DSA

Date :
Page No. :

⇒ Order of Growth

A function $f(n)$ is said to be growing faster than $g(n)$ if

→ $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$ OR → $\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$ [$n \geq 0$, $f(n), g(n) \geq 0$

→ $f(n) = 2n+5$
$g(n) = n^2+n+6$

$\lim_{n\to\infty} \frac{2n+5}{n^2+n+6}$ → Dividing N^{st} & D^{st} by n^2

$\lim_{n\to\infty} \frac{2/n + 5/n^2}{1 + 1/n + 6/n^2} = 0$ ∴ $f(n)$ is growing faster than $g(n)$

→ Direct way –

① Ignore the lower order terms
② Ignore leading constant

– Constant < log log n < $n^{1/3}$ < $n^{1/2}$ < n < n^2 < n^3 < n^4 < n^4 < 2^n < n^n

e.g. $f(n) = 2n^2 + n + 6$ → Ignore lower order term & leading coeff
$g(n) = 150n + 2$

$f(n)$'s order is n^2
$g(n)$'s order growth is n
∴ $g(n)$ is better.

Date :
Page No. :

Q2) $f(n) = C_1 \log n + C_2 \rightarrow \log n$
$g(n) = C_3 n + C_4 \log\log n + C_5 \rightarrow n$
↳ as n is higher order than $\log\log n$

∴ $f(n)$ is better than $g(n)$

{ Order $\propto \frac{1}{\text{better}}$ }

Q3) $f(n) = C_1 n^2 + C_2 n + C_3 \rightarrow n^2$
$g(n) = C_4 n \log n + C_5 n + C_6 \rightarrow n \log n$

∴ $g(n)$ is better

⇒ Best, Average & Worst Case
&
Asymptotic Notation

→
```
int getSum ( int arr[] , int n)
{
    int sum = 0
    for (int i=0 ; i<n; i++)
        sum = sum + arr[i];
    return sum;
}
```

→ $C_1 n + C_2$ (Linear, Constant) → n is the order

Date :
Page No. :

Q2) int getSum (int arr[], int n)
{
    if (n % 2 != 0)
        return 0;
    for (int i = 0 ; i < n ; i++)
        sum = sum + arr[i];
    return sum;
}

→ For Best Case → Constant
Average Case → Linear, under the assumption that even and odd are equally likely.
Worst Case → Linear, when n is even.

→ We make the decision of worst case, only when we have different cases.

⇒ Notations

→ Big O : Represent exact bound or upper ~~to~~ bound.

→ Theta : Represents exact bound.

→ Omega : Represent exact or lower bound.

⇒ Big O Notation (Upper bound on Order of Growth)

We say $f(n) = O(g(n))$ iff there exists constants c and n_0 such that $f(n) \leq c(g(n))$ for all $n \geq n_0$.

Date :
Page No. :

Ex→ $f(n) = 2n+3$ can be written as $O(n)$

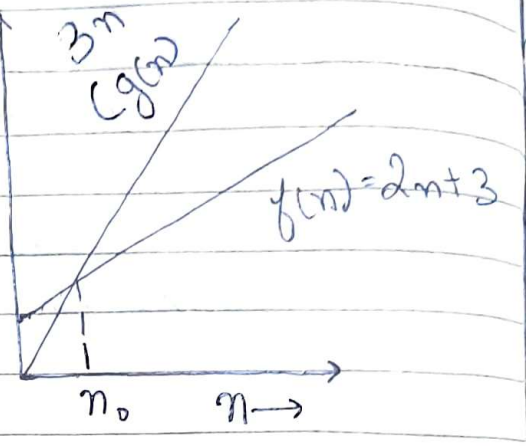
→ $f(n) \leq g(n)$ for all $n > n_0$

$(2n+3) \leq cn$ for all $n > n_0$

(Highest growing term is 2)

$c = 3$

↳ Highest growing term + 1



$(2n+3) \leq 3n$

$3 \leq n$

$n_0 = 3$

→ $\left\{ \frac{n}{4}, 2n+3, \frac{n}{100} + \log n, n+1000, \frac{n}{1000} \ldots \right\} \in [O(n)]$ (Only Highest order)

→ $\{ n^2+n, 2n^2, n^2+1000n, n^2+2\log n \ldots \} \in [O(n^2)]$

→ $\{ 1000, 1, 2, 3, 12 \ldots \} \in [O(1)]$

⟹ **Omega Notation (Lower bound)**

$f(n) = \Omega(g(n))$ iff there exist positive constants c and n_0 such that such that $0 \leq cg(n) \leq f(n)$ for all $n > n_0$.

Ex→ $f(n) = 2n+3 \rightarrow \Omega(n)$

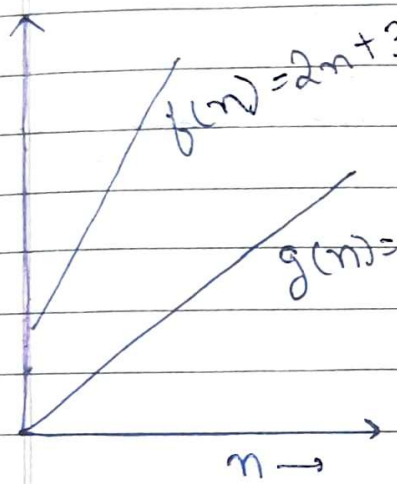
Date :
Page No. :

2) $2n^2 + 3n + 6 \rightarrow \Omega(n^2)$

→ $f(n) = 2n + 3 \rightarrow \Omega(n)$
↳ Highest growing term = 2

$C = 1 \rightsquigarrow$ [Highest growing term -1]

$g(n) = n$
$n \leq 2n + 3$ [$n_0 = 0$] → -ve not allowed



→ $\left\{\frac{n}{4}, \frac{n}{2}, 2n, 3n, 2n+3, n^2, 2n^2 \text{---}\right\} \in [\Omega(n)]$

→ If $f(n) = \Omega(g(n))$
then $g(n) = O(f(n))$

- Omega notation is useful when we have lower bound on time complexity.

Date :
Page No. :

⟹ Theta Notation (Exact order of growth)

$f(n) = \Theta(g(n))$ iff there exist positive constants C_1, C_2 and n_0 such that

$$0 <= C_1 g(n) <= f(n) <= C_2 g(n)$$

for all $n >= n_0$

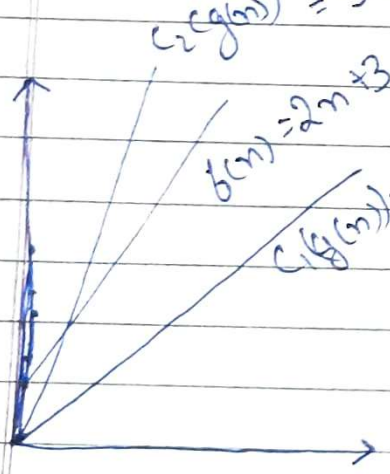
Ex ⇒ $f(n) = 2n+3 \rightarrow \underline{\Theta(n)}$

$$C_1 \times n \leq 2n+3 <= C_2 g(n)$$

$C_1 = 1 \rightarrow$ Any value smaller than Highest growing term
$C_2 = 3 \rightarrow$ " " larger " " " "

$$\underbrace{1 \times n \leq 2n+3}_{n \geq 0} \leq \underbrace{3 \times n}_{n \geq 3}$$

$[n_0 = 3]$ ✓



→ If $f(n) = \Theta(g(n))$
then $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$
and $g(n) = O(f(n))$ and $g(n) = \Omega(f(n))$

→ Theta is useful to represent time complexity

Date :
Page No. :

when we know exact bound. For example, time complexity to find sum, max and min in an array is $\Theta(n)$.

③ $\{ \frac{n^2}{4}, \frac{n^2}{2}, \ldots\ldots 2n^2, 2n^2 + 100n \ldots \} \in [\Theta(n^2)]$ ✓

→ Analysis of some common loop :-

① for (int i=0; i<n; i=i+c)
{
  // Some $\Theta(1)$ work
}

n = 10
c = 2
0 2 4 6 8
5

$\{10/2 = 5\} \Rightarrow n/c$

$\Theta(n/c)$
$= [\Theta(n)]$ ✓

② for (int i=n; i>0; i=i-c)
{
  // Some $\Theta(1)$ work
}

n = 10	n = 11
c = 2	c = 2
i = 10, 8, 6, 4, 2	i = 11, 9, 7, 5, 3, 1

$\lceil n/c \rceil \rightsquigarrow \Theta(n)$

Date :
Page No. :

3) for (int i=1; i<n; i = i*c)
{
// Some O(1) work
}

n=32, c=2	n=33, c=2
1, 2, 4, 8, 16	1, 2, 4, 8, 16, 32

→ $1, c, c^2, c^3 \ldots c^{k-1}$

$[c^{k-1} < n] \approx \Rightarrow k-1 < \log_c n$

$\{k < \log_c n + 1\} \approx$

$\therefore [O(\log n)] \approx$

4) for (int i=n ; i>1; i = i/c)
{
// Some O(1) work
}

n=32, c=2	n=33, c=2
32, 16, 8, 4, 2	33, 16, 8, 4, 2

- Similar to previous question
$[O(\log n)] \approx$

5) for (int i=2; i<n; i = pow(i,c))
{
// Some O(1) work
}

Date :
Page No. :

$c = 2, n = 32$

$2, 2^2, (2^2)^2 = 2, 4, 16$

→ $2, 2^c, (2^c)^c$ - - - -

→ $2, 2^c, 2^{c^2}, \ldots\ldots 2^{c^{k-1}}$

→ $2^{c^{k-1}} < n$

→ $c^{k-1} < \log_2 n$

$k-1 < \log_c \log_2 n$

$[k < \log_c \log_2 n + 1]$ ✓ ⇒ $[\Theta(\log \log n)]$ ✓

⑥
```
void fun (int n)
{
  for (int i=0; i<n; i++)
      // Some O(1) work                    ] O(n)

  for (int i = 1; i<n; i=i*2)
      // Some O(1) work                    ] O(log n)

  for (int i=1; i<100; i++)
      // Some O(1) work                    ] O(1)
}
```

$\therefore O(n) + O(\log n) + O(1) = [O(n)]$ ✓

(log n: ✗, O(1): ✗)

⑦
```
void fun (int n)
{
   for (int i=0; i<n; i++)
       for (int j=1; j<n; j= j*2)
           // Some O(1) work
}
```

Date :
Page No. :

→ In nested loops we simply multiply the no. of times the loops run

→ $O(n) \times O(\log n) \Rightarrow [O(n \log n)]$ ✓

⑧ Void fun (int n)
{
for (int i=0; i<n; i++)
for (int j=1; j<n; j=j*2)
// some O(1) work
] $O(n \log n)$

for (int i=0; i<n; i++)
for (int j=1; j<n; j++)
// some O(1) work
] $O(n^2)$

→ $O(n \log n) + O(n^2) \Rightarrow [\underline{O(n^2)}]$ ✓