

# DATA STRUCTURES AND ALGORITHMS CSE220

Prof. Ramesh Ragala

July 29, 2015

- The **step count** for estimating Time Complexity is **inexact**.
- **Reason:** The step count is one for either instruction  $x=y$  or  $x = y+(x*y)$
- The running time for latter instruction is high
- So this step count is not very useful comparative purpose of algorithms.
- But Step Count method can be used to predict the running time growth of the algorithm for larger instance size accurately.
- Larger Instance Size  $\rightarrow$  asymptote as  $n$  approaches infinity.
- So, We can use this to predict the relative performance of two programs when the instance size becomes large.
- Notations are used to Characterise main factors, affecting the Running time of algorithm
  - These Notations are not considering detail examination of primitive operations in algorithm.
- **Asymptotic Notations are used to formalized.**

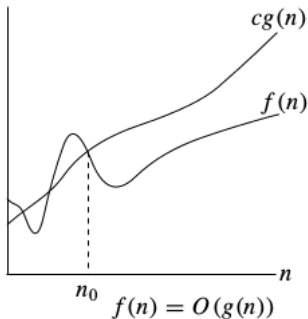
- Simply, Asymptotic Notations are used to formalize " An algorithm has running time or storage requirement that are **NEVER MORE THAN, ALWAYS GREATER THAN, or EXACTLY** some amount"
- There are five asymptotic notations are available
  - Big - Oh ( $O$ ) Notation.
  - Big - Omega ( $\Omega$ ) Notation.
  - Theta ( $\theta$ ) Notation.
  - Little -Oh ( $o$ ) Notation.
  - Little - Omega ( $\omega$ ) Notation.

# BIG OH (O) NOTATION

- **Definition:** " Let  $f(n)$  and  $g(n)$  are two positive functions, the function  $f(n) = O(g(n))$  [read as "f of n is big Oh of g of n " ] if and only if,  $\exists$  positive constants  $c$  and  $n_0$  such that  $f(n) \leq c g(n)$  for all  $n$ , where  $n \geq n_0$  .

- The Big-Oh notation allows
  - $f(n)$  is less than or equal to another function, up to a constant factor- $c$  and in the asymptotic sense as  $n$  grows towards infinity  $[n \rightarrow \infty]$ .
  - The growth rate of  $f(n)$  is always less than or equal to the growth rate of  $g(n)$  for large  $n$ -values.
  - The Big - Oh notation is used widely to characterize running times and space bounds in terms of some parameters -  $n$ .
  - $f(n)=O(g(n))$  states only that  $g(n)$  is an **upper bound** on the value of  $f(n)$  for all  $n$ , where  $n \geq n_0$ .
  - Big -Oh does not say about how good the bound is.

- Big -Oh Graph Representation:



- Rules of Big - Oh Notation:

- The constants are ignored. Example:  $O(100n)$  is write as  $O(n)$ . Here 100 is ignored.
- Lower order terms are also ignored. Example:  $O(2n^2+26n+39)$  is written as  $O(n^2)$ .

# BIG-OH NOTATION EXAMPLES

- Find the Big - Oh Representation of  $f(n) = 3n+2$ ?
- According to the definition of Big Oh Notation,  $f(n) \leq c g(n)$  for  $n \geq n_0$ .

$$3n + 2 \leq cn \quad \text{here } g(n) \text{ is } n$$

$$2 \leq c.n - 3n$$

$$2 \leq n(c - 3)$$

$$2/(c - 3) \leq n \quad \text{here we take } n \text{ as } n_0$$

$$n_0 \geq 2/(c - 3)$$

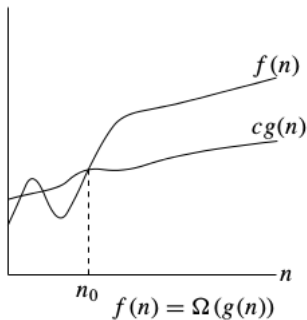
- According to the definition of Big Oh,  $c$  and  $n_0$  are positive constants.  $\rightarrow$  So we should choose  $c$ -value that leads  $n_0$  should be positive.
- So choose  $c$  has 4, that implies  $n_0$  has 2.
- So  $f(n) = 3n+2 = O(4n)$ . here we can neglect  $c$ -value in Big Oh representation.
- So  $f(n) = 3n+2 = O(n)$ , where  $n \geq 2$ .



# BIG OMEGA ( $\Omega$ ) NOTATION

- **Definition:** " Let  $f(n)$  and  $g(n)$  are two positive functions, the function  $f(n) = \Omega(g(n))$  [read as "f of n is big Omega of g of n" ] if and only if,  $\exists$  positive constants  $c$  and  $n_0$  such that  $f(n) \geq c g(n)$  for all  $n$ , where  $n \geq n_0$ ."
- $f(n) = \Omega(g(n))$  states only that  $g(n)$  is an **Lower bound** on the value of  $f(n)$  for all  $n$ , where  $n \geq n_0$ .

- Big -Omega Graph Representation:



# BIG-OMEGA NOTATION EXAMPLE

- Find the Big - Omega Representation of  $f(n) = 3n+2$ ?
- According to the definition of Big Omega Notation,  
 $f(n) \geq c g(n)$  for  $n \geq n_0$ .

$$3n + 2 \geq cn \quad \text{here } g(n) \text{ is } n$$

$$2 \geq c.n - 3n$$

$$2 \geq n(c - 3)$$

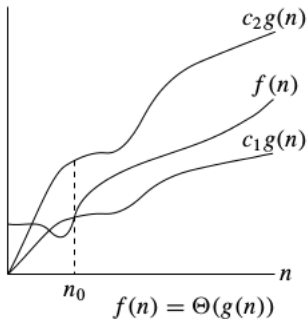
$$2/(c - 3) \geq n \quad \text{here we take } n \text{ as } n_0$$

$$n_0 \leq 2/(c - 3)$$

- According to the definition of Big Omega,  $c$  and  $n_0$  are positive constants.  
→ So we should choose  $c$ -value that leads  $n_0$  should be positive.
- So choose  $c$  has 3, that implies  $n_0$  starts from 0.
- So  $f(n) = 3n+2 = \Omega(3n)$ . here we can neglect  $c$ -value in Big Omega representation.
- So  $f(n) = 3n+2 = \Omega(n)$ , where  $n \geq 0$ .

- **Definition:** " Let  $f(n)$  and  $g(n)$  are two positive functions, the function  $f(n) = \theta(g(n))$  [read as "f of n is big Omega of g of n " ] if and only if,  $\exists$  positive constants  $c_1$ ,  $c_2$  and  $n_0$  such that  $c_1g(n) \leq f(n) \leq c_2 g(n)$  for all  $n$ , where  $n \geq n_0$ ".
- $g(n)$  is the an asymptotically tight bound for  $f(n)$ .

- Big -Omega Graph Representation:



# THETA NOTATION EXAMPLE

- Find the Theta Representation of  $f(n) = 3n+2$ ?
- Here we have calculate lower bound and upper bound of  $f(n) = 3n+2$ .
- In previous Slide we did estimate for the same function.
- So the lower Bound is  $3n+2 = 3n$ ,  $n \geq 0$ , where  $c_1 = 3$  and  $n_0$  starts from 0.
- The Upper Bound is  $3n+2 = 4n$ ,  $n \geq 2$ , where  $c_2 = 4$  and  $n_0$  starts from 2.
- Here we have to find the same  $n_0$  for lower and upper bound.
- Assume  $n_0$  is starting from 2.
- According to definition,  $c_1g(n) \leq f(n) \leq c_2 g(n)$  for all  $n$ , where  $n \geq n_0$ .
- $3n \leq 3n+2 \leq 3n+4$  for all  $n$ , where  $n \geq 2$ 
  - Here  $c_1 = 3$  and  $c_2 = 4$  and  $n_0 \geq 2$
  - So  $f(n) = 3n+2 = \theta(n)$  where  $c_1 = 3$  and  $c_2 = 4$  and  $n_0 \geq 2$