Advanced Analysis of Algorithm

Department of Computer Science
Swat College of Science & Technology

CS Course : Advanced Analysis of Algorithm
Course Instructor : Muzammil Khan

Class Task

- Group Task
  - Make group of 2 students
- Design an algorithm
  - That find
    - Maximum value &
    - Minimum value
      - In the array of size n
  - Also find the time complexity
    - Best, Worst and Average case running time

Extra Source

- http://homepages.ius.edu/RWISMAN/C455/html/notes/Chapter3/Asymptotic.htm

Chapter 4

Asymptotic Analysis

Order of Growth

- We require some simplifying assumptions
  - To ease our analysis
- We do this by ignoring the actual cost of each statement, and even the abstract costs
- Another simplifying assumption is that
  - It is only the rate of growth or order of growth of a function that interests us

Growth Functions – Algorithm Functions

- The behavior of algorithms are often describe by Standard Mathematical Functions
  - For different range of problems with
    - Different input size
  - Mostly referred as Growth Functions
- Growth Function classified as
  - Polynomial
    - Functions of positive powers of an integer
      - i.e. $f(n) = n^c$
        - Where c is positive constant
    - Examples are: $n^{0.5}$, n, n^2, n^5 etc…

Growth Functions (Cont...)

- Polylograrithmic
  - Functions are powers of logarithmic functions
    - i.e. $f(x) = (\log x)^c$
  - Examples are: $f(x) = (\log x)^{0.5}$, $\log x$, $(\log x)^2$, $(\log x)^5$ etc…
- Exponential
  - Functions are powers of a constant
    - i.e. $f(x) = C^{kn}$
      - Where k and c are constant
  - Examples are: $f(x) = 2^x$, 3^{4x} etc…

Growth Functions (Cont...)

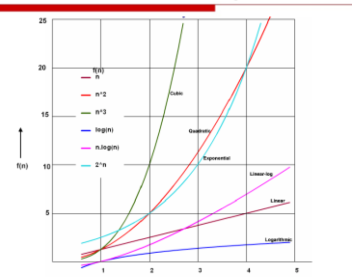
- Algorithms are classified as Polynomial, Logarithmic and Exponential based on
  - Running time expressed in terms of growth function
- Examples
  - An algorithm with running time $T(n) = 2^n$
    - Is said to be Exponential Algorithm
  - An algorithm with running time $T(n) = n^2$
    - Is said to be Polynomial Algorithm

Standard Growth Functions

- Standard Growth Functions use in Analysis of Algorithm are

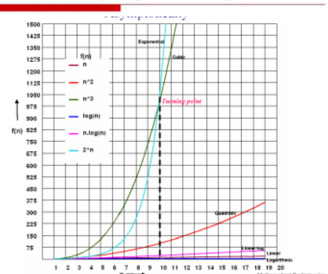
n	$lg\ n$	n	$nlg\ n$	n^2	n^3	2^n	$\sqrt{n}$
2	1	2	2	4	8	4	1.4
4	2	4	8	16	64	16	2
8	3	8	24	64	512	256	2.8
16	4	16	64	256	4,096	65,536	4
32	5	32	160	1,024	32,768	4,294,967,296	5.7
64	6	64	384	4,096	262,144	1.8×10^{19}	8
128	7	128	896	16,384	2,097,152	3.4×10^{38}	11
256	8	126	2,048	65,536	16,777,216	1.15×10^{77}	16
512	9	512	4,608	262,144	134,217,728	1.34×10^{154}	23

Growth Functions - Initially

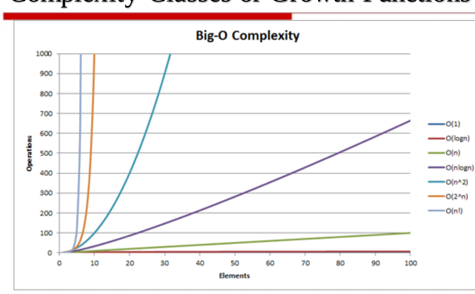


Growth Functions – Asymptotically

- Or Next Slide



Complexity Classes or Growth Functions



Growth Functions – Ranking

Algorithm	Class	Performance
log n	*Logarithm*	*Very good*
n	*Polynomial Linear*	*Good*
n log n	*n-logarithm*	*Fair*
n^2	*Polynomial Quadratic*	*Acceptable*
n^3	*Polynomial Cubic*	*Poor*
2^n	*Exponential*	*Bad*

Comparison of Algorithms

- 1000 means, 1000 instruction in 1 second and so on

Processing speed : 1 millisecond

		Maximum Problem Size		
Algorithm	Time Complexity	1 Second	1 Minute	1 Hour
A1	n	1000	6×10^4	3.6×10^6
A2	n lg n	140	4893	2.0×10^5
A3	n^2	31	244	1897
A4	n^3	10	19	153
A5	2^n	9	15	21

Asymptotic Analysis

- When we consider rate of growth
  - We need to consider only
    - The Leading Term of a formula
  - Since lower order terms are insignificant in comparison
- Consider
  - An *Algorithm1* with running time c_1n^2 and
  - Another *Algorithm2* with running time c_2nlgn
  - Even if c_2 is larger than c_1
    - Once *n* is large, *Algorithm1* is beaten by *Algorithm2*

Asymptotic Analysis (Cont...)

- We may try to determine the exact running time of an algorithm
  - But the extra precision is not worth the effort
- For large inputs
  - The multiplicative constants and lower order terms are dominated by effects of input size
- The input sizes are large enough to make only the order of growth of the running time relevant
  - We are studying asymptotic efficiency of algorithms
- We are concerned with
  - How running time of an algorithm increases with the size of the input

θ-Notation (Average Case Running Time)

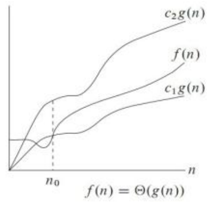
- Average case Running Time
  - Is the expected behavior
    - When the input is randomly drawn from a given distribution
  - Is an estimate of the running time for an "average" input
    - Computation of running time entails(involves) knowing all possible input sequences
      - The probability distribution of occurrence of these sequences
      - Often it is assumed that all inputs of a given size are equally likely
  - Represented by
    - θ-Notation (read as Theta)

θ-Notation (Cont...)

- If *f(n) is a growth function* for an algorithm
  - And *g(n) is a function* such that
    - For all positive real constants C_1, C_2 and for all $n \geq n_0$
  - Then $0 < C_2g(n) \leq f(n) \leq C_1g(n)$
    - $C_2g(n)$ is Lower bound and $C_1g(n)$ is Upper bound of $f(n)$
  - Then we say
    - $f(n) = \theta(g(n))$
      - Read as $f(n)$ in Theta of $g(n)$
- The behavior of $f(n)$ and $g(n)$ is shown
  - Follows that
    - For $n < n_0$, $f(n)$ is either above or below then $g(n)$

θ-Notation (Cont...)

- But for $n \ge n_0$, *f(n)* falls consistently between $C_1 g(n)$ and $C_2 g(n)$
- *g(n)* is said to be asymptotic tight bound for *f(n)*



θ-Notation (Cont...)

- There may be many functions for which *g(n)* is asymptotically tight bound
  - All such functions are said to be belonging to the group identified by *g(n)*
  - Symbolically we can denote the relationship as
    - f(n) ϵ θ(g(n))

θ-Notation Example

- $5n^2 - 19n \in \theta(n^2)$

Considering the ***upper bound***,

$5n^2 - 19n \le 5n^2 \quad for\ n \ge 0$

$5n^2 - 19n \le c_1 . n^2 \quad for\ n \ge n_1,\ where\ c_1 = 5\ and\ n_1 = 0$

Next, considering ***lower bound***,

$n \ge n/2 \quad for\ n \ge 1$ (Obvious !)

$n - 19/5 \ge 5n/(2 \times 19) \quad for\ n \ge 4$ (Divide right side by 19/5= 3.8)

$= 5n/38 \quad for\ n \ge 4$

$5n^2 - 19n \ge 25n^2/38 \quad for\ n \ge 4$ (Multiply both sides with $5n$)

$5n^2 - 19n \ge c_2 . n^2 \quad for\ n \ge n_2$ where $c_2 = 25/38$ and $n_2 = 4$

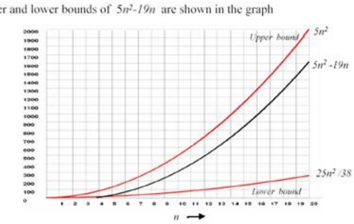
It follows, $\boldsymbol{0 < c_2 . n^2 \le 5n^2 - 19n \le c_1 . n^2 \quad for\ n \ge n_0,\ where\ n_0 = 4,\ c_1 = 5\ and\ c_2 = 25/38.}$

Therefore, $5n^2 - 19n \in \theta(n^2)$.

θ-Notation Example (Cont...)

- $5n^2 - 19n \in \theta(n^2)$

The upper and lower bounds of $5n^2 - 19n$ are shown in the graph



O-Notation (Worst Case Running Time)

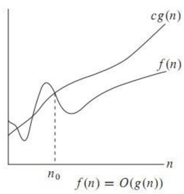
- Worst case Running Time
  - The behavior of the algorithm with respect to the worst possible case of the input instance
  - *Asymptotically* tight ***upper*** bound for f(n)
    - Cannot do worse
    - Can do better
    - *n* is the problem size
  - It gives us a guarantee that the algorithm will never take any longer
  - Represented by
    - O-Notation (read as big O)

O-Notation (Cont...)

- If *f(n) is a growth function* of an algorithm
  - And *g(n) is a function* such that
    - For some positive real constants C and for all $n \ge n_0$
  - Then
    - $0 < f(n) \le Cg(n)$
  - Then we say
    - $f(n) = O(g(n))$
      - Read as *f(n)* in Big-Oh of *g(n)*
- The behavior of *f(n)* and *g(n)* is shown
  - Follows that
    - For $n < n_0$, *f(n)* is either above or below then *g(n)*

O-Notation (Cont...)

- But for all $n \geq n_0$, *f(n)* falls consistently below *Cg(n)*
- The function *g(n)* is said to be asymptotic Upper bound for *f(n)*



O-Notation (Cont...)

- There may be many functions for which *g(n)* is asymptotically Upper bound
  - All such functions are said to be belonging to the group identified by *g(n)*
  - Symbolically we can denote the relationship as
    - $f(n) \in O(g(n))$

O-Notation Example $2n^2 \in O(n^3)$

- If $f(n) \leq cg(n)$, $c > 0$, $\forall n \geq n_0$ then $f(n) \in O(g(n))$

  $f(n) \leq cg(n)$ Definition of $O(g(n))$

  $2n^2 \leq cn^3$ Substitute

  $2n^2/n^3 \leq cn^3/n^3$ Divide by n^3

  Determine c

  $2/n \leq c$ if $n \rightarrow$ infinity then $2/n \rightarrow 0$
  $2/n$ maximum when $n=1$

  $0 \leq 2/1 \leq c = 2$ Satisfied by $c=2$

  Determine n_0

  $0 \leq 2/n_0 \leq 2$

  $0 \leq 2/2 \leq n_0$

  $0 \leq 1 \leq n_0 = 1$ Satisfied by $n_0=1$

  $0 \leq 2n^2 \leq 2n^3 \; \forall n \geq n_0=1$

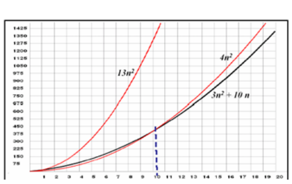
O-Notation Example (Cont...)

- $3n^2 + 10n \in O(n^2)$

O-Notation Example (Cont...)

- $3n^2 + 10n \in O(n^2)$

The graph depicts the two solutions. Observe that both $13n^2$ and $4n^2$ *eventually grow faster than* $3n^2+10$



O-Notation Example (Cont...)

- Show that $2n^2+n$ is in $O(n^2)$ by finding c and n_0
- Show that $1000n^2 + 50n$ is in $O(n^2)$ by finding c and n_0
- Show that n is in O(n lg n) by finding c and n_0
- Show that lg n is in O(n) by finding c and n_0
- Show that
  - $5n^3+10n \neq O(n^2)$
  - $2n^3+n \neq O(n^2)$
- Solve as many example you can

Ω -Notation (Best Case Running Time)

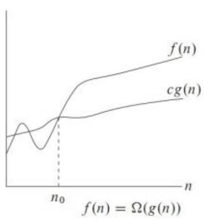
- Best case Running Time
  - The behavior of the algorithm with respect to the best possible case of the input instance
  - *Asymptotically* tight *Lower* bound for f(n)
    - Cannot do better
    - Can do worse
    - *n* is the problem size
  - It gives us a guarantee that the minimum time the algorithm will take
  - Represented by
    - Ω –Notation (read as big Omega)

Ω -Notation (Cont...)

- If *f(n) is a growth function* of an algorithm
  - And *g(n) is a function* such that
    - For some positive real constants *C* and for all $n \geq n_0$
  - Then
    - $0 < Cg(n) \leq f(n)$
  - Then we say
    - $f(n) = \Omega(g(n))$
      - Read as *f(n)* in Big-Omega of *g(n)*
- The behavior of *f(n)* and *g(n)* is shown
  - Follows that
    - For $n < n_0$, *f(n)* is either above or below then *g(n)*

Ω -Notation (Cont...)

- But for all $n \geq n_0$, *f(n)* falls consistently above *Cg(n)*
- The function *g(n)* is said to be asymptotic Lower bound for *f(n)*



Ω –Notation Example $3n^2 + n = \Omega(n^2)$

- If $cg(n) \leq f(n)$, $c > 0$ and $\forall\, n \geq n_0$, then $f(n) \in \Omega(g(n))$

$0 \leq cg(n) \leq h(n)$

$0 \leq cn^2 \leq 3n^2 + n$

$0/n^2 \leq cn^2/n^2 \leq 3n^2/n^2 + n/n^2$

$0 \leq c \leq 3 + 1/n$ $\quad$ $3+1/n = 3$

$0 \leq c \leq 3$ $\quad$ $\mathbf{c = 3}$

$0 \leq 3 \leq 3 + 1/n_0$

$-3 \leq 3-3 \leq 3-3 + 1/n_0$

$-3 \leq 0 \leq 1/n_0$ $\quad$ $\mathbf{n_0=1}$ satisfies

Ω –Notation Example 1

- $n^2 - 10n \in \Omega(n^2)$

$n \geq n/2$ *for* $n \geq 1$ (Obvious !)

$n-10 \geq n/(2 \times 10)$ *for* $n \geq 10$ (Divide right side by 10)

$= n/20$

$n^2 - 10n \geq n^2/20$ *for* $n \geq 10$ (Multiply both sides with *n* to maintain inequality)

$n^2 - 10n \geq c.\, n^2$ *for* $n \geq n_0$, where $c=1/20$ and $n_0=10$

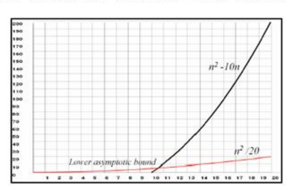
Therefore, $n^2 - 10n \in \Omega(n^2)$.

Observe that for $n \geq 10$, the function $n^2/20$ falls below the function n^2-10n

Ω –Notation Example 1 (Cont...)

- $n^2 - 10n \in \Omega(n^2)$

The behavior of functions n^2-10n and $n^2/20$ is shown in the graph.



Ω –Notation Example 2

- $3n^2 - 25n \in \Omega(n^2)$

$n \geq n/2$ *for* $n \geq 1$ (Obvious !)
$n - 25/3 \geq 3n/(2 \times 25)$ *for* $n \geq 9$ (Divide right side by $25/3 \approx 8.3$)
$= 3n/50$ *for* $n \geq 9$
$3n^2 - 25n \geq 9n^2/50$ *for* $n \geq 9$ (Multiply both sides with $3n$ to maintain inequality)
$3n^2 - 25n \geq c.n^2$ *for* $n \geq n_0$, where $c = 9/50$ and $n_0 = 9$

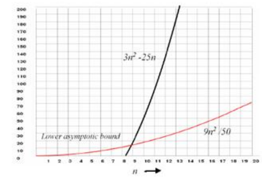
Therefore, $3n^2 - 25n \in \Omega(n^2)$.

Observe that for $n \geq 9$, the function $9n^2/50$ falls below the function $3n^2 - 25n$

Ω –Notation Example 2 (Cont...)

- $3n^2 - 25n \in \Omega(n^2)$

The behavior of functions $3n^2 - 25n$ and $9n^2/50$ is shown in the graph.



o – Notation, Small Oh

- If $f(n)$ is a growth function for an algorithm and $g(n)$ is some function such that
  - $0 \leq f(n) \leq C.g(n)$
    - For *all* $C > 0$ and *all* $n \geq n_0$
  - Then we say
    - $f(n) = o(g(n))$
      - Read as $f(n)$ in Small-Oh of $g(n)$
  - Symbolically
    - $f(n) \in o(g(n))$

o – Notation Example

- $n \in o(n^2)$

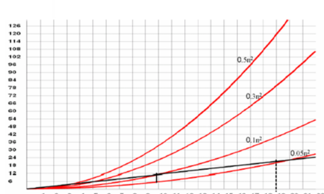
Consider, $n \leq c.n^2$ where c is ***any*** real constant
$1 \leq c.n$ (Dividing both side with n)
Or, $n \geq 1/c$
It follows that for all $c > 0$, we can find n_0 such that for all $n \geq n_0$, the above inequali holds true. For example, the ***some selections of c and corresponding*** $\boldsymbol{n_0}$ are
$c = 0.5$ *then* $n_0 = 3$; $c = 0.3$, *then* $n_0 = 5$; $c = 0.1$ *then* $n_0 = 10$; $c = 0.05$, *then* $n_0 = 20$

It follows that for all c we can find $n \geq n_0$ such that $n \leq c.n^2$
Therefore, $n \in o(n^2)$

o – Notation Example (Cont...)

- $n \in o(n^2)$

These possibilities are depicted in the graph.



ω - Notation, Small Omega

- If $f(n)$ is a growth function for an algorithm and $g(n)$ is some function such that
  - $f(n) \geq C.g(n)$ *or* $C.g(n) \leq f(n)$
    - For *all* $C > 0$ and *all* $n \geq n_0$
  - Then we say
    - $f(n) = \omega(g(n))$
      - Read as $f(n)$ in Small-Omega of $g(n)$
  - Symbolically
    - $f(n) \in \omega(g(n))$

Using Limit O - Notation

If f(n) and g(n) are growth functions such that

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = c, \quad \text{where } 0 \le c < \infty$$

then $f(n) \in O(g(n))$.

Example(1): $3n^2 + 5n + 20 \in O(n^2)$

$$\lim_{n \to \infty} \frac{3n^2 + 5n + 20}{n^2} = 3 + 5/n + 20/n^2 = 3$$

Therefore, $3n^2 + 5n + 20 \in O(n^2)$

Using Limit o - Notation

- *If f(n) and g(n) are growth functions such that*

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0,$$

then $f(n) \in o(g(n))$.

Example(1): $3n^2 + 5n \in o(n^3)$

$$\lim_{n \to \infty} \frac{3n^2 + 5n}{n^3} = 3/n + 5/n = 0$$

Therefore, $3n^2 + 5n \in o(n^3)$

Asymptotic Notation Constants

- If *C* is *a constant* then by convention
  - O(c) Є O(1)
  - θ (c) Є θ (1)
  - Ω(c) Є Ω(1)
  - The convention implies that
    - *The running time of an algorithm which does not depends on input size can be expressed in any of the above ways*
  - And
    - O(c.f(n)) Є O(f(n))
    - θ (c.f(n)) Є θ (f(n))
    - Ω(c.f(n)) Є Ω(f(n))
  - The relation ship implies that the *multiplier constant can be ignore*

Θ - O - Ω Relation

- ▪ If $f(n) \in \theta(g(n))$ then

  $f(n) \in \Omega(g(n))$, $f(n) \in O(g(n))$

  ▪ Conversely if $f(n) \in \Omega(g(n))$ *and* $f(n) \in O(g(n))$ then

  then $f(n) \in \theta(g(n))$

Example: Since, $n(n-1)/2 \in \theta(n^2)$, therefore

$n(n-1)/2 \in \Omega(n^2)$

$n(n-1)/2 \in O(n^2)$

o - O Relation

- ▪ If $f(n) \in o(g(n))$ then

  $f(n) \in O(g(n))$

  ▪ *Converse* is not true

That is, if $f(n) \in O(g(n))$, then $f(n) \notin o(g(n))$

Example(1): Since, $2^n \in o(n!)$, therefore

$2^n \in O(n!)$

Example (2): $n^2 + n \in O(n^2)$, but $n^2 + n \notin o(n^2)$

ω - Ω Relation

- ▪ If $f(n) \in \omega(g(n))$ then

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

  ▪ *Converse* is not true

That is, if $f(n) \in \omega(g(n))$, then $f(n) \notin \Omega(g(n))$

- Order Theorem

If $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$ then

$f_1(n) + f_2(n) \in O(\max(g_1(n), g_2(n)))$

End

- End of the Chapter

- You may have quiz next week