Suraj Pokhriyal
D, 44

Date
Page

Assignment

① Asymptotic Notation is used to describe the running time of an algorithm- how much time an algorithm takes with a give input, n.
There are three different notations: big O, big theta (Θ), and big omega (Ω)

① Big O

g(n) is "tight" upper bound.

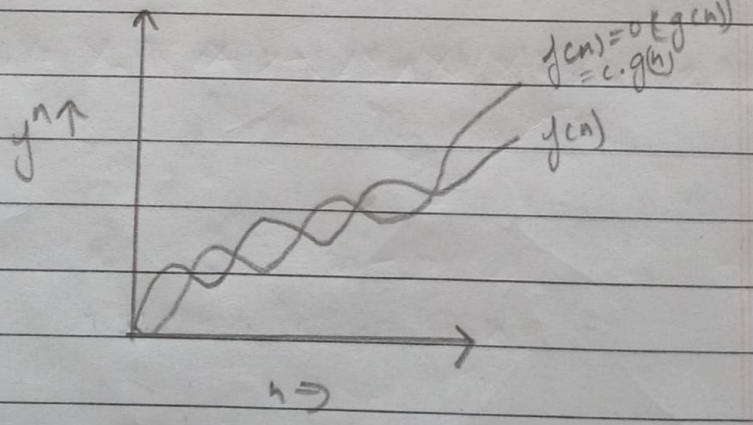
$f(n) = O(g(n))$

iff

$f(n) \leq c \cdot g(n)$

$\forall \quad n \geq n_0$ and

some constant, $c > 0$.



(2) Big Omega

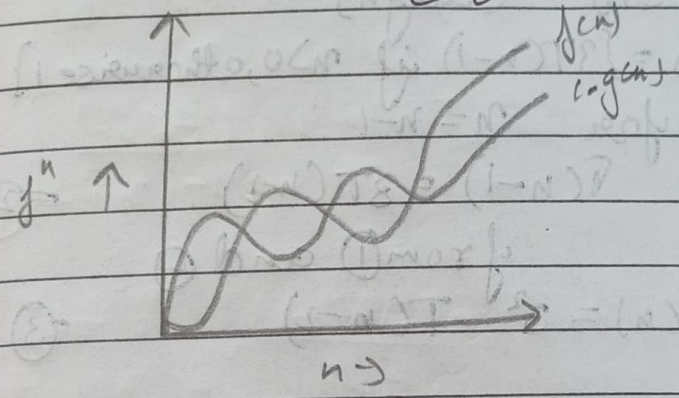
$g(n)$ is "tight" lower bound

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

iff

$f(n) \geq c \cdot g(n)$

$\forall\ n \geq n_0$ and some constant, $c > 0$



(3) Big Theta (Θ)

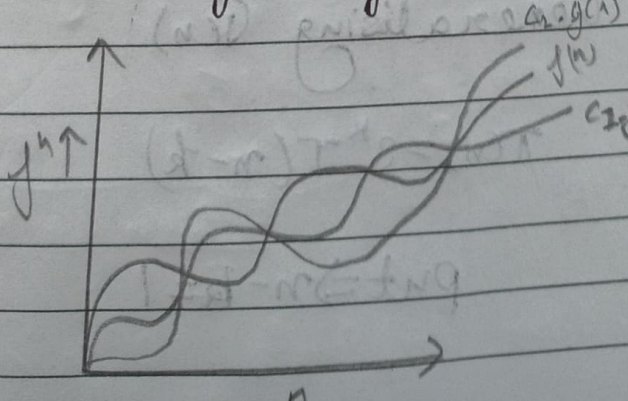
$f(n) = \Theta(g(n))$

$g(n)$ is both "tight" upper and "tight" lower bound of $f(n)$

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

iff

$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$



(2) Time complexity of :-

```
for(i=1 to n)
    { i = i*2; }
```

$2^k = n \Rightarrow$ Taking log

$k \log 2 = \log n$

$k = \log_2 n$

$T(n) = O(\log n)$

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

for $n = n-1$

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

from (1) and (2)

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

from (2) put $n = n-1$

$T(n-2) = 3T(n-3)$ -(4)

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

from (3) & (4)

$T(n) = 3^3 T(n-3)$ -(5)

Generalising $T(n)$

$T(n) = 3^k T(n-k)$ -(6)

put $\Rightarrow n-k=1$

$k = n-1$

put $k = n-1$ in (5)

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

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

from (1)

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

put $n=1$

$T(1) = 3T(0)$

$T(1) = 3.(1)$

$T(1) = 3$

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

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

(4) $T(n) = \{ 2T(n-1)-1 \text{ if } n>0, \text{otherwise} \}$ → (1)

for $n = n-1$

$T(n-1) = 2T(n-2)$ — (2)

from (1) & (2)

$T(n) = 2^2 T(n-2) - 1 - 2$ — (3)

from (2) put $n = n-1$

$T(n-2) = 2T(n-3) - 1$ — (4)

from (3) & (4)

④ $T(n) = 2^3 T(n-3) - 1 - 2 - 4$ — (5)

generalising term of eq (5)

$T(n) = 2^k T(n-k) - 1 - 2 - 4 \ldots 2^k$ — (6)

put

$n-k=1$

$k=n-1$

put $k=n-1$ in eq (6)

$$T(n) = 2^{n-1} T(1) - 1\left[\frac{1-2^{n-1}}{1-(n-1)}\right]$$

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

⑥

$T(n) = O(1)$ //

⑤

i	j
1	1
2	3
3	6
4	10
5	15
!	!
k	$\frac{k(k+1)}{2}$ for k iteration

(5) loop terminates when $\frac{k(k+1)}{2} > n$

∴ Time Complexity $k = O(\sqrt{n})$.

(6)

i	count
1	0
2	1
3	2
!	!
!	!
k	(k-1)

Since $k*(k-1) \leq n$

$\therefore k \leq \sqrt{n}$

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

(7)

i	j	k
1	logn	logn * logn
2		
2	logn.	logn * logn
1		
!		
n	logn	log * logn

$\therefore (n + \log n * \log n)$

$\Rightarrow O(n * (\log n)^2)$

(8) $a = n-3$, $d = -3$

$1 = (n-3) + (k-1)(-3)$ [$\because a_n = a + (n-1)d$]

$1 = n - 3 + (-3k) + 3$

$3k = n - 1$

$k = \frac{n-1}{3}$

$= O(n * n^2)$

Time Complexity : $O(n^3)$.

(9) for $i = 1 \rightarrow j = n$ times.

$i = 2 \rightarrow j = n/2$ times

$i = 3 \rightarrow j = n/3$ times

$i = k \qquad j = n/k$ times

$\therefore i = n$ & $j = \frac{n}{n}$ times.

Time Complexity : $(n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n})$

$n\underbrace{\left(\frac{1}{2} + \frac{1}{2} + \frac{1}{3} + \cdots \frac{1}{n}\right)}_{\log n}$

$\therefore$ Time Complexity : $O(n \log n)$

10) $f(n) = n^k$, $g(n) = c^n$

where $k=1$ & $c=2$

$\Rightarrow f(1) = (1)^1$ $\quad g(1) = 2^1$

$f(1) < g(1)$

$\Rightarrow f(2) = (2)^1$ $\quad g(2) = (2)^2 = 4$

$f(2) < g(2)$

Satisfies O notation $f(n) \leq g(n)$

$f(n_0) = c_0 \cdot g(n_0)$

$n_0^k = c_0 \cdot c^{n_0}$

$k=1$, $c=2$

$n_0^1 = c_0 \cdot 2^{n_0}$

$\left(\frac{n_0}{c_0}\right)^1 = (2)^{n_0}$

Comparing : $\boxed{n_0 = 1}$, $\frac{n_0}{c_0} = 2$

$c_0 = \frac{1}{2}$

$f(n) \leq 0.5\, g(n)$

$f(n) = o\, g(n)$