Name - Tisha Aggarwal
Sec - CST SPL2
Roll No - 31

Design and Analysis of Algorithms

Tutorial - 1

Q1) Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

Different Asymptotic notations are :-

1) Big - O Notation - It defines an upper bound of an algorithm. The function $f(n) = O(g(n))$ if and only if $fn <= c.g(n)$ for all $n >= n_0$ where c and n_0 are constants. Here, $g(n)$ is known as upper bound on values of $f(n)$.

Eg:- $f(n) = 3n + 3$, $g(n) = 4n$.

2) Omega Notation - Ω notation provides an asymptotic lower bound. The function $f(n) = \Omega g(n)$ if $f(n) >= c.g(n)$ for all $n >= n_0$ where c and n_0 are constants.

Here, $g(n)$ is known as lower bound on values of $f(n)$.

Eg:- $f(n) = 3n + 2$ and $g(n) = 3n$.

3) Theta Notation - It bounds a function from above and below so it defines exact asymptotic behaviour. Hence, it is also known as tightly bound.

The function $f(n) = \Theta(g(n))$ if $c1.g(n) <= f(n) <= c2.g(n)$ for all $n >= n_0$ where $C1$, $c2$ and n_0 are constants.

Eg:- $f(n) = 3n+2$, $g(n) = n$, $C1 = 3$ and $c2 = 4$.

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

Time complexity for a loop means no. of times the loop runs.

Loop will run for following values of i:-

$i = 1, 2, 4, 8, 16, 32, ... 2^k$ this means k times

The loop will run till $2^k = n$ which gives $k = \log n$

tisha

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

$T(n) = 3T(n-1)$
$= 3(3T(n-2))$
$= 3^2 T(n-2)$
$= 3^3 T(n-3)$
$\vdots$
$= 3^n T(n-n)$
$= 3^n T(0)$
$= 3^n$

Complexity $= O(3^n)$

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

$= 2(2T(n-2)-1) - 1$
$= 2^2(T(n-2)) - 2 - 1$
$= 2^2(2T(n-3)-1) - 2 - 1$
$= 2^3 T(n-3) - 2^2 - 2^1 - 2^0$
$\vdots$
$= 2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3} \ldots 2^2 - 2^1 - 2^0$
$= 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3} \ldots 2^2 - 2^1 - 2^0$
$= 2^n - (2^n - 1)$

{As, $2^{n-1} + 2^{n-2} + \ldots + 2^0 = 2^n - 1$}

$T(n) = 1$

Complexity $= O(1)$

Q5) After 1st iteration: S = S+1
After 2nd iteration: S = S+1+2

1+2+...+x <= n
(x * (x+1)) / 2 <= n
O(x^2) <= n
x = O (root (n))

s(k) = 1+ 2+ 3 + ... k = (K+1) * K / 2
n = (K+1) * K / 2
=> K = -1/2 +- sqrt(1 + 4 * n) / 2

$= O(-1/2 + sqrt(1+4n)/2) = O(sqrt(n))$

Q7) For $k = k*2$

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

G.P $\Rightarrow a = 1, r = 2$

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

$$n = 2^k$$

$$\log n = k$$

$i = 1, 2, \ldots n$

$j = \log n, \log n, \ldots \log n$

$k = \log n * \log n, \log n * \log n \ldots \log n * \log n$

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

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

Q6) $i^2 <= n$

$i <= \sqrt{n}$

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

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

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

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

$$T(n) = O(n)$$

Q8) $\underbrace{(n-3), (n-6), (n-9), \ldots (1)}_{K}$

$a = n - 3, \quad d = \cancel{n} - 6 - \cancel{n} + 3 = -3$

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

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

$3K = n - 1$

$K = \frac{n-1}{3} = O(n * n^2)$

$= O(n^3)$

Q9) For $i=1$, $j = n$ times
$i=2$, $j = n/2$ times
$i=3$, $j = n/3$ times
⋮
$i=K$, $j = n/K$ times
$i=n$, $j = n/n$ times

Total time complexity $= n + n/2 + n/3 + \dots + n/n$

$$\underbrace{n * (1 + 1/2 + 1/3 + \dots + 1/n)}_{\log(n)}$$

$$= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots \frac{1}{n}$$

$$= \sum_{K=1}^{n} \frac{1}{K}$$

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

Q10) $f(n) = n^K$ $\quad g(n) = c^n$
where $K >= 1$ & $c > 1$
Let $K = 1$ & $c = 2$

$f(1) = (1)^1$
$g(1) = (2)^1$
$f(1) < g(1)$

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

Satisfies O notation,
$f(n) \leq c\, g(n)$
$f(n_0) = C_0 \cdot g(n_0)$
$n_0^K = C_0 \cdot c^{n_0}$

$K = 1$, $c = 2$
$n_0^1 = C_0 \cdot 2^{n_0}$
$\left(\frac{n_0}{C_0}\right)^1 = (2)^{n_0}$

Comparing,

$$\boxed{n_0 = 1}$$

$$\frac{n_0}{c_0} = 2$$

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

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

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

tsh