CS & IT ENGINEERING

Algorithms

Analysis of Algorithms

Lecture No.- 01

By- Dr. Khaleel Khan Sir

ABOUT ME

Hello, I'm Dr. Khaleel Ur Rahman Khan.

1. Ph.D. in Computer Science.
2. Professor in Computer Science.
3. Has more than 28 Years of Experience in Teaching at Engineering Colleges.
4. Published more than 50 journal articles in the areas of Wireless Networks.
5. Seven candidates have been awarded Ph.D. under his Supervision.
6. Has more than 22 years of Educating and Mentoring the GATE Aspirants.

By- Dr. Khaleel

Slide

Topics to be Covered

Topic Introduction to the Course

Topic : Lecture Schedule

DAA :
(Design & Analysis of Algorithms)
* (4-6m)

1. **Analysis of Algorithms** (DA)
   - 1.1 Algorithm Concept and Lifecycle
   - 1.2 Analysis of Algorithms ; (What ; Why ; How)
   - 1.3 Methodology & Types of Analysis (How)
     - Apriori Analysis
     - Aposteriori "
   - 1.4 (*) Asymptotic Notations (ASN) : (3m)
   - 1.5 Framework for Analysing Recursive Algorithms (Industry + Placements)
   - 1.6 Apriori analysis of Non-Recursive Algorithms
   - 1.7 Analysing Loops :
   - 1.8 Space Complexity
   - 1.9 Mathematical Background [Log's + Series + . . .]

Time

(Maths)

Topic : Lecture Schedule

Design Strategies : II

2. Divide & Conquer

DA)

- 2.1 General Method | Control Abstraction
- 2.2 Max-Min Problem
- 2.3 Merge Sort
- 2.4 Binary Search
- 2.5 Quick Sort
- 2.6 Matrix Multiplication
- 2.7 Long Integer Multiplication (LIM)
- 2.8 Master Method for D and C Recurrences
- 2.9 Recursion Tree

Appl's

V (5)

CSIT

DA

(i) DC
(ii) GM
(iii) DP
(iv) BACKT
V BB

Topic : Lecture Schedule

3. **Greedy Method** [GM]

3.1 General Method

3.2 Knapsack Problem ✓

3.3 Job Sequencing with Deadlines (JSD)

3.4 Optimal Merge Patterns

* 3.4.1 Huffman Coding

3.5 Minimum Cost Spanning Trees (MCST) | Disc. Maths

✓ 3.5.1 Prims Method

✓ 3.5.2 Kruskal's Method

3.5.3 Dijkstra's Sp. Tree Algo.

3.6 Dijkstras Shortest Paths Problem

Topic : Lecture Schedule

4. Dynamic Programming (DP)

4.1 The Method
4.2 Difference between DP, Greedy Method and DandC
4.3 Multistage Graphs
4.4 Travelling Salesperson Problem
4.5 Binary Knapsack Problem
4.6 All Pairs Shortest Paths
4.7 Bellman-Ford Single Source Shortest Paths
* 4.8 Longest Common Subsequence
* 4.9 Matrix Chain Multiplication
* 4.10 Sum of Subsets
4.11 Reliable System Design
* 4.12 Optimal Cost Binary Search Tree

Topic : Lecture Schedule

→ GATE-DA

5. Graph Algorithms [D.S + Disc. Maths)

5.1 Representation of Graphs (DS)

5.2 Graph Traversals (DS)

DFS

- 5.2.1 Undirected Connected Graphs
- 5.2.2 Undirected Disjoint Graphs: DFT
- 5.2.3 Directed Graphs & Types of Edges
- 5.2.4 DAG

BFS

- 5.2.5 FIFO BFS
- 5.2.6 LIFO BFS
- 5.2.7 LC BFS

5.3 Parenthesization Theorem

5.4 Components

→ Connected Components

→ Strongly Connected Components

→ Bi Connected Components

5.5 Articulation points

Topic : Lecture Schedule

Tool for Algo. Developers (DA)

6. Heap Algorithms

6.1 Operations : Create, Insert, Delete, Modify

6.2 Applications : Heapsort

Topic : Lecture Schedule

7. Sets [D.S]

7.1 Representations

7.2 Operations

Topic : Lecture Schedule

8. **Soring Algorithms** (GATE - DA + CSIT)

   8.1 Basic terminology

   8.2 Methods

   8.2.1 Bubble Sort

   8.2.2 Selection Sort

   8.2.3 Insertion Sort

   8.2.4 Radix Sort

Topic : Lecture Schedule

9. Backtracking & Branch and Bound

Text – Books :

1. Introduction to Algorithms – Cormen (LCS Book)

   CLRS

2. Fundamentals of Algorithms – Horowitz and Sahni

outcomes :

(i) Subject Knowledge [Foundation]

(ii) Competetive Exams : (GATE + TIFR + ISRO + BARC + UGC-NET + State-level Exams + DRDO)

(iii) placements

written test

Interviews

+ Coding (after GATE)

[(DS + Algo) coding]

→ 50-55 Hrs

→ 8-10 Marks [Min] $\left(\frac{10-12}{12-14}\right)$

→ Pre Requisites :

*

1) Familiarity with Prog. language → Constructs

if-then-else (C|C++|Java)

for loop for(i=1; ~~i~~<=n; ++i)

for i ← 1 to n

2) Maths Background

(Functions + Log's + Exponents + Series)

3) Basic Data Structures (Stacks, Q's, L.L, Trees, Graph)

recursion

4) Discrete Maths :

(recurr. relations)

What is an Algorithm : (soln)

Defn : Algorithm Consists of Finite Set of [Steps/Stmnts] to Solve a given Problem,

(Function)

[one/more basic opn] <CS + Non-CS>

(Computer based Algo.)

→ Mohd. Musa [Al Khwarizmi] (Persian)

↓

(Algorithm)

Ex:
1. $\boxed{a = b + c;}$ Addition; =
2. for $i \leftarrow 1$ to n
   $d = e * m;$
3. Push(s, a);

⇒ Algorithm

↓

Steps | Stmnts

↓

one | more fund. op'ns

$a + b *$

Properties

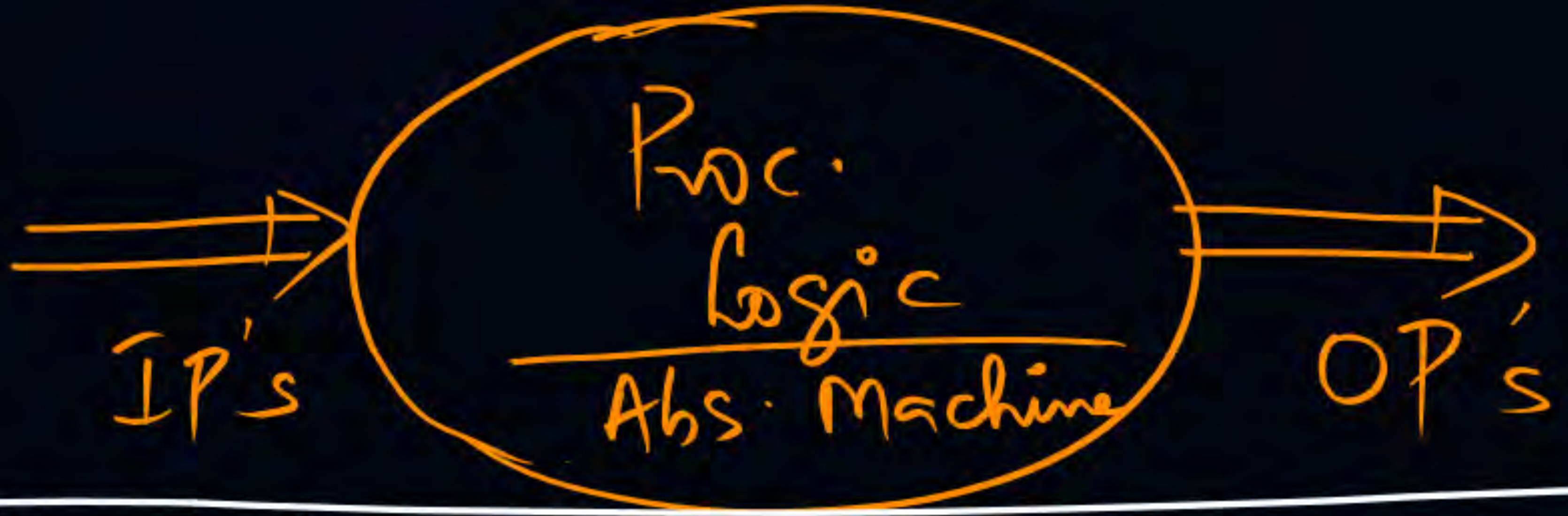
definiteness [Clear]

Effective [Finite Time]

→ May accept 0/more Inputs

→ Must Produce atleast one output

(ii)



Lifecycle Steps:

⟨DAA⟩

(i) Problem Definition

(ii) Requirements | Conditions [S.R.S] : Constraints

(iii) Design | Logic

(iv) Express Algo (pseudo)

(v) Validation ⟨Correctness⟩

(vi) Analysis

(vii) Implementation

(viii) Testing & Debugging

1) Need for Analysis (Why & What):

- To make Performance Comparison (Analysis)
- To determine the Resource Consumption (Metric)

P

A_1 A_2 A_3

efficient

Time & Space

- Time → CPU
- Space → Memory

1) Time Complexity

2) Space Complexity

1) (P.C)

2) (Mobile Computing) → (Power|Energy)

3) (Distributed Computing) → (Bandwidth|Channel Capacity)

* Methodology of Analysis (How to Analyse the Algo)

Time + Space

1. ($x \leftarrow y + z$)

(What's the Time taken to execute this Statement)

a) (Aposterior Analysis)

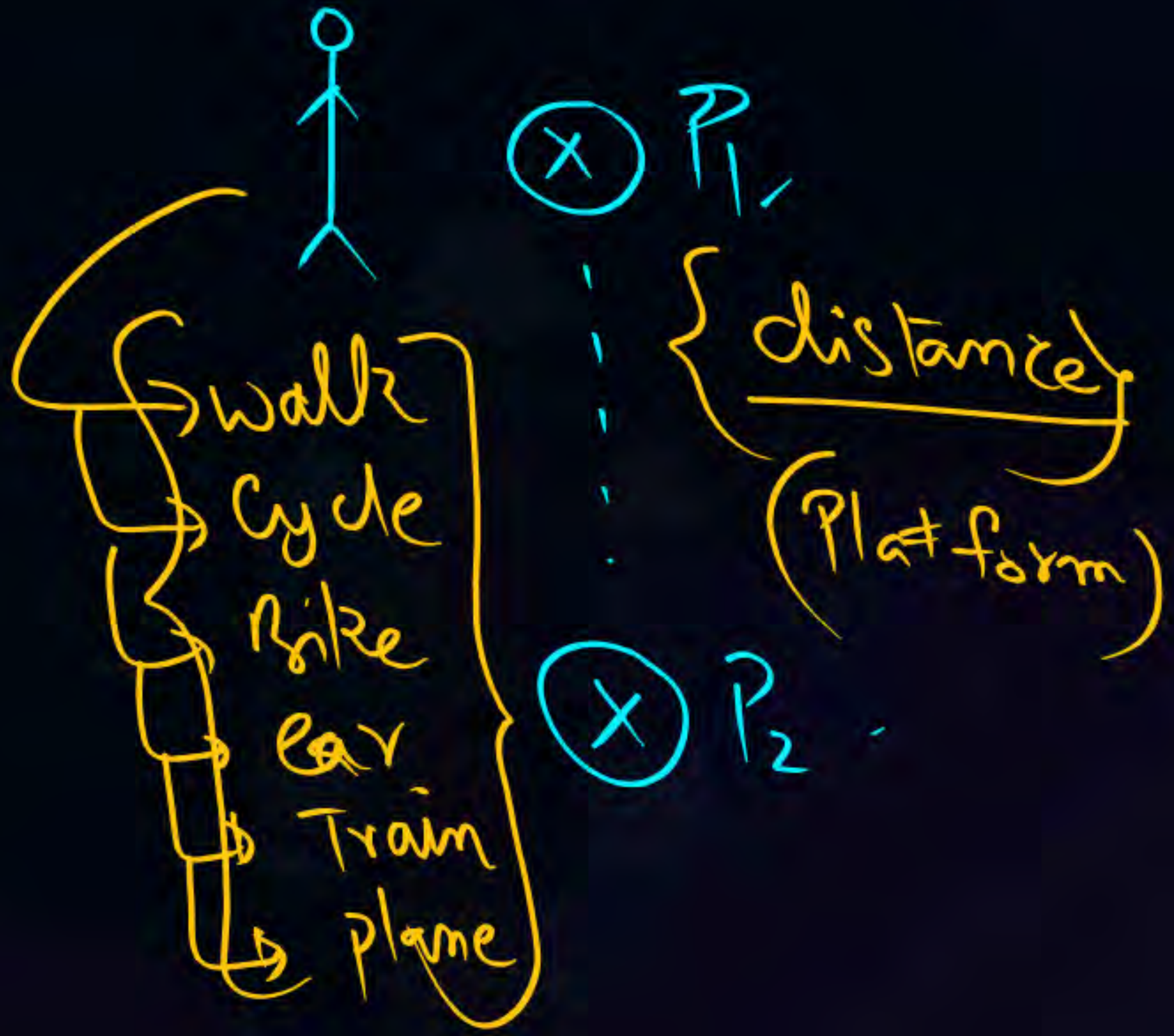
Platform

H/w

S/w

< CPU + Mem + I/O .. >

< O.S + Compiler >



1) Aposteriori Analysis [Platform Dependent Analysis]

[Experimentation Analysis]

Advantages :

→ Exact values of Time & Space in real units

Drawbacks

→ Difficult to carry out

→ It is necessary to implement & execute the Algorithm

→ Experiments can be carried out on a limited set of inputs, ∴ care must be taken to ensure that the inputs are representative;

→ It is difficult to compare efficiency of two Algorithms, unless experiments are carried on same platform, (*)

$\langle$ Non-Uniformity $\rangle$

Desirable Req's:

Platform Independent

Apriori Analysis

- → Takes into account all possible inputs (classes)
- → Allows us to evaluate & Compare the efficiency of Two Algorithms, independent of platform (H/w & S/w)
- → Should be able to carry out by Studying high-level description of Algorithm without implementation.

Apriori Analysis (How)

< Components of Apriori Analytic Framework >

1. A language (Pseudo-language) for describing Algorithm Statements (Paper)
2. A Computation Model (RAM model) that algorithm executes within it;
3. A metric for measuring Algo running Time
   (Fund. opn)
4. An approach (formal Notation) for characterizing running Time of Algo; [ASN]

```
Algorithm Test (a, b, c, n)
{
    integer a, b, c, i;

    1. c ← a + b;

    2. for i ← 1 to n
           b ← a * c;

    3. for i ← 1 to n
           for j ← 1 to n
               b = c + 5;
}
```

$2 = 1 + 1$

$(1 + (n+1) + 2n) + n + n$

$(1 + (n+1) + n)$

$n + n(n+1) + n*n + n*n + n*n$

RAM Model of Computer:

Memory (Inf.)

$S_1 S_2 S_3 - - - - S_{1000}$

CPU

Basic opn's (1-unit of Time)

(Step-Count Method)

Time taken = Time of all the Steps

$$= \sum_{i=1}^{K} \text{Step}_i$$

$$\text{Time-Test} = \underbrace{2}_{S_1} + \underbrace{(4n+2)}_{S_2} + \underbrace{(4n^2+4n+2)}_{S_3}$$

: 'n' : input Size

Quadratic

$$= (4n^2 + 8n + 6) \text{ : Total time}$$

→ Math Function w.r.to input Size 'n'

Fn's

Polynomial

(n^x) $x \geq 0$

Exponential

a^n, $(a > 1)$

Alternate method of determining Running Time in Apriori Analysis

⟹ determination of order of Magnitude of a Stmnt / Construct / DS operation

→ Refers to the Frequency/count of the (No. of times) Fund. op'n in the Statement

Algorithm Test (a, b, c, n)
{
integer a, b, c, i, n, j;

1. $c \leftarrow a+b;$: 1

2. for $i \leftarrow 1$ to n : n
   $b \leftarrow a*c;$

3. for $i \leftarrow 1$ to n : n^2
   for $j \leftarrow 1$ to n
   $b \leftarrow c+5;$

}

Time : $1+n+n^2$

: $\boxed{(n^2+n+1)}$

$\sim$: $4n^2+8n+6$

- fn

ASN : $O(n^2)$

charact.

Algorithm Test1
{
integer a, b, c;

1. read (a, b, c);   1

2. if (a < b)
    if (a < c) then   : 3
        print (a)
    else if (c < b)   : 3
        print (c);
    else print (b);
}

Time : Constant

: O(1)

Algo Test 3 (n)   Time : $c + n^2 + O(f)$
{
integer a, b, c, n;   if f(): log n

1. a = b + c;   : 1   : $c + n^2 + \log n$

2. print (a);   : 1   : $O(n^2)$

3. for i ← 1 to n * n   : n^2
    c = a + b;

4. f(c);   : O(f)
}

Note: The objective of Apriori Analysis is to represent the Time Complexity of an Algorithm by a Mathematical function w.r.to input-Size say 'n';

Polynomial

(n^x)

$x \geq 0$

(Lesser rate of growth)

(Fast)

Exponential

(a^n)

$a > 1$

Faster rate of growth

<Slow>

1 - time

n+1 times

(n+1) times

n - times

for(i=1; i<=3; ++i)

i = i+1

i	
1 ≤ 3 ✓	
2 ≤ 3 ✓	
3 ≤ 3 ✓	
4 ≤ 3 ✗	

4 - times

3

n

```
for(i=1;i<=100;++i);
printf("%d", i);
```

101

Topic: Analysis of Algorithms

Analyzing algorithms involves thinking about how their resource requirements-the amount of time and space they use-will scale with increasing input size.

Proposed Definition of Efficiency (1): An algorithm is efficient if, when implemented, it runs quickly on real input instances.

Proposed Definition of Efficiency (2): An algorithm is efficient if it achieves qualitatively better worst-case performance, at an analytical level, than brute-force search.

Proposed Definition of Efficiency (3): An algorithm is efficient if it has a polynomial running time

THANK - YOU