Designing and Analysis of Algorithms

Course Code: ECS 5101/CS514

- DR RAHUL MISHRA
- IIT PATNA

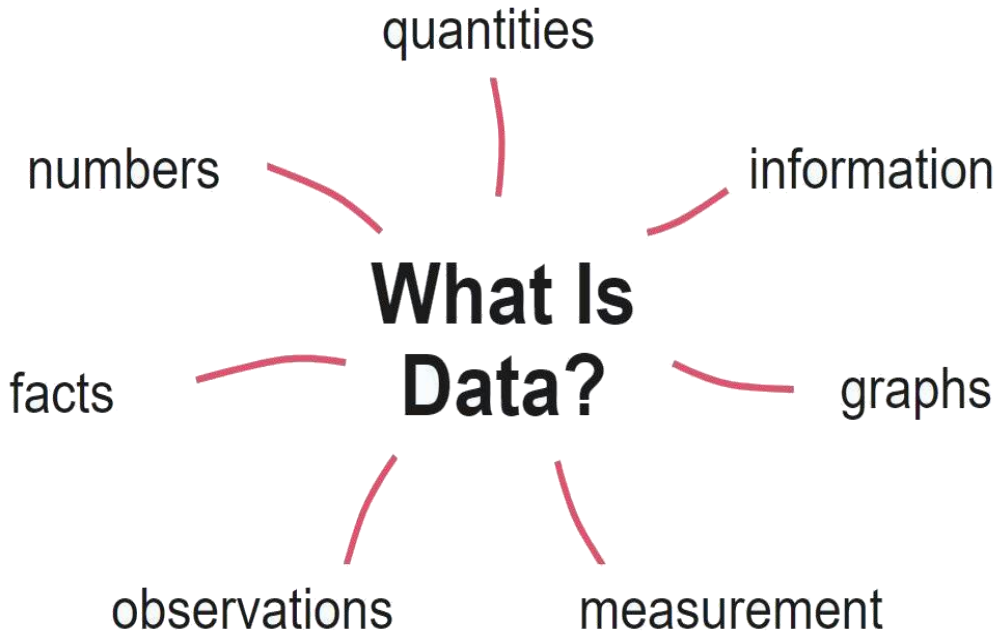
- **Lecture 1**

Data

- Data refers to any set of information that can be stored, analysed, and used to inform decision-making.
- Data can take many forms, such as **numbers**, **words**, **images**, or **any other type of input that can be recorded digitally or manually.**
- Data can come from various sources, including **sensors, surveys, databases, social media, and more**.

Data

- In order to be useful, data must be organized, analysed, and interpreted.



- Data analysis involves using statistical and computational tools to identify patterns, trends, and relationships within the data.
- This analysis can be used to draw conclusions, make predictions, and inform decision-making in a wide variety of fields, including business, healthcare, science, and more.

Data Structure

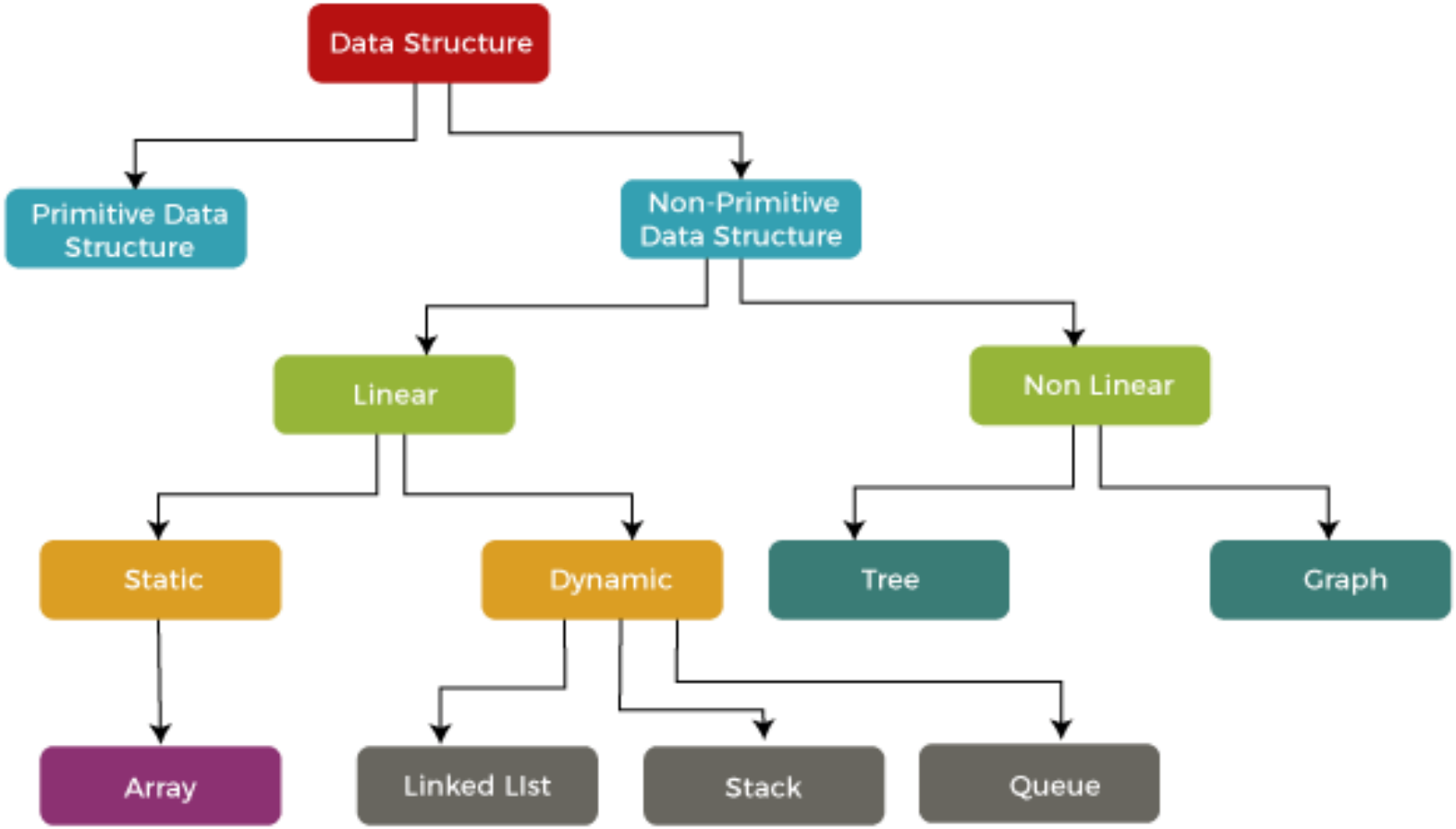
- Data may be organized in many different ways; the logical or mathematical model of a particular organization of data is called a ***data structure.***

- The choice of particular data structure depends on two considerations:

  1). It must be rich enough in structure to mirror the actual relationships of the data in the real world.

  2). The structure should be simple enough that one can effectively process the data when necessary.

Data Structure



Data Structure

- **Arrays**
  - ❖ An array is a data structure that stores a collection of elements, all of the same type, in contiguous memory locations.
  - ❖ The elements in an array are accessed using an index, which is an integer value that represents the position of the element in the array.
  - ❖ Arrays are commonly used for a variety of tasks, including storing and manipulating collections of data such as lists, tables, and matrices.

Arrays

- It can also be used to represent data structures such as stacks, queues, and trees.
- In most programming languages, arrays have a fixed size, meaning that once an array is created, its size cannot be changed.
- Some languages also support dynamic arrays, which can grow or shrink in size as needed.
- Arrays are a fundamental data structure in computer programming and are used extensively in a wide range of applications, including scientific computing, data analysis, and web development.

Linked Lists

- A linked list is a linear data structure in computer science that is used to store a sequence of elements or nodes.

- Linked lists are not stored in contiguous memory locations. Instead, each element or node in a linked list contains a reference to the next node in the sequence.

- The first element in the sequence is called the head of the linked list, and the last element is called the tail. Each node in a linked list contains two fields, one for storing the data and the other for storing a reference to the next node. The last node in the list has a null reference as its "next" field to signify the end of the list.

- There are several types of linked lists, including singly linked lists (where each node has a reference to only the next node), doubly linked lists (where each node has a reference to both the next and previous nodes), and circular linked lists (where the last node in the list has a reference to the first node, creating a loop).

Data Structure

- **Tree and Graph**

❖ A tree is a hierarchical data structure consisting of nodes connected by edges. It is a special type of graph where each node has exactly one parent, except for the root node which has no parent. A tree is a recursive data structure because each node can have its own subtree. Trees are commonly used to represent hierarchical structures, such as file systems, organizational charts, and family trees.

❖ A graph is a collection of nodes, also called vertices, that are connected by edges. Unlike a tree, a graph can have multiple edges between the same pair of nodes and can also have cycles, which are loops that start and end at the same node. Graphs are used to represent complex relationships between objects or entities, such as social networks, road maps, and computer networks.

Stack and Queue

A **stack** is a data structure that stores elements in a last-in-first-out (LIFO) order. This means that the last element added to the stack is the first one to be removed. Elements can only be added or removed from the top of the stack. This makes a stack useful for tasks such as keeping track of function calls in a program, undoing operations in a text editor, or processing expressions in a calculator.

A **queue**, on the other hand, is a data structure that stores elements in a first-in-first-out (FIFO) order. This means that the first element added to the queue is the first one to be removed. Elements can only be added at the back of the queue and removed from the front. This makes a queue useful for tasks such as processing requests in a web server, printing documents in a printer, or handling messages in a message queue.

Data Structure Operations

- ***Traversing:*** Accessing each record exactly once.
- ***Searching:*** Particular record finding
- ***Inserting:*** Adding a new record
- ***Deleting:*** Removing record

Linear and Non-Linear Data Structures

Criteria	Linear Data structure	Non-Linear Data structure
Basic	In this structure, the elements are arranged sequentially or linearly and attached to one another.	In this structure, the elements are arranged hierarchically or non-linear manner.
Types	Arrays, linked list, stack, queue are the types of a linear data structure.	Trees and graphs are the types of a non-linear data structure.
Implementation	Due to the linear organization, they are easy to implement.	Due to the non-linear organization, they are difficult to implement.
Traversal	As linear data structure is a single level, so it requires a single run to traverse each data item.	The data items in a non-linear data structure cannot be accessed in a single run. It requires multiple runs to be traversed.
Arrangement	Each data item is attached to the previous and next items.	Each item is attached to many other items.

Linear and Non-Linear Data Structures

Criteria	Linear Data structure	Non-Linear Data structure
Levels	This data structure does not contain any hierarchy, and all the data elements are organized in a single level.	In this, the data elements are arranged in multiple levels.
Memory utilization	In this, the memory utilization is not efficient.	In this, memory is utilized in a very efficient manner.
Time complexity	The time complexity of linear data structure increases with the increase in the input size.	The time complexity of non-linear data structure often remains same with the increase in the input size.
Applications	Linear data structures are mainly used for developing the software.	Non-linear data structures are used in **image processing** and **Artificial Intelligence**.

About the algorithm :-

A road trip from Patna to Delhi.

↳ which path to follow ?

Factors →

1. Numerous Points (way point)
2. Resource constraints
3. Subjectivity → Human bias.

Need for a methodology, a systematic approach that eliminates manual calculation.

Informal Definition of Algorithm

- "An algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output in a finite amount of time"

⇒ An algorithm is thus sequence of computational steps that transform the input into the output.

⇒ A tool for solving a well-specified computational problem.

⇒ The algorithm describes a specific computational procedure for achieving that input/output relationship for all problem instances.

⇒ Consider a sorting problem :-

Input : A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$

Output : A permutation (reordering) $\langle a_1', a_2', \ldots, a_n' \rangle$ of the input sequence such that $a_1' \leq a_2' \leq a_3' \leq \ldots \leq a_n'$

For example :-

$\langle 31, 41, 59, 26, 41, 58 \rangle$ Input sequence are also called instance

↳ output :- $\langle 26, 31, 41, 41, 58, 59 \rangle$

(Because sequence could be an instance of problem.)

* "Sorting is a fundamental operation in computer science"

* Which algorithm is best for a given application depends on →

i) the number of items to be sorted

ii) the extent to which the items are already somewhat sorted

iii) possible restrictions on the item values

iv) the architecture of the computer

v) Kind of storage devices to be used → Memory organization.

Correctness of an Algorithm →

⇒ An algorithm for a computational problem is correct if, for every problem instance provided as input, it halts – finishes its computing in finite time – and outputs the correct solution to the problem instance.

* An algorithm can be specified in English, as a computer program, or even as a hardware design.

* The only requirement is that the specification must provide a precise description of the computational procedure to be followed.

? What kind of problems are solved by algorithms →

(a) Determining the sequences of the 3 billion chemical base pairs that make up human DNA. (String Matching)

(b) The internet enables people all around the world to quickly access and retrieve large amount of information (Network flow Analysis)

* E-commerce enables goods and services to be negotiated and exchanged electronically.
* Inventory Management in large-scale Industry.
* To determine shortest route from one intersection to another
* [We are given a mechanical design in terms of a library of parts, where each part may include instance of other parts, and we need to list the parts in order to make each part appears before any part that uses it.]

(Topological sorting)

* Any Many more ---

Critical Considerations in algorithm design and selection :-

① Time complexity → ("delay game")] Asymptotic Notation

② Space complexity →

③ Computational complexity → (① + ②) Strike a balance between time & space complexity.

④ Approximation → Make a balance between precision and computation efficiency.

⑤ Failure Probability → In road trip from Patna to New Delhi, the failure probability could be unforeseen circumstances such as sudden road closure / unexpected traffic incidents

⑥ Robustness and error handling →

⑦ Scalability

⑧ Parallelization and Concurrency

Algorithm designing techniques :-

a) **Brute-force methodology** :- Simplest designing technique — systematically exploring all possible solution until the correct one.
(String matching)

b) **Backtracking** :- is a systematic strategy, exploring all possible solutions in a structured manner.
(N-Queen problem)

c) **Greedy algorithm** :- operate by making locally optimal choices at each step.

d) **Dynamic programming** :- divide a complex problem into simpler overlapping subproblems i.e. that each subproblem is solved only once.

(e) Divide-and-Conquer :-

↳ breaking down a complex problem into smaller, more manageable subproblems.
↳ Each subproblem is solved independently and solutions are combined.

(f) NP-completeness :- are deemed challenging, with solutions that may necessitate exponential time for discovery.

(g) Randomized algorithms: Leverages randomness as a fundamental concept
↳ Incorporate a degree of unpredictability