Growth of function

- The order of growth of running time of an algorithm characterize the algorithm efficiency and allow to compare relative performance of algorithms.

- The study of asymptotic efficiency of algorithms look at large input size to make only the order of growth of running time relevant.

Asymptotic Analysis

- Evaluate the performance of an algorithm in terms of input size (not actual running time).

- Evaluate how does the time (or space) taken by an algo. increase with input size.

Example - Let us consider a search problem in sorted array.
- One way is linear search $\Theta(n)$
- Other is binary search $(\Theta(\log n)$
- Now run linear search in fast computer, binary search in slow computer.
- for small value of n fast computer take less time.
- After certain value of n binary search will definitely start taking less time compared to linear search even on fast machine

- Reason is order of growth of L.S is linear as compared to binary search logarithmic with respect to input size.

Asymptotic notation

- It describe the running time of algo and are defined in term of function whose domains are the set of natural no. N.

(AJAY RAWAT)

- mathematical tools to represent time complexity of algorithms for asymptotic analysis.

classmate

Date ______

Page ______

- It applies to functions like Insertion Sort whose worst case running is $an^2 + bn + c$ for some constant a, b, c and we write $\Theta(n^2)$ (abstracted away some details of the func)

- As Asymptotic notation applies to function, we write as $\Theta(n^2)$ as a function of $an^2 + bn + c$.

Θ notation (theta)

- for a given function $g(n)$ denoted by $\Theta(g(n))$ the set of functions

- $\Theta(g(n)) = \{ f(n)$: there exist positive constant c_1, c_2 and n_0 such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0 \}$

- function $f(n)$ belongs to set $\Theta(g(n))$ if there exist positive constants c_1 and c_2 that is sandwiched between $c_1 g(n)$ and $c_2 g(n)$ for sufficient large n.

- As $\Theta g(n)$ is a set we write $f(n) \in \Theta(g(n))$ indicate $f(n)$ is a member of $g(n)$

- We use $f(n) = \Theta(g(n))$

* It gives tight bound on the growth rate of a function.

example - Show that $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

Positive constants c_1, c_2 and n_0 such that

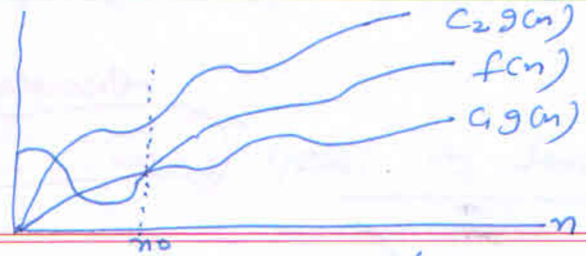
$c_1 n^2 \le \frac{1}{2}n^2 - 3n \le c_2 n^2$ for all $n \ge n_0$

Dividing by n^2

$c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2$

right hand inequality hold for any value of $n \ge 1$ by choosing $c_2 \ge 1/2$

left hand inequality hold for any value of $n \ge 7$ by choosing $c_1 \le 1/14$



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

classmate
Date
Page

By choosing $c_1 = 1/14$ $c_2 = 1/2$ $n_0 = 7$ we can verify

$$\frac{1}{2}n^2 - 3n = \Theta(n^2)$$

- other choice may exist but imp is some choice exists.

example - Show $6n^3 \neq \Theta(n^2)$

Let c_2 and n_0 exist such that $6n^3 \le c_2 n^2$ for all $n \ge n_0$

Divide by n^2

$n \le c_2/6$ which cannot possibly hold for arbitrarily large n since c_2 is constant.

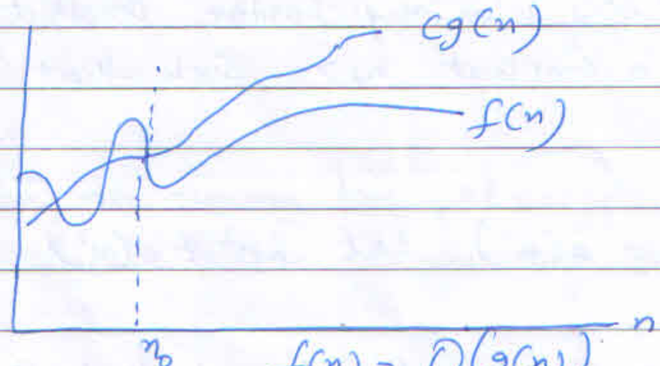
O notation (Big O)

- O notation provides an asymptotic upper bound
- for a given function $g(n)$ denote by $O(g(n))$ the set of functions

$$O(g(n)) = \{f(n): \text{there exist positive const } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le c g(n) \text{ for all } n \ge n_0\}$$

- $f(n) = O(g(n))$ indicate that a function $f(n)$ is a member of the set $O(g(n))$.
- $f(n) = \Theta(g(n))$ implies $f(n) = O(g(n))$ as Θ is stronger notation than O.

$$\Theta(g(n)) \subseteq O(g(n))$$



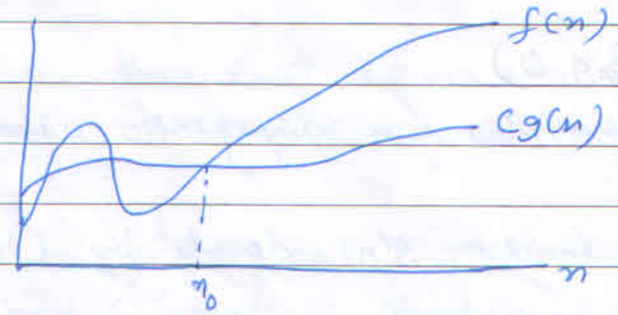
$f(n) = O(g(n))$

Ω notation (Big Omega)

- Ω notation provides an asymptotic lower bound.

- for a given function $g(n)$ denote by $\Omega(g(n))$ the set of functions
  $\Omega(g(n)) = \{ f(n)$: there exist positive constant c and n_0 such that $0 \le c g(n) \le f(n)$ for all $n \ge n_0 \}$

- for all values n at or to the right of n_0 the value of $f(n)$ is on or above $c g(n)$.



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

Note for any 2 functions $f(n)$ and $g(n)$ we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

o notation (little-oh)

- O notation denote an upper bound that may or may not asymptotically tight.

- for a given function $g(n)$ the set of function
  $o(g(n)) = \{ f(n)$: for any positive constant $c > 0$ there exists a constant $n_0 > 0$ such that $0 \le f(n) < c g(n)$ for all $n \ge n_0 \}$

- example $2n = o(n^2)$ but $2n^2 \ne o(n^2)$

- o notation denote an upper bound that is not asymptotically tight

Growth of function

- Diff between O notation and o notation is that in $f(n) = O(g(n))$ the bound $0 \le f(n) \le c\,g(n)$ holds for some constant $c > 0$ but in $f(n) = o(g(n))$ the bound $0 \le f(n) < c\,g(n)$ holds for all constant $c > 0$

ω-notation (little omega)

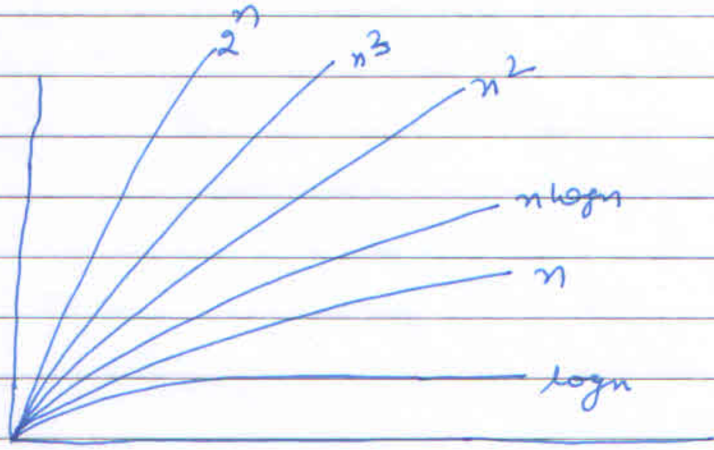
- ω notation denote a lower bound that is not asymptotically tight.

- for a given function $g(n)$ the set of function
  $\omega(g(n)) = \{ f(n) :$ for any positive constant $c > 0$ there exist a constant $n_0 > 0$ such that $0 \le c\,g(n) < f(n)$ for all $n > n_0 \}$

- eg $n^2/2 = \omega(n)$ but $n^2/2 \ne \omega(n^2)$

1) 2^{2^n}
2.) $n!$
3) 4^n
4) 2^n
5) n^2
6) $n \log n$
7) $\log(n!)$
8) n
9) $2^{\log n}$
10) $\log^2 n$
11) $\sqrt{\log n}$
12) $\log \log n$
13) 1

Increasing rate of Growth



Growth rate of Common functions

(AJAY RAWAT)