ASSIGNMENT-2

NAME: ESHWANTH.C

REG NO: 192521002

SUB: DESIGN AND ANALYSIS OF ALGORITH

SUB CODE: CSA 0688

1. Explain Asymptotic Notation with details.

→ The goal of analysis of an algorithm is to compare algorithm in running time & also memory managment.

→ Expressing the compleaity in terms of its relationship to know function, This type analysis called asymptoic analysis.

→ Asymptotic Notation the mathematical way of representing the time complexity.

→ The notation we use to describe the asymptotic running time of an algorithm are defined in terms of function whose domain we set of natural numbers.

There are three types of Notations:

→ Big oh (O) Notation.

→ Big omega (Ω) Notation.

→ Theta (Θ) Notation.

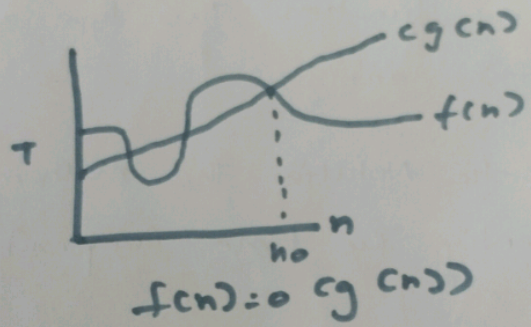
Big oh (O) notation:

Asymptotic less then. This notation mainly represent upper bound of algorithm run time. Big oh notation is useful to calculate the maximum amount of time execution.

Formula:

$f(n) <= (g(n)$

$O(g(n)) = \{ f(n)$: there exist positive constants $0 \le f(n) \le g(n)$ for all $n \ge no \}$



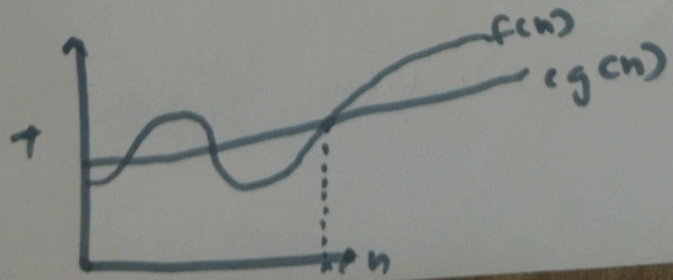
$f(n) = O(g(n))$

omega (Ω) notation:

Asymptatic "greater then" it represent lower bound of algorithm run time by using omega notation we can calculate min amount of time

formula: $f(n) >= c\, g(n)$

$\Omega\, g(n)) = \{ f(n)$: exist positive constant $0 \le cg(n) \le f \; n \ge no$. $\}$



Theta (Θ) notation:

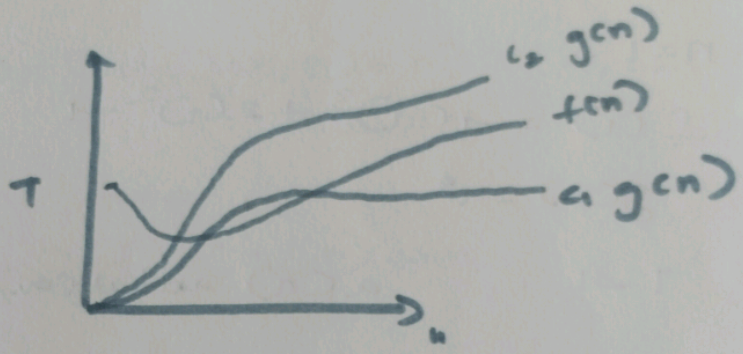
Asymptotic "Equality" at represent average band of algorithm running time. By using Theta notation we can calculate average amount of time, so, it is called time compolexity.

Formula:

$$c_1 g(n) <= f(n) <= c_2 g(n)$$

$\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2 }

$0 < c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$



Master theorem problem:

$$T(n) = 3T(n/4) + n$$

$T(n) = 3T(n/4) + n$ using Master theorm.

$a = 3, \quad b = 4$

$$4(n) = n$$

$$\Rightarrow n \cdot \log_4 3$$

$$f(n) < n \log_4 3$$

$$= O(n)$$

Hence proved.

3. $f(n) = 4n^3 + 2n^2 + n$

$g(n) = n^3$

P.T $f(n)$ is $\Theta(g(n))$

$n = 0$:

$f(0) = 4(0)^3 + 2(0)^2 + 0$

$g(n) = 0^3$

$0 = 0$.

$\Theta(n)$ notation //

$n = 1$:

$f(1) = 4(1)^3 + 2(1)^2 + 1$

$g(n) = 1^3$

$7 < 1$ $\quad$ $a(n)$ notation //

$n = 2$

$f(2) = 4(2)^3 + 2(2)^2 + 1$

$g(2) = 2^3$

$33 < 8$

$a(n)$ notation //

$n = 3$

$f(3) = 4(3)^3 + 2(3)^2 + 1$

$g(3) = 3^3$

$163 < 27$ $\quad$ $\Omega(n)$ notation.

4. Apply the substitution method to solve the recurrence

$T(n) = 2T(n/2) + n$. iteration method

$T(n) = 2T(n/2) + n$

$T(n) = n/2$

$T(n) = 2(2T(n/4) + n/2) + n$ → (1) | $n = 4T(n/4 + 2n$

$T(n) = 2(2T(n/8) + n/4) + n$ → (2) | $n = 8T(n/8 + 3n)$

$T(n) = 2(2T(n/16) + n/8) + n)$ → (3) | $n = 2(n/12 + 4n)$

⋮

times.

$T(n) = 2^k T(n/2^k) + kn$

$\frac{n}{2^k} = 1$

$n = 2^k$

$k = \log_2 n$

$T(n) = 2^{\log_2 n} T(n \log_2 n) + \log_2 n$

$n(1)$

$= nT(1) + (1)\log_2$

$= on(\log n)$

linear

Proved //

5) $h(n) = 5n^4 + 3n^3 + n$ P.T. $h(n)$ is $\Omega(n^4)$

$n=0$

$h(0) = 5(0)^4 + 3(0)^3 + 0$

$= 0$

$n(0)\ 0^4$

$= 0$

$n=1$

$n(1) = 5(1)^4 + 3(1)^3 + 1$

$= 9$

$n=1$

$n(1) = n^4$

$= 1^4$

$= 4$

$n(-2) = 9 > 4$

Condition satisfy

$n=2$

$n(2) = 5(2)^4 + 3(2)^3 + 2$

$= 105$

$n(-2) = 105 > 16$

$n=2$

$n(2) = 2^4$

$= 16$

condity satisfy

$n=3$

$n=3$

$n(3) = 5(3)^4 + 3(3)^3 + 3$

$= 459$

$\therefore n(-2) = 459 > 81$

$n(3) = 3^4$

$= 81$

Condition satisfy

6. **Explain Master Theorem with details :**

The Master Theorem is a tool used to solve recurrence relation that arise in the analysis of divide and conquer algorithms.

$T(n) = aT(n/b) + f(n)$

$f(n) = \Theta(n^k \log_n^3)$

case 1:

If $\log_b^a > k$ Then $\Theta(n \log^a b)$

$\therefore T(n) = \Theta(n \log_b^a)$

case 2:

If $\log_b^a = k$ then $T(n) = \Theta(n \log_b^a, \log n)$

case 3:

$f(n) > n \log_b^a$ then $T(n) = \Theta(f(n))$

where,

n = size of input

a = no. of subproblems in the recursion

n/b = size of each problem

$f(n)$ = cost of work done outside the runtime call.