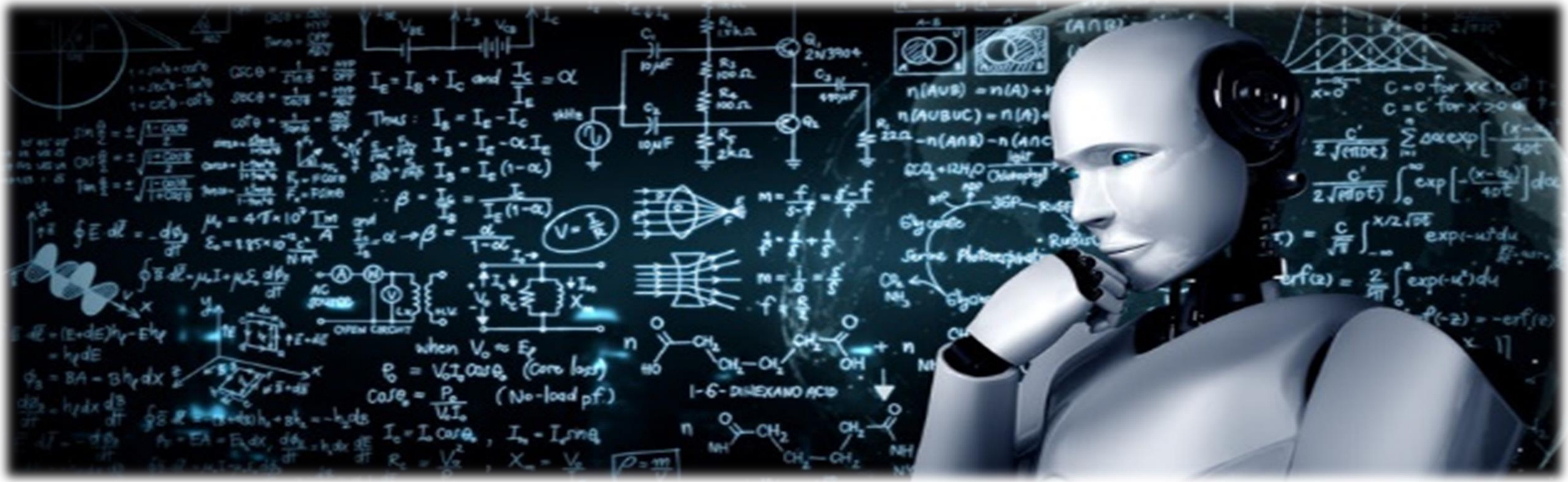


Lecture 02 - Introduction to Asymptotic Notations

Design and Analysis of Algorithms – IT1205

Year 02 Semester 02

SCU
SLIIT CITY UNI

Lecture Contents

- Asymptotic Notations
  - ❑ O Notation
  - ❑ θ Notation
  - ❑ Ω Notation
- Selection Sort Algorithm
- Bubble Sort Algorithm

Asymptotic Notations

Asymptotic notations are the **mathematical notations** used to describe the running time of an algorithm. It used when the input tends towards a particular value or a limiting value.

Why we need of Asymptotic Notation?

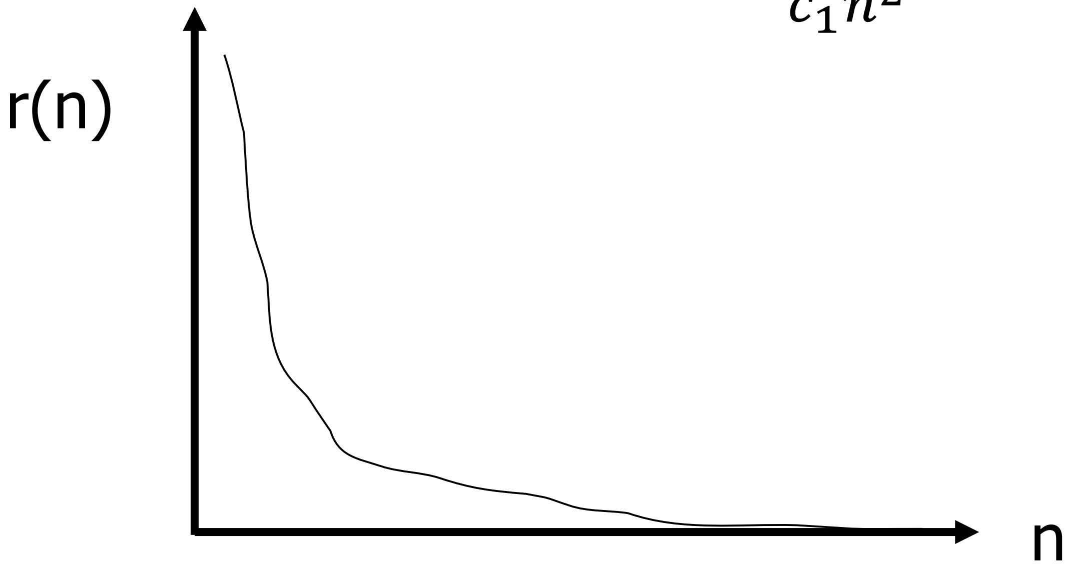
- Ignore machine dependent constants.
- RAM Model have some problems.
- Exact analysis is very complicated
- Sufficiently large size of n.
- Growth of T(n) as n --> ∞

Asymptotic Notations (Cont.)

Step count is determined to be

$$\mathbf{c_1n^2 + c_2n + c_3,} \qquad c_1 > 0$$

Let's take the ratio $r(n) = \frac{c_2n+c_3}{c_1n^2}$



When n is large r(n) tends to zero.

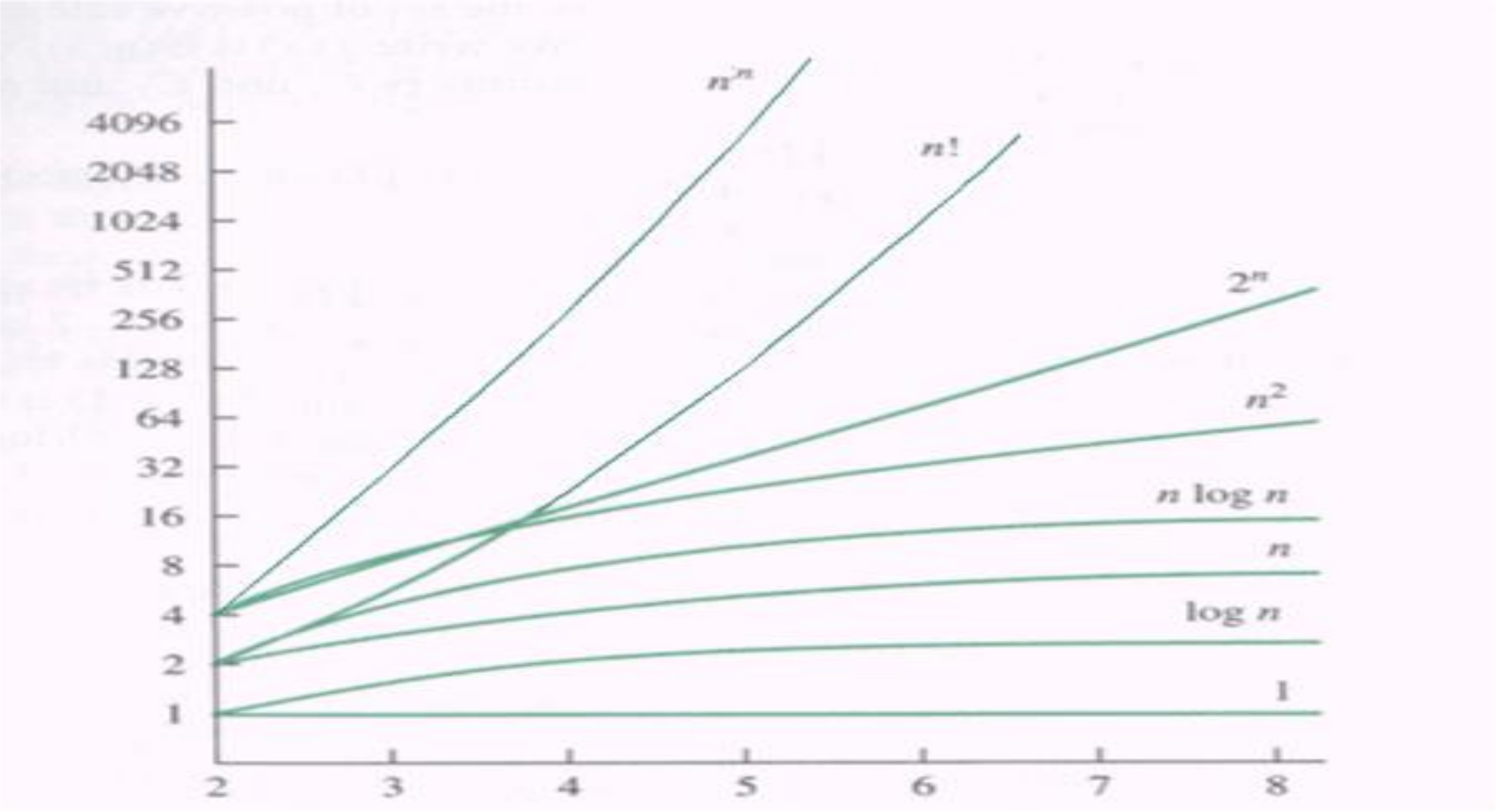
Asymptotic Notations(Contd.)

Suppose that programs A and B perform the same task. Assume that one person has determined the step counts of these programs to be $\mathbf{t_A(n)=2n^2+3n}$ and $\mathbf{t_B(n)=13n}$.

- Which program is the faster one ?
- What is the answer ,if the step count of the program B is 2^n+n^2 ?

Graphs of functions

$N^n > n! > 2^n > n^3 > n^2 > n \log a > n > \log n > 1$



Asymptotic Notations(Contd.)

There are three notations.

O - Notation

Θ - Notation

Ω - Notation

Asymptotic Notations (Contd.)

- Focus on what's important by abstracting away low-order terms and constant factors.
- How we indicate running times of algorithms.
- A way to compare "sizes" of functions:
  - $O \approx \leq$ -- Consider the **Upper Bound**
  - $\Theta \approx \geq$ --**Consider the Both(Average)**
  - $\Omega \approx =$ -- Consider the **Lower Bound**

Big O - Notation

- Introduced by Paul Bechman in 1892.
- We use Big O-notation to give an upper bound on a function. And consider as the **Worst Case Scenarios.**

Definition:

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

Eg: What is the big O value of f(n)=2n + 6 ?

g(n)=n therefore **f(n)=O(n)**

$\mathbf{a_n\, x^n + \ldots + a_1\, x + a_0}$ is $\mathbf{O(x^n)}$ for any real numbers $\mathbf{a_n, \ldots, a_0}$ and any nonnegative number **n .**

Big O – Notation(Contd.)

Assignment ($s \leftarrow 1$)

Addition ($s+1$)

Multiplication ($s*2$)

Comparison ($S<10$)

O(1)

Big O – Notation(Contd.)

Find the Big Oh value for following fragment of code.

for i $\leftarrow$ 1 to n ----------------- n Times

for j $\leftarrow$ 1 to i ----------------- (n-1) Times

Print j ----------------- (n-2) Times

$n * (n^2 - 3n + 2) = n^3 - 3n^2 + 2n$

$O(n^3)$

Big O – Notation(Contd.)

Find the Big O value for the following functions.

(i) $T(n) = 3 + 5n + 3n^2$

(ii) $f(n) = 2^n + n^2 + 8n + 7$

(iii) $T(n) = n + \log n + 6$

Answers:

(i) $O(n^2)$

(ii) $O(2^n)$

(iii) $O(n)$

Back to the example

Alternative calculation:

	cost	times
$sum \leftarrow 0$	c_1	1
for $i \leftarrow 1$ to n	c_2	$n+1$
$\quad sum \leftarrow sum + A[i]$	c_3	n

$T(n) = c_1 + c_2\,(n+1) + c_3\,n = (c_1 + c_2) + (c_2 + c_3)\,n = c_4 + \mathbf{c_5}\,\boldsymbol{n}$

$\rightarrow O\,(\mathrm{n})$

Proof: $c_4 + c_5\,n \leq c\,n \rightarrow$ TRUE for $n \geq 1$ and $c \geq c_4 + c_5$

Ω - Notation

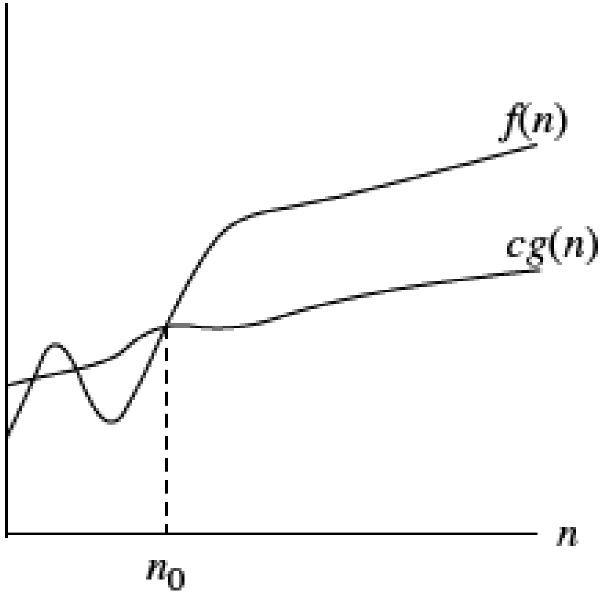
Which will provide the **lower bound** of the function. And consider as the **Best Case Scenarios**.

Definition:

$\Omega(g(n))$ = { f(n) : there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_o$}

Ω-notation

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



$g(n)$ is an ***asymptotic lower bound*** for $f(n)$.

Θ - Notation

This is used when the function *f* can be bounded both from above and below by the same function *g*.

Definition:

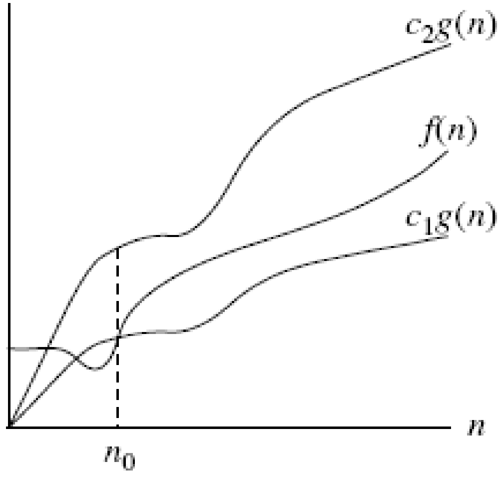
$\Theta(g(n))$ ={ f(n): there exist positive constant c_1, c_2, and n_0 such that $0 \leq c_1\, g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$ }

Θ - Notation (Cont.)

Θ-notation

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

Lecture Notes for Chapter 3: Growth of Functions



$g(n)$ is an ***asymptotically tight bound*** for $f(n)$.

Analysis of Selection Sort Algorithm

- This is an another efficient algorithm for *sorting small number of elements.*
- *Selection Sort Algorithm consist of 5 main steps.*
    1. *Initialize the "min" as leftmost element*
    2. *Search the minimum value in the list*
    3. *Swap with leftmost value and minimum value*
    4. *leftmost "min" incremented by 1, to go for next occurance*
    5. *Repeat the process until the numbers are sorted*

Pseudocode for Selection Sort

```
Selection-SORT(A)
for i = 1 to n - 1
    min = i

        for j = i+1 to n
            if A[j] < A[min] then
                min = j;
            end if
        end for

        swap A[min] and A[i]
    end for
```

Analysis of Bubble Sort Algorithm

- *Bubble Sort Algorithm consist of 5 main steps.*
    1. *Start with the first element.*
    2. *Compare the current element with the next element.*
    3. *If the current element is greater than the next element, then swap both the elements. If not, move to the next element.*
    4. *Repeat steps 1 – 3 until we get the sorted list.*

Pseudocode for Bubble Sort

```
Bubble-SORT(A)
for i = 1 to n - 1
        for j = 1 to n-1
            if A[j] > A[j+1] then
                swap A[j] and A[j+1]
            end if
        end for
end for
```

Activity

Convert this number set into Ascending Order using,

- Selection Sort
- Bubble Sort

1.

3	9	7	4	1	5

2.

5	3	1	4	7	6

Summary

Asymptotic Notations

- ❑ O Notation
- ❑ θ Notation
- ❑ Ω Notation

Selection Sort Algorithm

Bubble Sort Algorithm


SEE YOU
next
week!