frac{3 \cancel{\log 2}}{2 \cancel{\log 2}}$

$c = \cancel{1.5}\ 2 \qquad c > \log_a b \qquad O(n^2)$

$\quad 2 \quad\ \ 1.5.$

eq- $T(n) = 8T\left(\frac{n}{2}\right) + 1000n^2$

$a=8,\ b=2,\ c=\cancel{n^2}\ 2$

$\log_a b = \log_8 2 = \frac{\log_2 8}{\log_2 2} = \frac{3 \log 2}{\log 2} = 3$

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

Note:

$c < \log_b a - T(n) = O\left(n^{\log_b a}\right)$