DSA - Lecture 1 Note

Data Structures and Algorithms - IT1170

Introduction to Algorithms

1. What is an Algorithm?

An **algorithm** is a **step-by-step** procedure to **solve a specific problem**. It is a **set of instructions** that take an **input**, process it, and produce an **output**.

Example of an Algorithm (Sorting Numbers):

Input: 3, 1, 7, 2, 9, 8, 5, 4, 6

Output: 1, 2, 3, 4, 5, 6, 7, 8, 9

Steps to Sort (Selection Sort Example):

1. Find the smallest number in the list.
2. Swap it with the first element.
3. Move to the next position and repeat until the list is sorted.

Characteristics of an Algorithm:

- **Definiteness:** Each step must be precisely defined.
- **Finiteness:** Must complete in a finite number of steps.
- **Effectiveness:** Each step should be simple enough to execute.
- **Input & Output:** Must take at least one input and produce at least one output.

---

2. Properties of an Algorithm

A well-defined algorithm should have the following properties:

- **Correctness:** Produces the right output for every valid input.

- **Unambiguity:** Every step must be **clear and well-defined**.
- **Generality:** Must work for **all possible cases**.
- **Simplicity:** Easy to understand and implement.
- **Efficiency:** Should use the least amount of time and resources.
- **Termination:** Must stop after a finite number of steps.

---

3. Applications of Algorithms

Algorithms are used in **many areas of computing** such as:

- **Data Retrieval** – Searching and fetching information from databases.
- **Network Routing** – Finding the fastest path in communication networks.
- **Sorting & Searching** – Used in databases and e-commerce.
- **Artificial Intelligence (AI)** – Machine learning and decision-making.
- **Graph Algorithms** – Used in GPS navigation and shortest path calculations.

---

4. Pseudocode

Pseudocode is a **simplified way of writing an algorithm** in a format that resembles a programming language but does not follow strict syntax.

Rules of Writing Pseudocode:

- Uses **plain English** for easy understanding.
- Proper **indentation** for readability.
- Uses **loops and conditions** explicitly.
- `//` is used for **comments**.
- `=` is used for **assigning values**.

Example: Find the Maximum of Two Numbers

```
BEGIN
   INPUT a, b
   IF a > b THEN
      PRINT a
   ELSE
      PRINT b
   ENDIF
END
```

---

5. Algorithm Analysis

Algorithm analysis helps determine the **efficiency** of an algorithm in terms of:

- **Memory Usage:** How much space is required?
- **Number of Steps:** How many operations are performed?
- **Execution Time:** How long does it take to run?

Why is Analysis Important?

- Helps in comparing different algorithms.
- Predicts runtime for larger inputs.
- Optimizes performance for better efficiency.

Types of Cases in Algorithm Analysis:

1. **Best Case:** Minimum steps required (fastest execution time).
2. **Worst Case:** Maximum steps required (slowest execution time).
3. **Average Case:** Expected number of steps for random input.

---

6. Methods of Algorithm Analysis

1. Operation Count Method

- Counts selected operations (e.g., **additions, multiplications, comparisons**).

- Helps understand which operations are expensive.
- **Example:** In sorting, the number of **comparisons** and **swaps** are counted.

2. Step Count Method (RAM Model)

- Assumes a **single processor**.
- Each basic operation (`+`, , `=`, etc.) takes **one step**.
- Each **memory access** takes **one step**.
- **Formula:** Running Time = Sum of Steps.

Example of RAM Model Analysis:

```
n = 100   // 1 step
n = n + 100   // 2 steps
PRINT n   // 1 step
```

Total Steps: `1 + 2 + 1 = 4`

Example: Printing Numbers from 1 to 10

```
i = 1 → 1 step
WHILE i <= 10 → 11 steps
   PRINT i → 10 steps
   i = i + 1 → 20 steps
```

Total Steps = 42

Example: Printing Even Numbers from 10 to 20

```
FOR i = 10 TO 20 STEP 2 → 6 steps
   PRINT i → 6 steps
```

Total Steps = 12

---

7. Problems with RAM Model

- Step count **varies** between different hardware architectures.
- Complex algorithms (e.g., **sorting algorithms**) require more advanced analysis.
- **Some operations** take different times in different machines (e.g., multiplication may take longer than addition).

---

8. Complexity of Algorithms

Algorithm complexity is measured using **Big O Notation**.

Common Complexities:

Notation	Complexity Type	Example
$O(1)$	Constant Time	Accessing an array index
$O(\log n)$	Logarithmic Time	Binary search
$O(n)$	Linear Time	Scanning an array
$O(n \log n)$	Log-Linear Time	Merge Sort
$O(n^2)$	Quadratic Time	Bubble Sort
$O(2^n)$	Exponential Time	Recursive Fibonacci

---

9. Summary

- **Algorithm:** Step-by-step instructions to solve a problem.
- **Properties:** Must be correct, simple, and efficient.
- **Applications:** Used in sorting, searching, AI, networks, and databases.
- **Pseudocode:** Writing an algorithm in structured steps before coding.
- **Analysis:** Measures efficiency based on time and memory usage.
- **Complexity:** Helps understand how an algorithm performs as input size increases.
