TCS-409 DESIGN & ANALYSIS OF ALGORITHMS

TUTORIAL - 1

D. Uma Vyshnavi
2017492, CST- SPL-2
Rollno - 37

① What do you understand by Asymptotic Notations. Define different Asymptotic notations with examples.

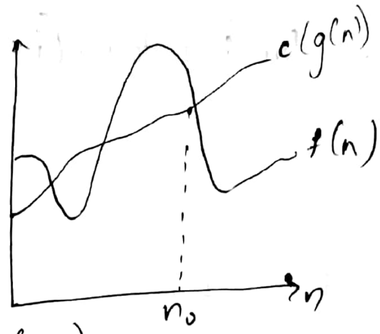
Sol- Asymptotic Notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or limiting value.

Three Asymptotic Notations

big O, big Theta $\Theta$, big Omega ($\Omega$).

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

$f(n) \le C * g(n) \; \forall n,$




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

Big Theta ($\Theta$)

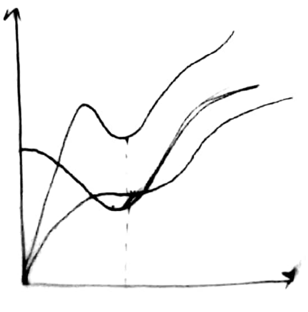
$f(n) = \Theta(g(n))$

g(n) is both upper bound of function f($\theta$)

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

$n > \max(n_1, n_2)$.

for some const $c_1 > 0$ & $c_2 > 0$.



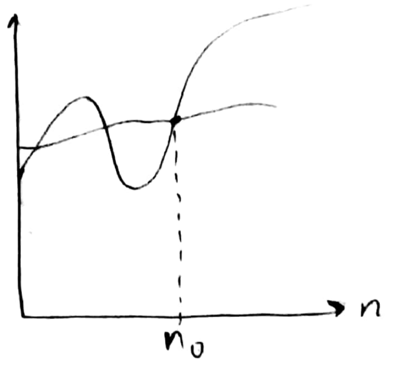
$f(n) = \Theta(g(n))$

Big Omega ($\Omega$)

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

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

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




Omega gives the lower bound of a function.

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

(2) What should be the time complexity of

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

sol:-

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

| | |
|---|---|
| i | 1 |
| 1 | 2 |
| 2 | 4 |
| 4 | 8 |
| 8 | ⋮ |
| ⋮ | ⋮ |
| n = | $2^k$ |

$\log n = \log 2^k$

$\boxed{k = \log_2 n}$

Scanned with CamScanner

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

$T(1) = 1$ using subtract & conquer

$T(n) = 3T(n-1)$ — 1

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

$T(n-1) = 3T(n-2)$ — 2

$T(n-2) = 3T(n-3)$ — 4

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

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

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

⋮

$= 3^* n\ T(n-n)$

$= 3n$

$O(3n)$

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

$T(n) = \{2T(n-1) - 1 \rightarrow 1$

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

$T(n-1) = 2T(n-2) - 1 \rightarrow 2$

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

$T(n-3) = 2T(n-4) - 1 \rightarrow 4$

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

$T(n) = 4(2T(n-3) - 1) - 3 = 8T(n-3) - 4 - 3 = 8T(n-3) - 7$

$T(n) = 8[2T(n-4) - 1] - 7 = 16T(n-4) - 8 - 7 = 16T(n-4) - 15$

$T(n) = 16T(n-4) - 15 \longrightarrow 5$

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

since $T(n-1)$ is reducing by 1

let $n-k = 1 \Rightarrow n-k$

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

$= O(2^n - 2^n + 1) = O(1)$

(5) What should be the complexity of

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

Sol. Let loop executes k times

loop will execute till $s \le n$

After 1st

$s = S + 1$

After 2nd

$s = S + 1 + 2$

After 3rd

$s = S + 1 + 2 + 3$

⋮

$s = S + 1 + 2 + \dots k$

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

$O(k^2) <= n$

$k = O(\text{root}(n))$

Scanned with CamScanner

5) Time complexity of

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

Sol: for $i = n/2$; $i <= n$; $i++$ // $\therefore O(n/2) = O(n)$

for $j = 1$; $j <= n$; $j = j*2$ // $k = \log_2 n$ $\therefore O(\log_2 n)$

for $k = 1$; $k <= n$; $k = k*2$ // $x = \log_2 n$ $\therefore O(\log_2 n)$

$T(n) = O(n) \times O(\log_2 n) \times O(\log n)$

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

$= O(n \cdot \log^2 n) = O(n \log^2 n)$

$\boxed{O(n \log^2 n)}$

(6) Time complexity of :-

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

sol as $i^2 <= n$

$i <= \sqrt{n}$

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

$\sum_{i=1}^{n} 1+2+3+4+ ... +\sqrt{n} \Rightarrow T(n) = \frac{\sqrt{n} \times (\sqrt{n}+1)}{2}$

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

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

(8) Time complexity of :-

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

Sol $T(n) = T(n/3) + n^2$

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

$c = \log_3 1 = 0$

$n^0 = 1 > (T f(n) = n^2)$

$\boxed{T(n) = O(n^2)}$

(9) Time Complexity of:-

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

Sol: 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 $i = n \Rightarrow j = 1 \ldots$

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

$\sum_{j=n}^{1} n(1 + 1/2 + 1/3 + 1/4 + \ldots + 1/n] \Rightarrow \sum_{j=n}^{1} n(\log n)$

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

$\boxed{T(n) = O(n\log n)}$

Scanned with CamScanner

(10) For the functions, n^k & c^n, what is the asymptotic relation between these functions?

Assume that $k >= 1$, & $c > 1$ are constant.

Find out the value of c & no for which relation holds.

Sol As given, $n^k$ & $c^n$

relation between $n^k$ & $c^n$ is →

$n^k = O(c^n)$ as, $n^k \leq a c^n$

$\forall\ n \geq n_0$ and some constant areas

$a > 0$.

for, $n_0 = 1$

$c = 2$

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

$\boxed{n_0 = 1\ \&\ c = 2}$