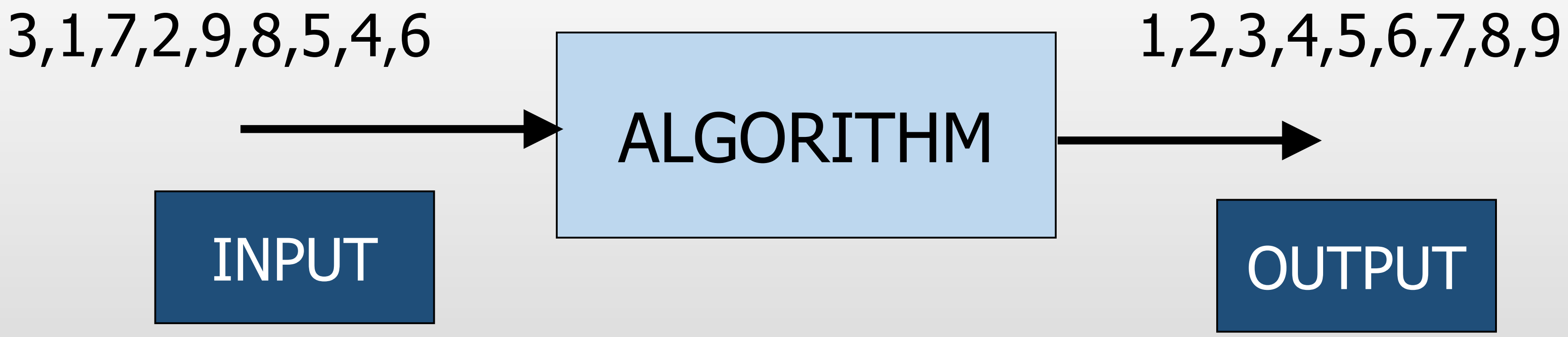
IT2070 – Data Structures and Algorithms

Lecture 05

Introduction to Algorithms

ALGORITHMS

- Algorithm is any well defined computational procedure that takes some value or set of values as input and produce some value or set of values as output.

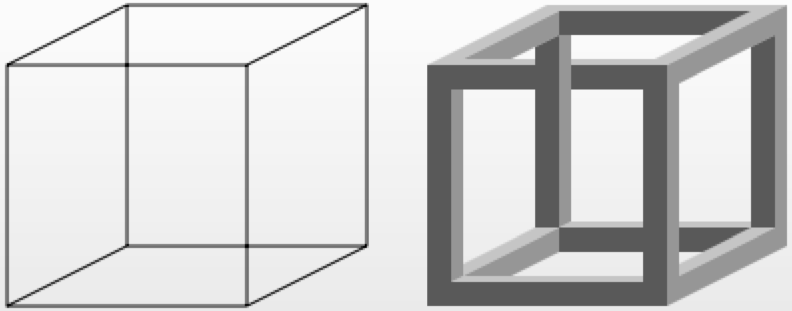


ALGORITHM (Contd.)

1. Get the smallest value from the input.
2. Remove it and output.
3. Repeat above 1,2 for remaining input until there is no item in the input.

Properties of an Algorithm.

- Be correct.
- Be unambiguous.
- Give the correct solution for all cases.
- Be simple.
- It must terminate.



Necker_cube_and_impossible_cube

Source:http://en.wikipedia.org/wiki/Ambiguity#Mathematical_interpretation_of_ambiguity

Applications of Algorithms

- Data retrieval
- Network routing
- Sorting
- Searching
- Shortest paths in a graph

Pseudocode

- Method of writing down a algorithm.
- Easy to read and understand.
- Just like other programming language.

  - More expressive method.
  - Does not concern with the technique of software engineering.

Pseudocode Conventions.

- English.
- Indentation.
- Separate line for each instruction.
- Looping constructs and conditional constructs.
- // indicate a comment line.
- = indicate the assignment.

Pseudocode Conventions.

- Array elements are accessed by specifying the array name followed by the index in the square bracket.
- The notation ".." is used to indicate a range of values within the array.

  Ex:

  A[1..i] indicates the sub array of A consisting of elements A[1] , A[2] , .. , A[i].

Analysis of Algorithms

Idea is to predict the resource usage.

- Memory
- Logic Gates
- **Computational Time**

Why do we need an analysis?

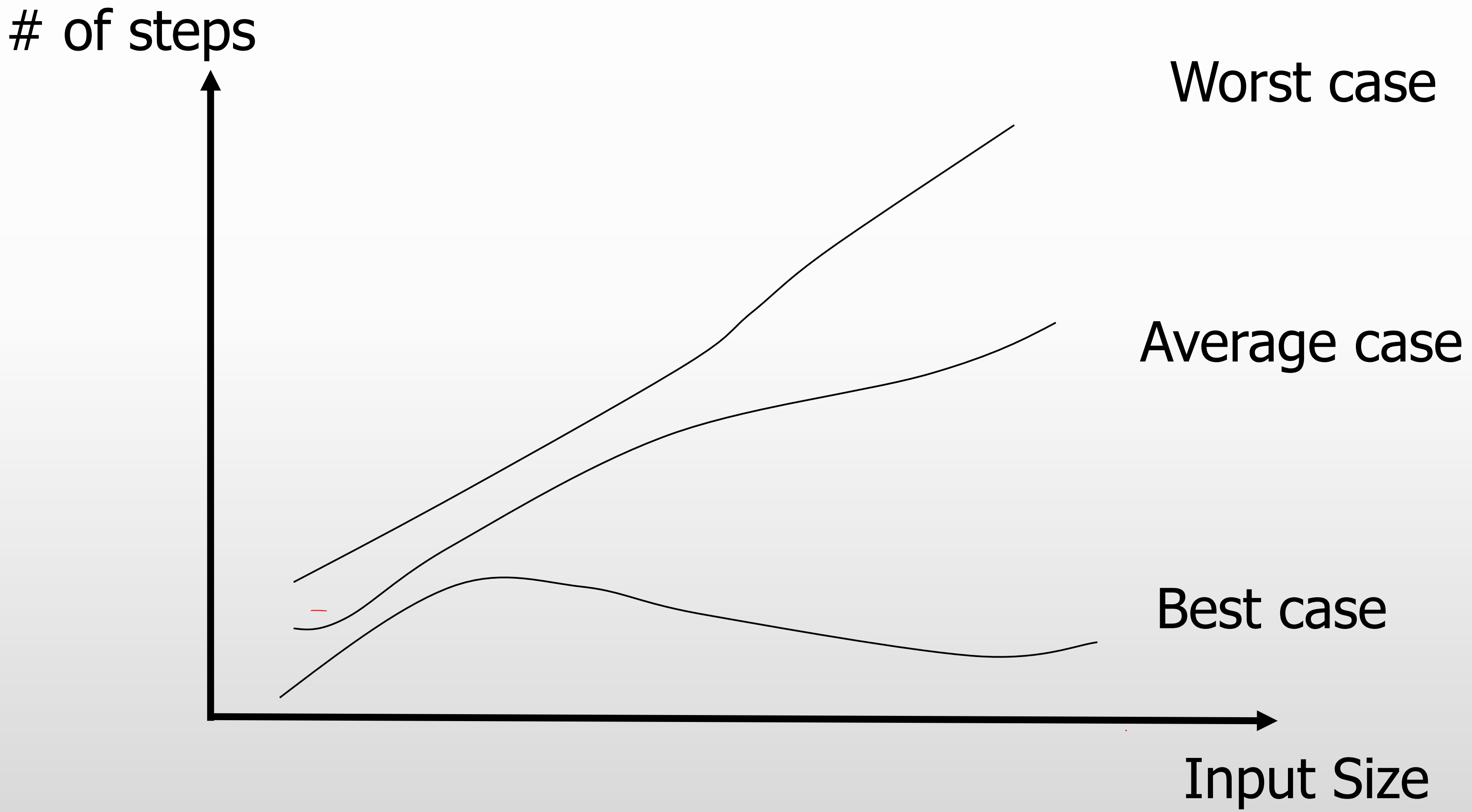
- To compare
- Predict the growth of run time

Worst, Best and Average case.

Running time will depend on the chosen instance characteristics.

- **Best case**:
  Minimum number of steps taken on any instance of size n.
- **Worst case:**
  Maximum number of steps taken on any instance of size n.
- **Average case:**
  An average number of steps taken on any instance of size n.

Worst,Best and Average case(Contd.)



Analysis Methods

- Operation Count Methods
- Step Count Method(RAM Model)
- Exact Analysis
- Asymptotic Notations

Operation count

- Methods for time complexity analysis.
- Select one or more operations such as add, multiply and compare.
- Operation count considers the **time spent on chosen operations** but not all.

Step Count (RAM Model)

- Assume a generic one processor.
- Instructions are executed one after another, with no concurrent operations.
- +, - , =, it takes exactly one step.
- Each memory access takes exactly 1 step.
- **Running Time = Sum of the steps.**

SLIIT Discover Your Future

RAM Model Analysis.

Example1:

n = 100	1step
n = n + 100	2steps
Print n	1step

Steps = 4

Example2:

$$sum = 0$$

1 assignment

$$\text{for } i = 1 \text{ to } n$$

n+1 assignments
n+1 comparisons
n additions

$$sum = sum + A[i]$$

n assignments
n additions
n memory accesses

Steps = 6n+3

Question 01

- Using RAM model analysis, find out the no of steps needed to display the numbers from 1 to 10.

i = 1 → 1 step

While i <=10 → 11 steps

print i → 10 steps

i = i + 1 → 10 + 10 = 20 steps

Steps = 42

Question 02

- Using RAM model analysis, find out the no of steps needed to display the numbers from 10 to 20.

i = 10 → 1 step

While i <= 20 → 12 steps (Hint :20 – 10 + 2 = 12)

print i → 11 steps

i = i + 1 → 11 + 11 = 22 steps

Steps = 46

Question 03

- Using RAM model analysis, find out the no of steps needed to display the even numbers from 10 to 20.

for i = 10 to 20 → (12+ 12 + 11) steps = 35 steps

    if i % 2 == 0 → 2 * 11 = 22 steps

        print i → 6 steps

Steps = 63

Problems with RAM Model

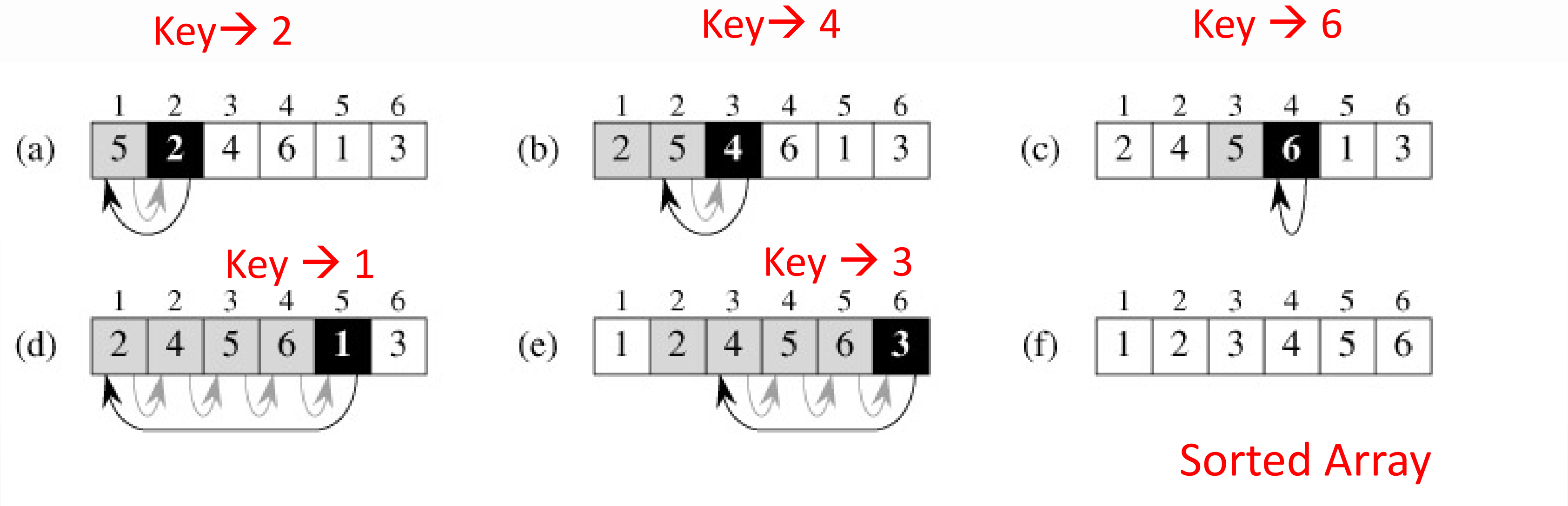
- Differer number of steps with different architecture.

  eg: sum = sum + A[i]   is a one step in the CISC processor.

- It is difficult to count the exact number of steps in the algorithm.

  eg: See the insertion sort , efficient algorithm for sorting small number of elements.

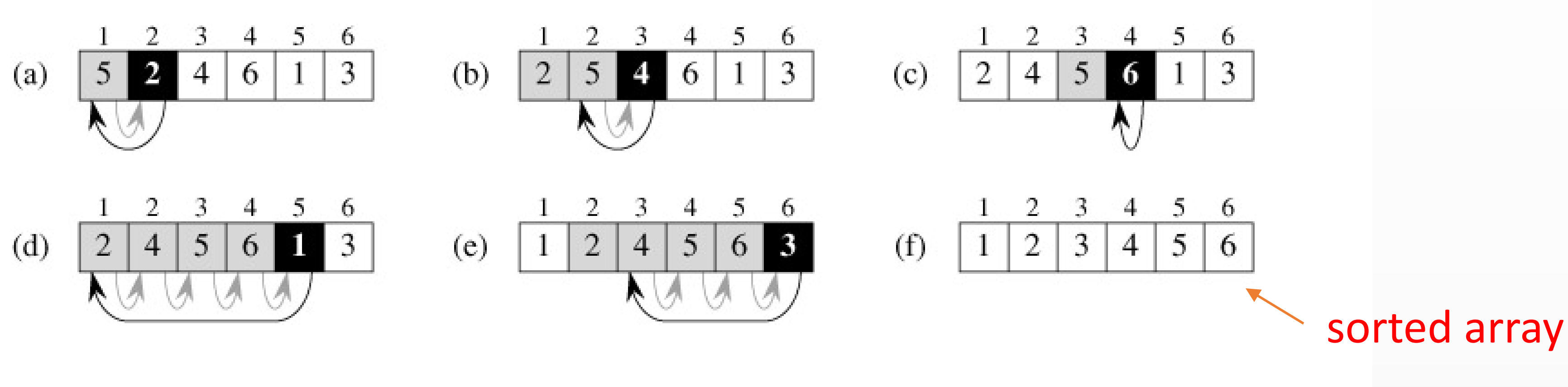
Insertion sort



Pseudocode for insertion sort.

```
INSERTION-SORT(A)
for j = 2 to A.length
        key = A[j]
        // Insert A[j] into the sorted sequence A[1..j-1]
        i = j - 1
        While i > 0 and A[i] > key
                A[i+1] = A[i]
                i = i-1
        A[i+1]  = key
```

Insertion sort - Example



- *(a)-(e)* The iterations of the *for* loop → lines 1-8.
- In each iteration, the black rectangle holds the key taken from *A*[*j*],
- Key is compared with the values in shaded rectangles to its left → line 5.
- Shaded arrows show array values moved one position to the right → line 6,
- Black arrows indicate where the key is moved to → line 8.

Exact analysis of Insertion sort

- Time taken for the algorithm will depend on the input size (number of elements of the array)

Running Time (Time complexity):

This is the number of primitive operations or steps executed through an algorithm given a particular input.

Running Time : T(n)

	INSERTION-SORT(A)	Cost	Times
1	**for** j = 2 **to** A.length	c_1	n
2	key = A[j]	c_2	n-1
3	// Insert A[j] into the sorted // sequence A[1..j-1]	0	n-1
4	i = j – 1	c_4	n-1
5	**While** i > 0 **and** A[i] > key	c_5	$\sum_{j=2}^{n} t_j$
6	A[i+1] = A[i]	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	i = i-1	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8	A[i+1] = key	c_8	n-1

i^{th} line takes time c_i where c_i is a constant.

For each j=2,3,…,n , t_j be the number of times the while loop is executed for that value of j

Running Time(contd.)

$$T(n) = c_1 n + c_2 (n-1) + c_4 (n-1) + c_5 \sum_{j=2}^{n} t_j$$

$$+ c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1)$$

- Best Case (Array is in sorted order)
  - $T(n) \rightarrow an+b$
- Worst Case (Array is in reverse sorted order)
  - $T(n) \rightarrow cn^2 + dn + e$

Worst Case T(n) $\rightarrow cn^2 + dn + e$

Worst case: The array is in reverse sorted order.

- Always find that $A[i] > key$ in while loop test.
- Have to compare *key* with all elements to the left of the jth position $\Rightarrow$ compare with $j-1$ elements.
- Since the while loop exits because i reaches 0, there's one additional test after the $j-1$ tests $\Rightarrow t_j = j$.
- $\sum_{j=2}^{n} t_j = \sum_{j=2}^{n} j$ and $\sum_{j=2}^{n} (t_j - 1) = \sum_{j=2}^{n} (j-1)$.
- $\sum_{j=1}^{n} j$ is known as an ***arithmetic series***, and equation (A.1) shows that it equals $\frac{n(n+1)}{2}$.

SLIIT Discover Your Future

Worst Case $T(n) \rightarrow cn^2 + dn + e$

- Since $\sum_{j=2}^{n} j = \left(\sum_{j=1}^{n} j\right) - 1$, it equals $\frac{n(n+1)}{2} - 1$.
  [The parentheses around the summation are not strictly necessary. They are there for clarity, but it might be a good idea to remind the students that the meaning of the expression would be the same even without the parentheses.]
- Letting $k = j - 1$, we see that $\sum_{j=2}^{n}(j-1) = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$.
- Running time is

$$\begin{aligned} T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\ &\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\ &= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right)n \\ &\quad - (c_2 + c_4 + c_5 + c_8)\ . \end{aligned}$$

- Can express $T(n)$ as $an^2 + bn + c$ for constants a, b, c (that again depend on statement costs) $\Rightarrow T(n)$ is a *quadratic function* of n.

Asymptotic Notations

- RAM Model has some problems.
- Exact analysis is very complicated.

Therefore we move to **asymptotic notation**

- Here we focus on determining the biggest term in the complexity function.
- Sufficiently large size of n.

Asymptotic Notations(Contd.)

- There are three notations.

O - Notation

Θ - Notation

Ω - Notation

Big O - Notation

- Introduced by Paul Bechman in 1892.
- We use Big O-notation to give an upper bound on a function.

Definition:

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

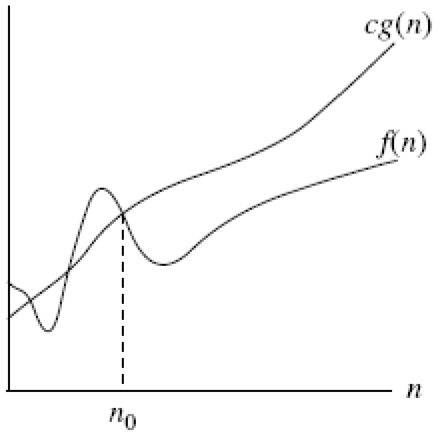
Eg: What is the big O value of f(n)=2n + 6 ?

c = 4

n_o = 3

g(n)=n therefore

f(n) = O(n)



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

If $f(n) \in O(g(n))$, we write $f(n) = O(g(n))$

SLIIT Discover Your Future

Back to the example

- Alternative calculation:

	cost	times
sum = 0	c_1	1
for *i* = 1 to *n*	c_2	n+1
sum = sum + A[*i*]	c_3	n

$$T(n) = c_1 + c_2\,(n+1) + c_3\,n$$

$$= (c_1 + c_2) + (c_2 + c_3)\,n$$

$$= c_4 + c_5\,n \quad \rightarrow O\,(n)$$

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

Big O – Notation(Contd.)

Assignment (s = 1)

Addition (s+1)

Multiplication (s*2)

Comparison (S<10)

O(1)

Question

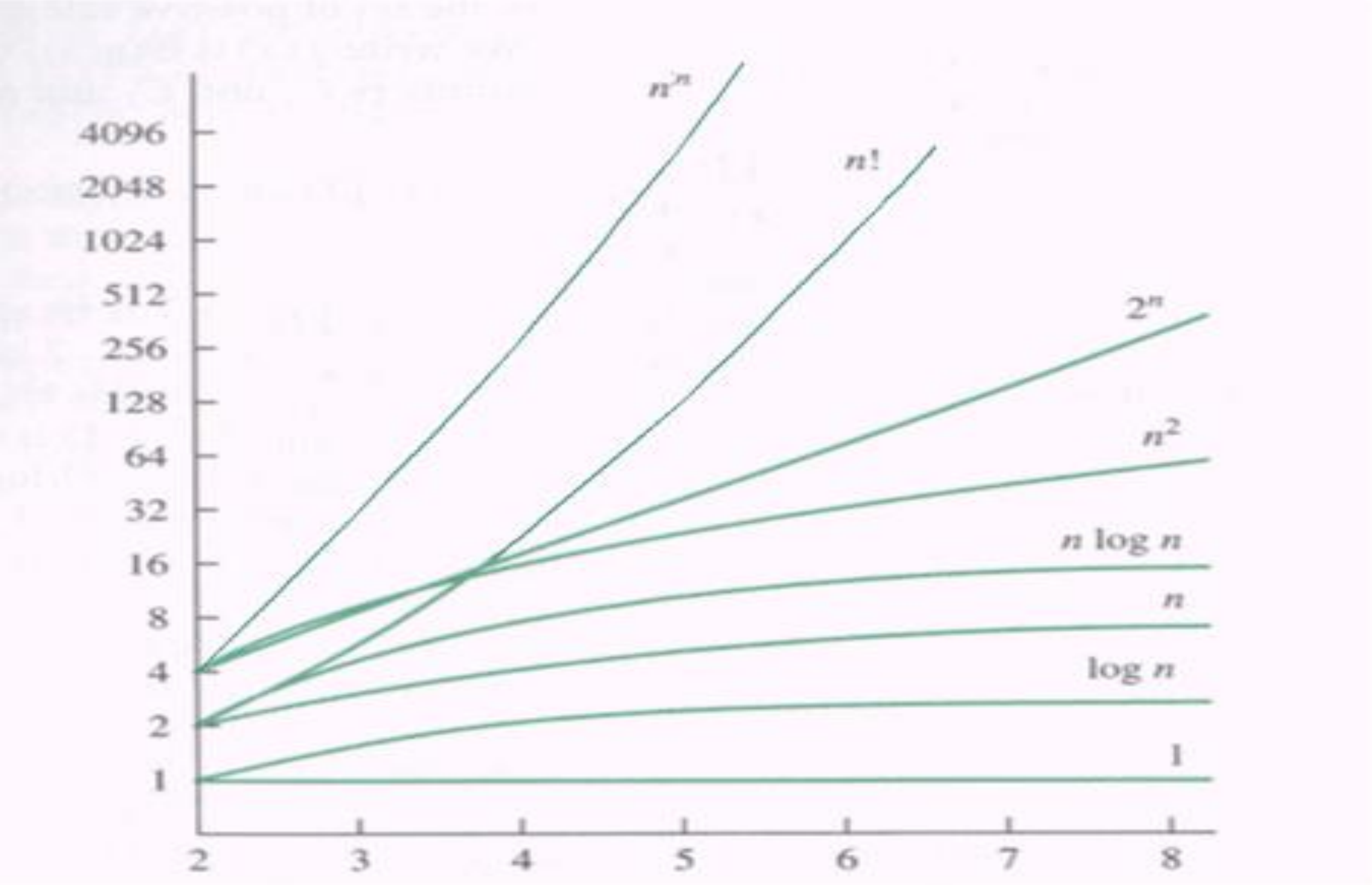
- Find the Big O value for following fragment of code.

```
for i = 1 to n
    for j = 1 to i
        Print j
```

$O(n^2)$

SLIIT
Discover Your Future

Graphs of functions



n	$\log n$	n	$n\log n$	n^2	n^3	2^n
4	2	4	8	16	64	16
8	3	8	24	64	512	256
16	4	16	64	256	4,096	65,536
32	5	32	160	1,024	32,768	4,294,967,296
64	6	64	384	4,094	262,144	$1.84 * 10^{19}$
128	7	128	896	16,384	2,097,152	$3.40 * 10^{38}$
256	8	256	2,048	65,536	16,777,216	$1.15 * 10^{77}$
512	9	512	4,608	262,144	134,217,728	$1.34 * 10^{154}$
1024	10	1,024	10,240	1,048,576	1,073,741,824	$1.79 * 10^{308}$

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) $O(2^n)$

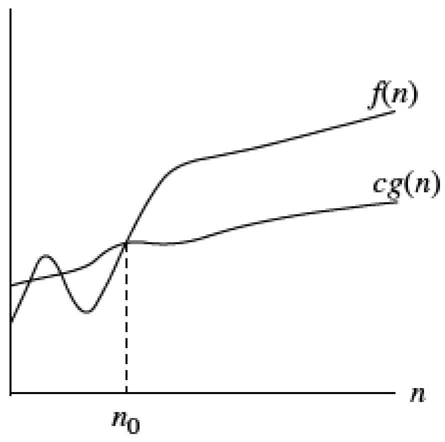
(iii) $O(n)$

Ω - Notation

- Provides the lower bound of the function.

Definition:

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



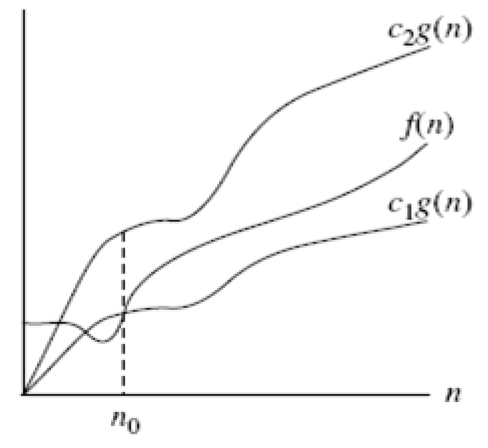
$g(n)$ is an *asymptotic lower bound* for $f(n)$.

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 $n \geq n_0$ }



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

Summary

- What is an algorithm?
- Properties of an algorithm.
- Design methods.
- Pseudocode.
- Analysis(Operation count & Step count, RAM model).
- Insertion Sort.
- Asymptotic Notation

References

- T.H. Cormen, C.E. Leiserson, R.L. Rivest, Clifford Stein Introduction to Algorithms,3rd Edition, MIT Press, 2009.