DAA

Tutorial - 1

ANSHIKA KUKRETI

CST

Semester IV

10

2017468

10 - March - 2022

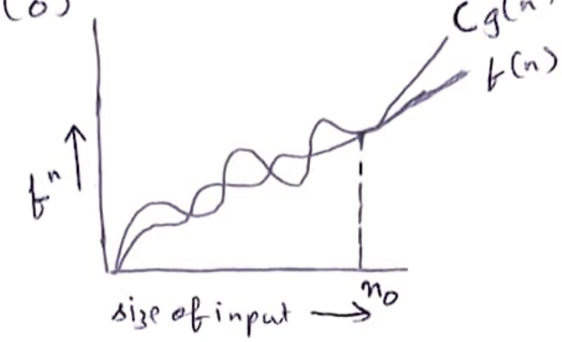
Anshika Kukreti

Q1 Asymptotic Notations -

asymptotic - tending to infinity

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

1) Big O (o)



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

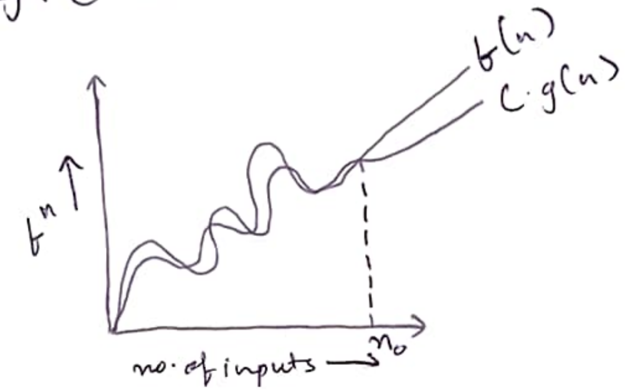
iff $f(n) \leq C\, g(n)$

$\forall\, n \geq n_0$

for constant $c > 0$

$\Rightarrow$ 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)

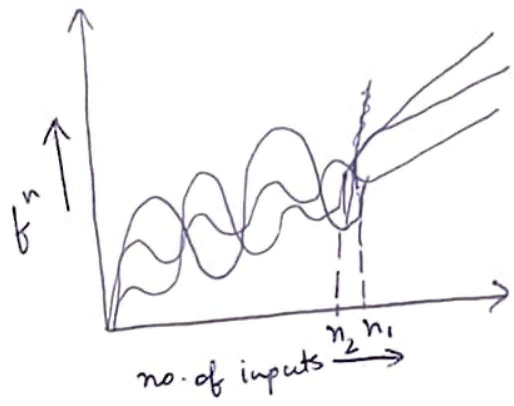
$f(n) = \Omega\, g(n)$

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

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

Avishika Kukreti

3) Theta (θ)



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

g(n) is both 'tight' upper & lower bound of f^n f(n)

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

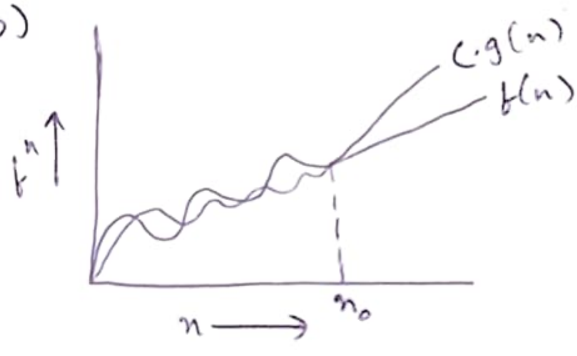
iff

$$C_1(g(n)) \leq f(n) \leq C_2 \cdot g(n)$$

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

for some constant $c_1 > 0$ & $c_2 > 0$

4) small o(o)



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

g(n) is upper bound of f^n f(n)

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

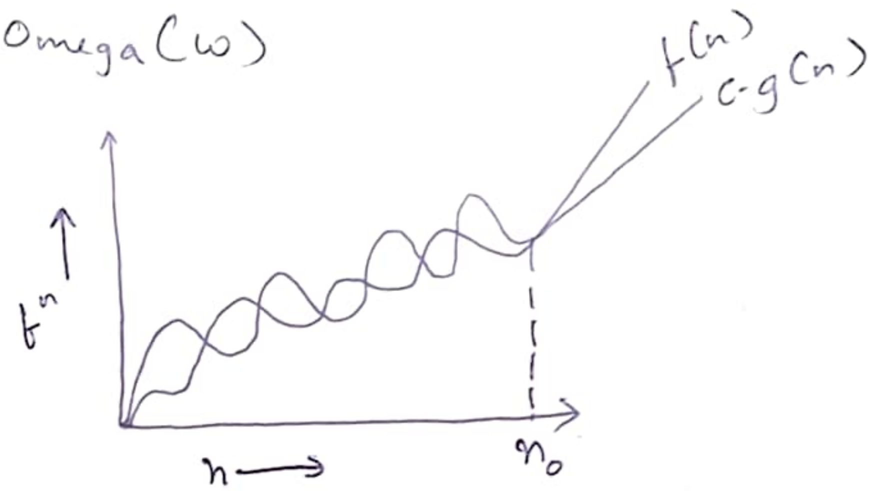
when $f(n) < c \cdot g(n)$

$\forall\ n > n_0$

& $\forall\ c > 0$

Anshika Kulkarni

5) Small Omega (ω)



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

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

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

when $f(n) > c.g(n)$

$\forall\ n > n_0$

& $\forall\ c > 0$

Anshika Kukreti

Q2 what should be time complexity of
for (i = 1 to n) { i = i * 2 }

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

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

kth value $\Rightarrow T_k = ar^{k-1}$

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

$\Rightarrow n = 2^k$

$2n = 2^k$

$\log 2n = k \log 2$

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

$\log n + 1 = k$

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

$= O(\log n)$

Anstika Kukreti

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 & 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

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

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

putting $n-k = 0$
$\Rightarrow n = k$

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

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

$\Rightarrow T(n) = 3^n \times 1$ $[T(0) = 1]$

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

Anshika Mukvati

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

$$T(n) = 2T(n-1) - 1 \quad — ①$$

let $n = n-1$

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

from ① & ②

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

$$T(n) = 4T(n-2) - 2 - 1 \quad — ③$$

let $n = n-2$

$$T(n-2) = 2T(n-3) - 1 \quad — ④$$

from ③ & ④

$$T(n) = 4[2T(n-3) - 1] - 2 - 1$$

$$T(n) = 8T(n-3) - 4 - 2 - 1$$

$$\Rightarrow T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} \dots 1$$

$$GP = 2^{k-1} + 2^{k-2} + 2^{k-3} + \dots 1$$

$$a = 2^{k-1}$$

$$r = 1/2$$

$$\Rightarrow \frac{a(1-r^n)}{1-r}$$

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

Anshika Kukreti

$= 2^k (1 - (1/2)^k)$

$= 2^k - 1$

Let $n - k = 0$

$\Rightarrow n = k$

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

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

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

$T(n) = O(1)$

Krutika Kukreti

Q5 what should be time complexity of

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

$i = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \; \ldots$

Sum of $s = 1 + 3 + 6 + 10 + \ldots + T_n$ — ①

also, $s = 1 + 3 + 6 + 10 + \ldots + T_{n-1} + T_n$ — ②

from ① - ②

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

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

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

$\Rightarrow$ for k iterations

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

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

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

$O(k^2) <= n$

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

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

Arshika Kukreti

Q6 Time complexity of -

```
void fn(int n)
{ int i, count=0;
  for (i=1; i*i<=n; ++i)
    count++;
}
```

as $i^2 <= n$

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

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

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

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

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

$T(n) = \underline{\underline{O(n)}}$

Arshika Kukreti

Q7 Time complexity of -

```
void fn (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 $k = k * 2$

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

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

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

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

$n \Rightarrow 2^k$

$\log n = k$

$\Rightarrow$

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

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

$O(n \log^2 n)$

Krishika Vhakreti

Q8 Time complexity of

```
function (int n)
{ int (n==1)                        // O(1)
  return;                           // i=1,2,3, ... n => O(n)
  for(i=1 to n)                     // j=1,2,3, ... n*n => O(n^2)
  { for (j=1 to n)
    { print ("*");
    }
  }
  function (n-3);
}
```

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

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

$c = \log_3 1 = 0$

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

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

Anshika Kukreti

Q9 Time complexity of -

```
void function (int n)
{ for (i = 1 to n)
    { for (j = 1 ; j <= n; j = j+1)
        print ("*");
    }
}
```

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

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

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

⋮

for $i = n \Rightarrow j = 1$

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

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

$$\Rightarrow \sum_{j=n}^{1} n[\log n]$$

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

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

Arshika Thukrati

Q10 For the functions, n^k and c^n, what is the asymptotic relation between these functions?
assume that $k >= 1$ & $c > 1$ are constant
Find out the value of c and n_0 for which relation holds.

as given, n^k and c^n

relation b/w n^k & c^n is

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

as $n^k \leq a c^n$

$\forall n \geq n_0$ & some constant $a > 0$

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq a 2^1$

$\Rightarrow n_0 = 1$ & $c = 2$

Avshika Uniyal