Design and Analysis of Algorithms

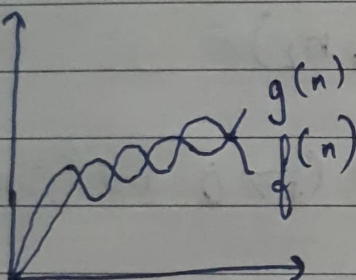
Tutorial - 1

Name: Swetank Singh
CST SPL1
Roll no. 23

Q1 Asymptomatic Notation

→ They help you find the complexity of an algorithm when input is very large.

i) Big O



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

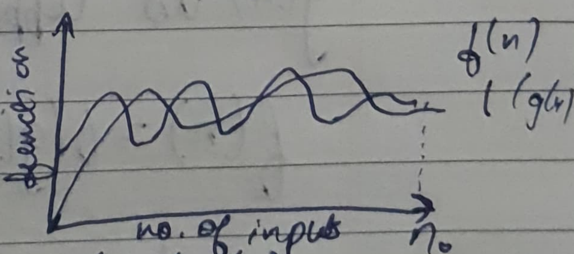
iff $f(n) \leq g(n)$

$\forall\ n \geq n_0$

for some constant $c > 0$

⇒ g(n) is tight upper bound of f(n).

2) Big Omega (Ω)



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

g(n) is tight lower bound of f(n)

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

iff $f(n) \geq (g(n)$

$\forall\ n \geq n_0$ for some constant $c > 0$.

3) Theta (θ)

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

g(n) is both 'tight' upper and lower bound for f(n)

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

iff $C_1 g(n) \le f(n) \le C_2 \cdot g(n)$

$\forall \; n \geq C\max(n_1, n_2)$

for some constant $C_1 > 0$ and $C_2 > 0$

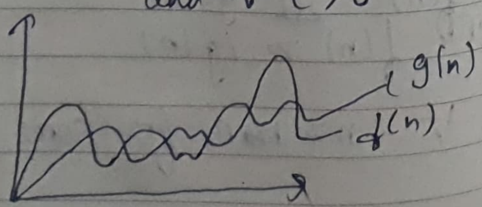
4) Small O(o)

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

g(n) is upperbound of f(n).

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

when $f(n) < C g(n) \; \forall \; n > n_0$ and $\forall \; C > 0$



5) Small omega (ω)

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

g(n) is lower bound of f(n)

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

when $f(n) > C g(n) \; \forall \; n > n_0$ and $\forall \; C > 0$.

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

for (i=1 to n) // i = 1, 2, 4, 8 ---- n.
{ i = i * 2 ; } // O(1)

$\Rightarrow \sum_{i=1}^{n} 1 + 2 + 4 + 8 + \dots n$

GP k^{th} value $\Rightarrow T_k = a r^{k-1}$

$\Rightarrow 1 \times 2^{k-1}$

$\Rightarrow n = 2^{k}$

$\Rightarrow 2n = 2^{k}$

$\Rightarrow \log 2n = k \log 2$

$\Rightarrow \log_2 + \log n = k \log 2$

$\log n + 1 = k$

$O(k) = O(1 + \log n)$

$= O(\log n)$

classmate

Date
Page

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

$T(n) = 3T(n-1)$ — 1

put $n = n-1$

$T(n-1) = 3T(n-2)$ — 2

from 1 and 2,

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

$= 9T(n-2)$ — 3

putting $n = n-2$ in 1,

$T(n) = 3(T(n-3))$ — 4

$\Rightarrow T(n) = 27(T(n-3))$

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

putting $n-k = 0$

$\Rightarrow n = k$

$T(n) = 3^n\ [T(n-n)]$

$T(n) = 3^n\, T(0)$

$[T(0) = 1]$

$T(n) = 3^n \times 1$

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

classmate

Date
Page

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

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

$T(1) = 2T(0)$
$T(0) = 1$

Substituting value of $T(n-1)$ then $T(n-2)$ ---- till $T(1)$ in eqⁿ of $T(n)$.

we get, $T(n) = 2^n \times T(0)$

$T(n) = 2^n \times 1$

$= O(2^n)$

5-
```
int i=1, s=1;
while (s<=n)
{
  i++;
  s= s+i;
  printf(" # ");
}
```

$i = 1\ 2\ 3\ 4\ 5\ 6$ ---

$s = 1+3+6+10+15+$

sum of $s = 1+3+6+10+$ ---- $+h$ — 1

also $s = 1+3+6+10+$ ---- $T_n + h$ — 2

from ① - ②,

$0 = 1+2+3+4+ \dots n - T_n$

$\Rightarrow T_k = 1+2+3+4+ \dots k$

$T_k = \frac{1}{2} k (k+1)$

$\Rightarrow$ for k iterations,

$1+2+3+ \dots + k <= n$

$\Rightarrow \frac{k(k+1)}{2} <= n$

$\Rightarrow \frac{k^2+k}{2} <= n$

$\Rightarrow O(k^2) <= n$

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

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

6. void function (int n) {

```
int i, count = 0;
for (i=1; i*i <= n; i++)
    count++;          // O(1)
}
```

as $i^2 <= n$

$\Rightarrow i <= \sqrt{n}$

$i = 1, 2, 3, 4 \dots, \sqrt{n}$

$\sum_{i=1}^{n} 1+2+3+4+ \dots + \sqrt{n}$

$\Rightarrow T(n) = \frac{\sqrt{n} \times (\sqrt{n}+1)}{2}$

$T(n) = \frac{n \times \sqrt{n}}{2}$

$T(n) = O(n)$

7. void function (int n)

```
{
  int i, j, k, count = 0;
  for (i = n/2; i <= n; i++)
    for (j = 1; j <= n; j = j*2)
      for (k = 1; k <= n; k = k*2)
        count++;
}
```

for $k = k*2$

$k = 1, 2, 4, 8 \dots n$

$\Rightarrow$ GP $\rightarrow a = 1, r = 2$

$$= \frac{a(r^n - 1)}{r - 1} = \frac{1(2^k - 1)}{1}$$

$n = 2^k$

$\Rightarrow \log n = k$

i	j	k
1	log n	log n * log n
2	log n	log n * log n
⋮		
n	log n	log n * log n

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

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

8. function (int n)

```
{ if (n==1) return                          // O(1)
  for (i=1 to n) {     // i=1,2,3,4__n => n
   for (j=1 to n) {    // j=1,2,3,4__n^ =>
     printf ("*");
   }
  }
function (n-3),      T(n/3)
}
```

$\Rightarrow T(n) = T(n/3) + n^2$

$\Rightarrow a = 1, b = 3, f(n) = n^2$

$\Rightarrow a = 1, b = 3, f(n) = n^2$

$c = \log_3 1 = 0$

$\Rightarrow n^0 = 1 > (f(n) = n^2)$

$\Rightarrow T(n) = \Theta(n^2)$

9. void function (int n) {

```
 for (i=1 to n) {                    // O(n)
   for (j=1; j<=n; j=j+1)
     printf ("*");                   // O(1)
 }
}
```

for $i=1 \Rightarrow j = 1, 2, 3, 4 \ldots n = n$

for $i=2 \Rightarrow j = 1, 3, 5, \ldots n = n/2$

for $i=3 \Rightarrow j = 1, 4, 7, \ldots n = n/3$

⋮

for $i=n \Rightarrow j = 1 \ldots$ 1

classmate

Date

Page

$$\Rightarrow \sum_{j=n}^{1} n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \cdots + 1$$

$$\Rightarrow \sum_{j=n}^{1} n\left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots \frac{1}{n}\right]$$

$$\sum_{j=n} n[\log n]$$

$$\Rightarrow T(n) = [n\log n]$$

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

10- Relation b/w n^k and c^n is

$n^k = O(c^n)$

as

$n^k \le a c^n \;\forall\; n \ge n_0$ and some constant $a > 0$

for $n_0 = 1$
$c = 2$

$\Rightarrow 1^k = a_2 1$

$$\boxed{n_0 = 1 \quad \text{and} \quad \cancel{c=2}\;\; c > 1}$$

[signature]