CS502-Summary-Chapter 1

1. What is Computer Science?

Computer Science is the science of

- algorithm processing
- representation
- storage and
- transmission

of information.

2. What is Algorithm?

An algorithm is a well defined computational procedure that converts input into output.

`Input -> Algorithm -> Output`

Example: a set of n real numbers

3. Why we need to study Algorithms?

Algorithm address issues related to: feasibility, effeciency & performance, and scalability.

- Study of algorithm enables us to determine, if a computer program is feasible, or infeasible.
- Effecient algorithm lead to an effecient computer program, & effecient use of hardware resource.
- Algorithm helps us to understand issues related to scalability.
- Analysis of algorithm provides a language for talking about program behavior.

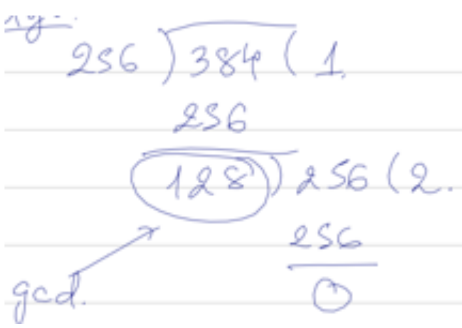
However, we should understand that computer program effeciency is only certain facet of overall computer resource usage.

3.1. Examples:

3.1.1. Greatest Common Divisor (gcd) & Euclidean Algorithm

```
gcd(2,4) = 2
gcd(10,5) = 10
gcd(27,112) = 1
gcd(56432,92431)=?
gcd(256,384)=128
```

Algorithm:



gcd(361, 190) = 19.

190) 361 (1
190
171) 190 (1
171
19) 171 (9
171
0

gcd

=> This is Euclidean algorithm. (2000 yrs old).

3.1.2. Method of Repeated Squaring

$2 * 2 = 2^2 = 4$ –> **1 multiplication**

$2 * 2 * 2 = 2^3 = 8$ –> **2 multiplications**

$2 * 2 * 2 * 2 * 2 * 2 = 2^6$ –> **1 multiplications**

=> With this behavior, $2^8 = 256$ (7 multiplications)

Instead of 7 multiplications, we can do it shorter with only 3 multiplications:

$2 \times 2 = 2^2 = 4$ 1 MULTIPLY.
$2^2 \times 2^2 = 2^4$ 1 "
$2^4 \times 2^4 = 256 = 2^8$ 1 "
3 " (1

So for 2^{32} it usually takes 31 multiplications now only needs 5 multiplications: $2^2, 2^4, 2^8, 2^{16}, 2^{32}$.

4. MATHS Notation:

- $\mathbb{P} = \{1, 2, 3, \dots\}$: Set of positive numbers
- $\mathbb{N} = \{0, 1, 2, 3, \dots\}$: Set of natural numbers
- $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$: Set of positive & negative and 0.
- $\mathbb{R} = (-\infty, +\infty)$: Set of real numbers
- $\mathbb{R}^+ = (0, +\infty)$: Set of positive real numbers
- $\mathbb{R}_0^+ = [0, +\infty)$: Set of real numbers ≥ 0
- $\mathbb{Q} = \{\frac{m}{n} | m, n \in \mathbb{Z}; n \neq 0\}$: Set of rational numbers

- $\mathbb{C}_0 = \{a + ib | a, b \in \mathbb{R}, i = \sqrt{-1}\}$: Set of complex numbers

5. Some important Facts

5.1. Logarithms

1. $logxy = logx + logy$
2. $log_a b * log_b a = 1$
3. $log_a x^y = ylog_a x$
4. $a^{log_b n} = n^{log_b a}$

5.2. GP

1. $S = a + ar + ar^2 + ar^{n-1}$
   $= \frac{a(1-r^n)}{1-r}; r \neq 1$
   $S = na; r = 1$
2. $S = a + ar + ar^2 + \dots$
   $= \frac{a}{1-r}; |r| < 1$

5.3. AP

1. $S = 1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$
2. $S = a + (a + d) + (a + 2d) + \dots + [a + (n-1)d]$
   $= [a + (a + (n-1)d)]\frac{n}{2}$

5.4. Calculus

1. $\frac{d}{dx}x^n = nx^{n-1}$
2. $\frac{d}{dx}a^x = a^x lna$

5.5. MATHS Interlude

1. $S = 1 + x + x^2 + \dots = \frac{1}{1-x}; |x| < 1$
   $S = \sum_{i=0}^{\infty} x^i = \frac{1}{1-x}; |x| < 1$
2. $\frac{dS}{dx} = 1 + 2x + 3x^2 + 4x^3 + \dots = \frac{1}{(1-x)^2}$
   $\frac{1}{(1-x)^2} = 1 + 2x + 3x^2 + 4x^3 \dots$
   $\rightarrow \frac{x}{(1-x)^2} = x + 2x^2 + 3x^3 + 4x^4 + \dots = \sum_{i=1}^{\infty} ix^i$

=> $\sum_{i=1}^{\infty} ix^i = \frac{x}{(1-x)^2}$; $|x| < 1$

5.6. Examples

Worst case running time as a function of n is

$4n^2 - 3n\log n + 17.5n - 43n^{2/3} + 75 = f(n)$.

LET $\boxed{n = 10^7}$

highest

(i) $4n^2 = 4 \times (10^7)^2 = 4 \times 10^{14}$

ii). $3n.\log n = 3(10^7).\log_{10} 10^7 = 21 \times 10^7$

iii) $17.5n = 17.5 \times 10^7$

iv) $43n^{2/3} < 43n = 43 \times 10^7$

v) $75 < 10^7$.

Big OH

for $n \to \infty$ $f(n) \simeq 4n^2 = O(n^2)$

ignore 4.

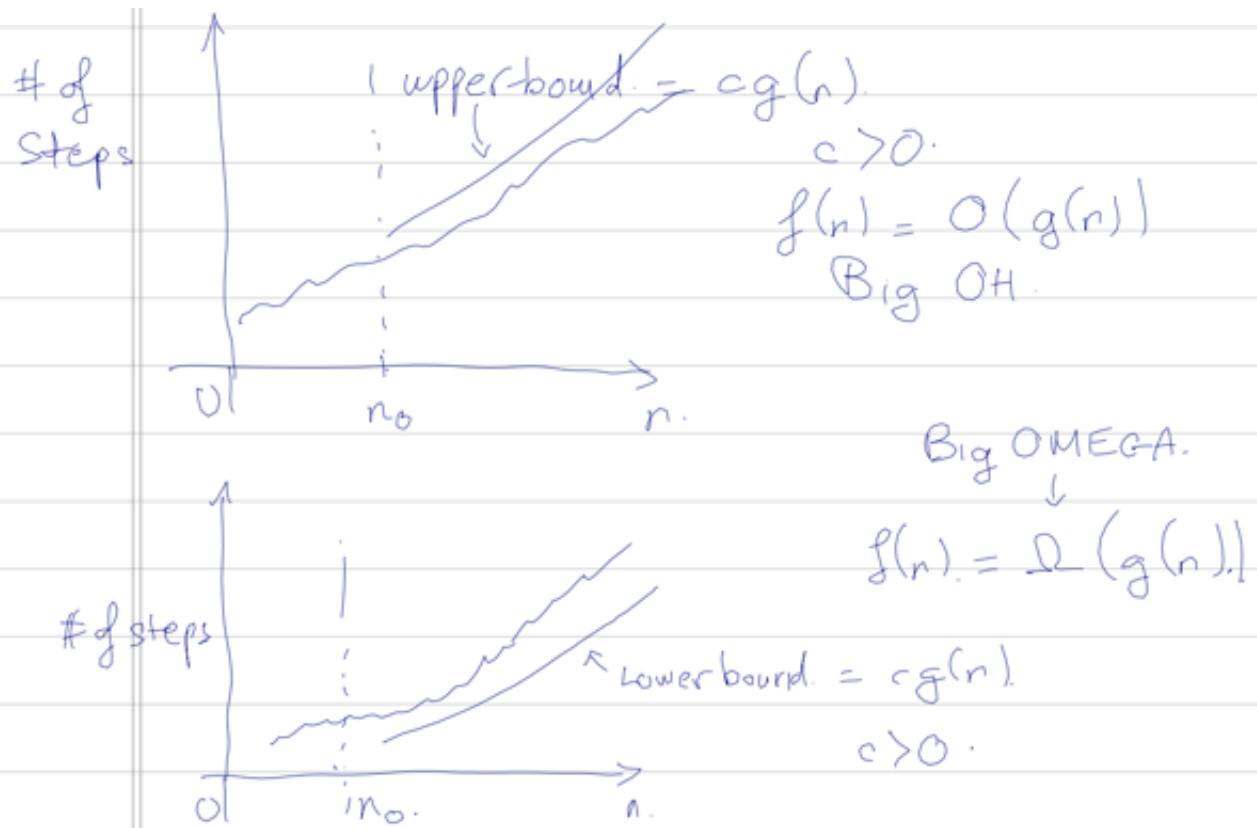
$\Rightarrow \boxed{f(n) \text{ is } \propto n^2 \text{ for large } n.}$

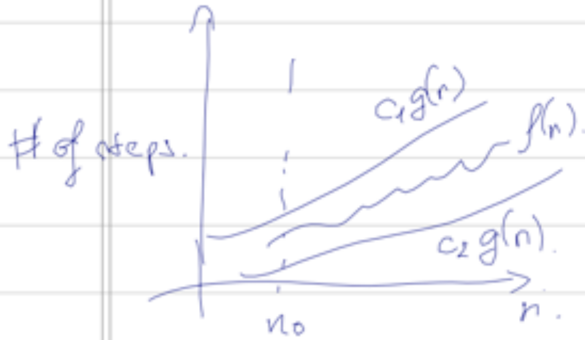
proportional to

6. Asymptotic Notation

> Asymptotic Notation are languages that allow us to analyze an algorithm's running time by identifying its behavior as the input size for the algorithm increases. This is also known as an algorithm's growth rate.

Computational Steps (fastest to slowest)	Example (N=100)
Constant – 1	1
Logarithmic - $Log(n)$	6.64
N-Linear	100
Log-Linear – $N * logn$	664
Quadratic - N^2	10,000
Cubic - N^3	100,000
Exponential – 2^n	127000000000….
Factorial - $N!$	3000000000000000000…





$c_1, c_2 > 0$.

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

BIG THETA

$f(n)$ = # of steps (EXACT) in the algorithm.

n is important parameter of algorithm.

LET $L = \lim\limits_{n \to \infty} \frac{f(n)}{g(n)}$

Let

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

Condition	Notation	Name
$0 \leq L < \infty$	O	Big O
$0 < L \leq \infty$	Ω	Big Omega
$0 < L < \infty$	Θ	Big Theta
$L = 0$	o	Small O
$L = +\infty$	ω	Small Omega

6.1. Problems

Indicate whether

- $f = O(g)$; or
- $f = \Omega(g)$; or
- $f = \Theta(g)$; or
- $f = o(g)$; or
- $f = \omega(g)$

6.1.1. Examples

1) $f(n) = (n-100)$; $g(n) = n-200$

SOLUTION: $L = \lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n-100}{n-200} = 1 = $ CONST

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

2) $f(n) = \sqrt{n}$; $g(n) = n^{2/3}$

SOLUTION:

$$L = \lim \frac{f(n)}{g(n)} = \frac{\sqrt{n}}{n^{2/3}} = \frac{n^{1/2}}{n^{2/3-1/2}} = \lim_{n\to\infty} \frac{1}{n^{1/6}}$$

$= 0$. (small oh).

$\therefore$ $f(n) = o(g(n))$

3) $f(n) = 100n + \log n$; $g(n) = n + (\log n)^2$.

$$L = \lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{100n + \log n}{n + (\log n)^2}$$

n is much bigger than $\log n$.

$$= \lim_{n\to\infty} \frac{100n}{n} = 100$$

Constant

ignore $\log n$ & $(\log n)^2$.

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

4). $f(n) = n \log n$, $g(n) = 10n \log(10n)$.

$$L = \lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n \log n}{10n \log(10n)} = \lim_{n\to\infty} \frac{n \log n}{10n(\log 10 + \log n)}$$

$$= \lim_{n\to\infty} \frac{\log n}{\underset{\substack{\uparrow \\ \text{ignore}}}{10 \log 10} + 10 \log n} = \frac{1}{10} \text{ (constant)}$$

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

5) $f(n) = \log(2n)$, $g(n) = \log(3n)$

$f(n) = \log 2n = \log 2 + \log n$.

$g(n) = \log 3n = \log 3 + \log n$.

$$\Rightarrow L = \lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{\log 2 + \log n}{\log 3 + \log n} = 1$$

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

6) $f(n) = 10 \log n$, $g(n) = \log n^2$.

$\rightarrow g(n) = 2 \log n$.

$$L = \lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{10 \log n}{2 \log n} = 5 \text{ constant.}$$

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

7) $f(n) = n^{1.01}$, $g(n) = n(\log n)^2$

$$L = \lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{n^{1.01}}{n(\log n)^2} = \frac{n \cdot n^{0.01}}{n \cdot (\log n)^2}$$

$$= \lim_{n\to\infty} \frac{n^{0.01}}{(\log n)^2} = \infty$$ (n is much bigger than $\log n$)

$\therefore f(n) = \omega(g(n))$

8) $f(n) = \frac{n^2}{\log n}$, $g(n) = n(\log n)^2$

$$L = \lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^2}{\log n \cdot n(\log n)^2} = \frac{n}{(\log n)^3} = \infty$$

$\therefore L = \omega(g(n))$

9) $f(n) = n^{0.1}$, $g(n) = (\log n)^{10}$

$$L = \lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^{0.1}}{(\log n)^{10}} = \infty$$ (n is much bigger than $\log n$)

$\Rightarrow f(n) = \omega(g(n))$ $\boxed{\text{USE L'HOSPITAL'S RULE}}$

LOPITAL'S RULE

10). $f(n) = (\log n)^{\log n}$, $g(n) = \dfrac{n}{\log n}$

$$L = \lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{(\log n)^{\log n} \cdot \log n}{n}$$

$$= \lim_{n\to\infty} \frac{(\log n)^{(1+\log n)}}{n} = 0$$

$$\Rightarrow f(n) = o(g(n))$$

wrong

<u>Sol:</u> $n = 10^m$, $\log n = \log 10^m = m$

$$f(n) = m^m, \quad g(n) = \frac{n}{m} = \frac{10^m}{m}$$

$$L = \lim_{n\to\infty} \frac{m^m \cdot m}{10^m} = \lim_{n\to\infty} \left(\frac{m}{10}\right)^m m \to \infty$$

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

7. Assignments

Assignment 1. The following problems are from. Introduction to Algorithms, by CLRS

Points	Second Ed.	Third Ed
10	Page 13: 1.2-2.	Page 14: 1.2-2
10	Pag 13: 1.2-3	Page 14: 1.2-3.

Hint: Use EXCEL spreadsheet

Answers $\begin{cases} 2 \leq n \leq 43 \\ n = 15 \end{cases}$