Course Title – Data Structure Using C

Course Code - 313301

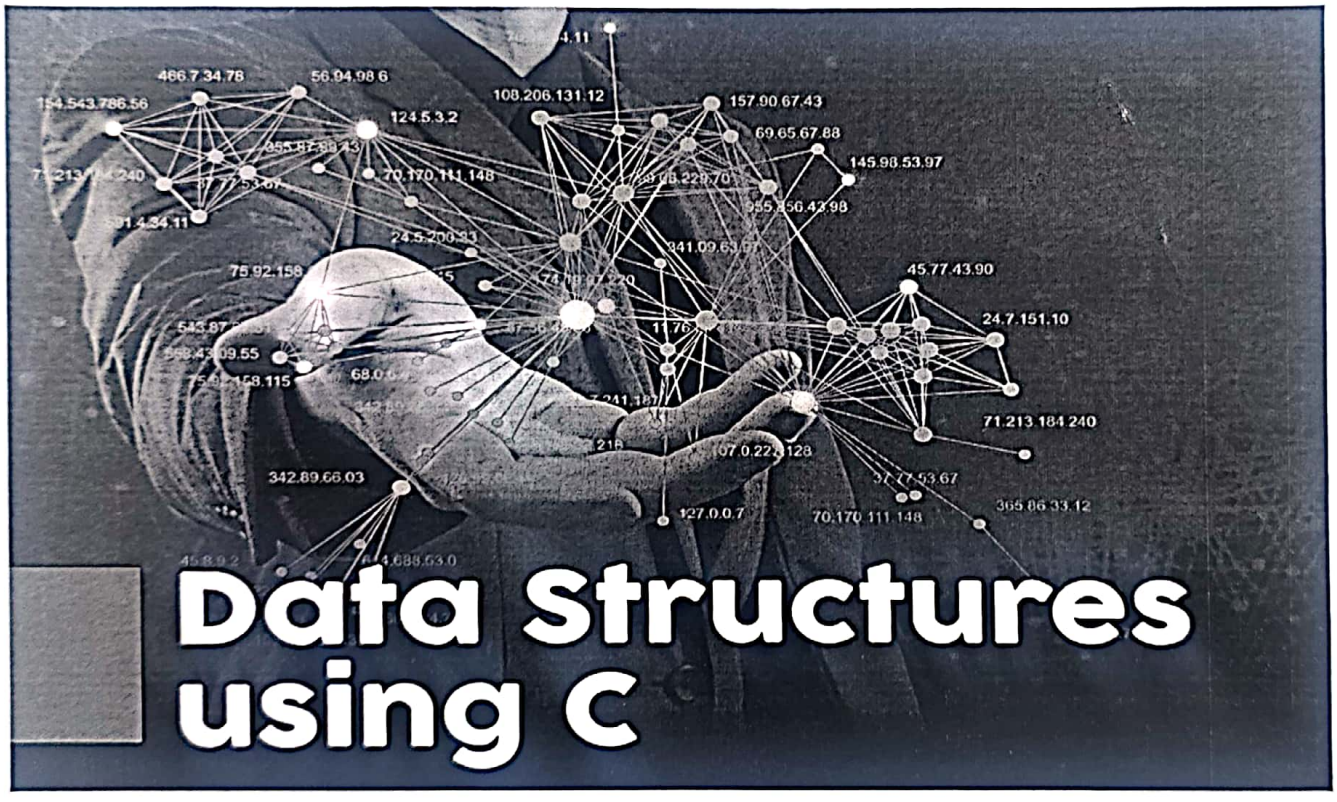
Programme Name - Computer Engineering

Semester - Third

Unit - I Introduction to Data Structures

Total Marks : 04

Prepared By: Mr. Milind R. Birhade



Unit - I 04-Marks

Introduction to Data Structures

1.1 Introduction:

Concept and need of Data Structure
Defination, Abstract Data type

1.2 Types of Data Structures:

(i) Linear Data Structures
(ii) Non-Linear Data Structures

1.3 Operations on Data Structures:

(i) Traversing
(ii) Insertion
(iii) Deletion

Data Structure Using C

313301

Unit-I Introduction to Data Structure

(1.1) Introduction:

In Computer Science, large amount of data are processed every day. To store and manage this data efficiently, proper techniques are required. Data Structure is one such technique that helps in organizing data in systematic way.

Data structures improve the performance of programs by making operation like Searching, insertion, deletion, and sorting faster and Easier.

Data: Data is a collection of raw facts and figures.

Ex. Students Marks, Roll Numbers, Employee No

Information: Processed and meaningful information is called information

ex. Result sheet of students.

***Concept of Data Structures**

Organizing data logically

Reducing processing time

using memory ~~efficiently~~ efficiently

Making operations easier

* **Data Structure:** In simple terms, a Data Structure is a systematic way of organizing, managing, and storing data in a computer so that it can used efficiently.

Data Structure Concept is fundamentally concerned with 5 core things

1] Organization of data.
2] Associativity (relationships) among data elements.
3] Accessing methods (how we get to the data)
4] Operation on data (like inserting, deleting or sorting).
5] Processing alternatives for data.

Why do we need to choose the right data Structure?

* **To Capture relationships:** It helps express how data elements connect to one another.
* **Efficiency:** It ensure that processing and accessing the data is as fast as possible with minimal memory usage.

*The formal Defination (The Triplet concept) 2-Marks

A data Structure as an instances of an Abstract Data Type (ADT).

Formally, a data structure is defined as a triplet

$$\text{Data Structure} = (D, F, A)$$

Where:

D (Domains): Represents the set of data types/domains being used.

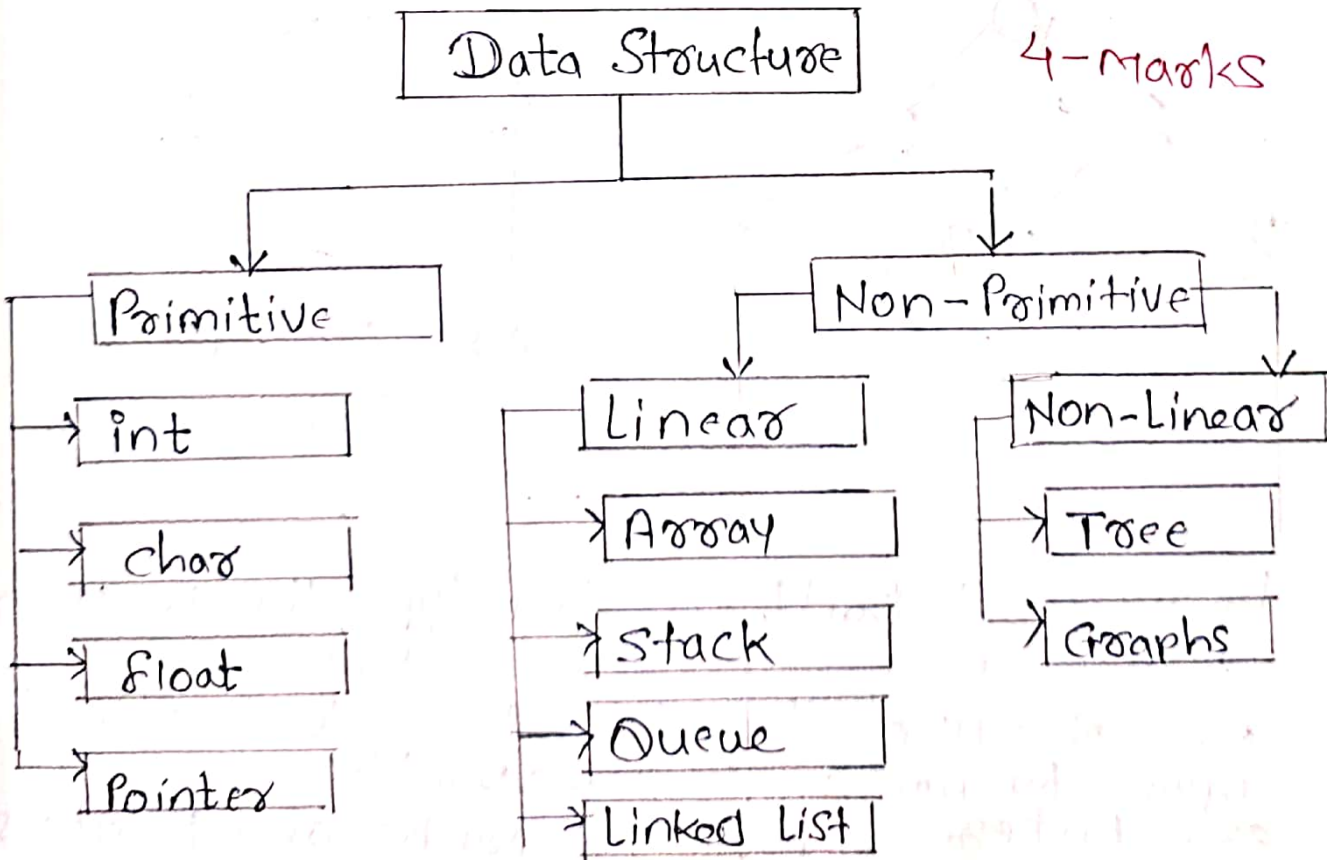
F (Functions/operations) Represents the set of operation you can perform on that data (eg. insert(), delete()).

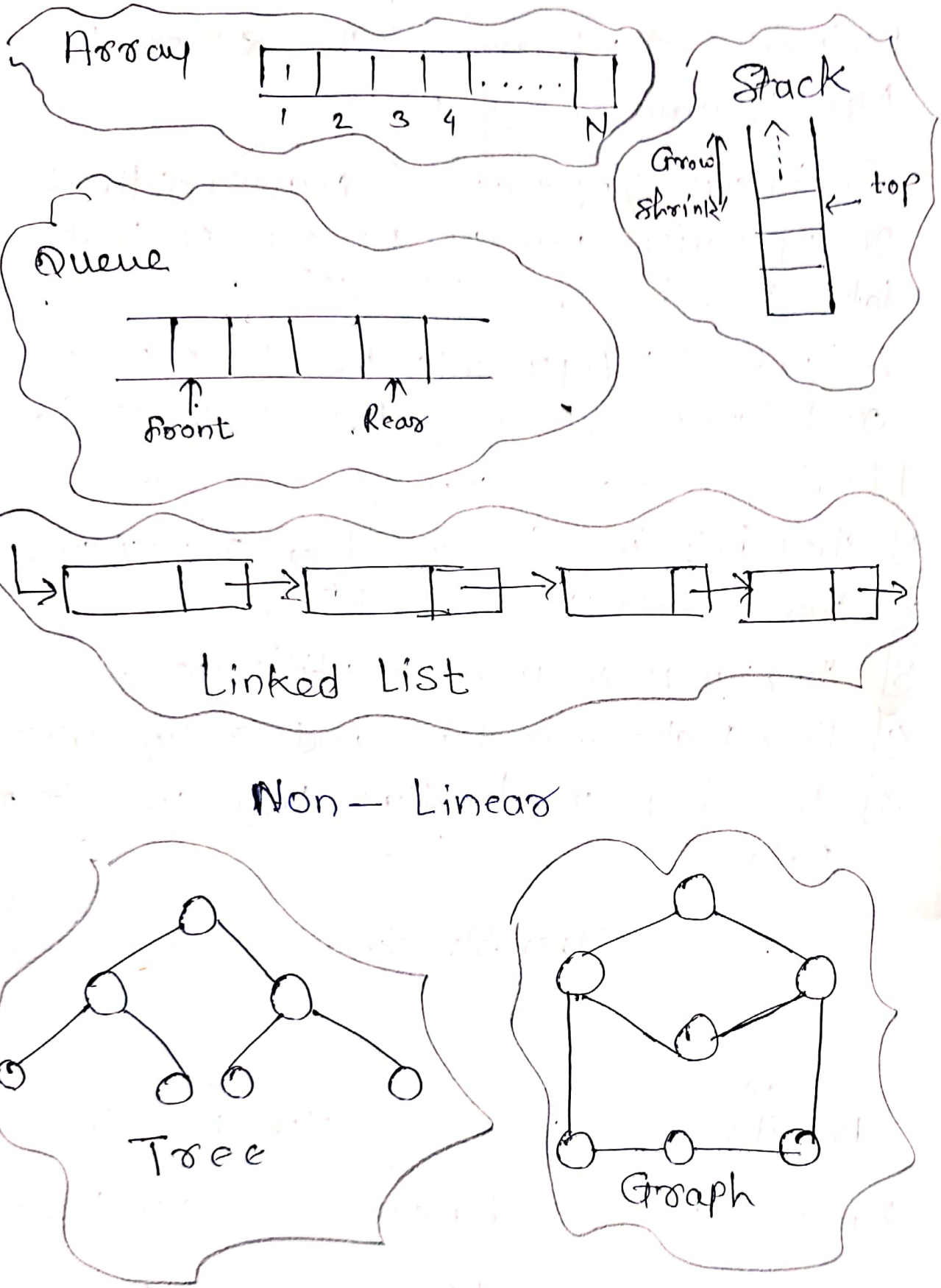
A (Axioms): Represents the set of rules or behaviors governing those operations.

Need of Data Structure: 2-Marks

1] They help in storing data efficiency.
2] They reduce execution time.
3] They improve memory utilization.
4] They make searching and sorting easier.
5] They help in handling large amounts of Data.

Data Structure 4-Marks





Primitive: Built-in data types that are natuly available in most programming languages. Data objects of those types can be directly operated upon by machine level instructions.

ex. Integers, Reals., character and Pointers.

Non- Primitive : Data structures that are derived from primitive data structures.
They store a collection of data elements together, which can be:

Homogeneous: Elements of the same data type

Heterogeneous: Elements of different data type.

ex. Array, structure, union, linked list, stack, queue, tree & graph.

Linear: In Linear data structure, elements are arranged sequentially.

ex. Array, Linked list, stack, queue

Easy to traversal

Elements are stored one after another.

Non-Linear: In non Linear data structure, elements are arranged hierarchically.

ex. Tree & Graph.

Complex structure & multiple relationships between elements

2 Marks

	Linear	Non-Linear
1	Element are arranged Sequentially	Elements are arranged hierarchically
2	Easy traversal.	Complex traversal.
3	Single Relationship	Multiple Relationship.
4	Ex. Array, stack.	Ex. Tree, Graph.
5	Less memory utilized	More Memory utilized.

Operations on Data structure 2-marks

1) Traversing: Accessing each element one by one
2) Insertion: Adding a new element into the structure
3) Deletion: Removing an element from the structure
4) Searching: Finding the location of an element
5) Sorting: Arranging data in ascending or descending order
6) Merging: Combining two data structures into one

Algorithm

An algorithm is a step by step procedure used to solve problem.

Characteristics of Algorithm

Input: Algorithm should take input
Output: Algorithm should produce output
Definiteness: steps must be clear
Finiteness: Algorithm must stop after finite steps
Effectiveness: steps should be simple and executable

Time complexity: Time complexity represents the amount of time required by an algorithm.
or
The amount of time an operation takes to run as the data size(n), grows.

Space complexity: The amount of extra memory required to perform the operation.

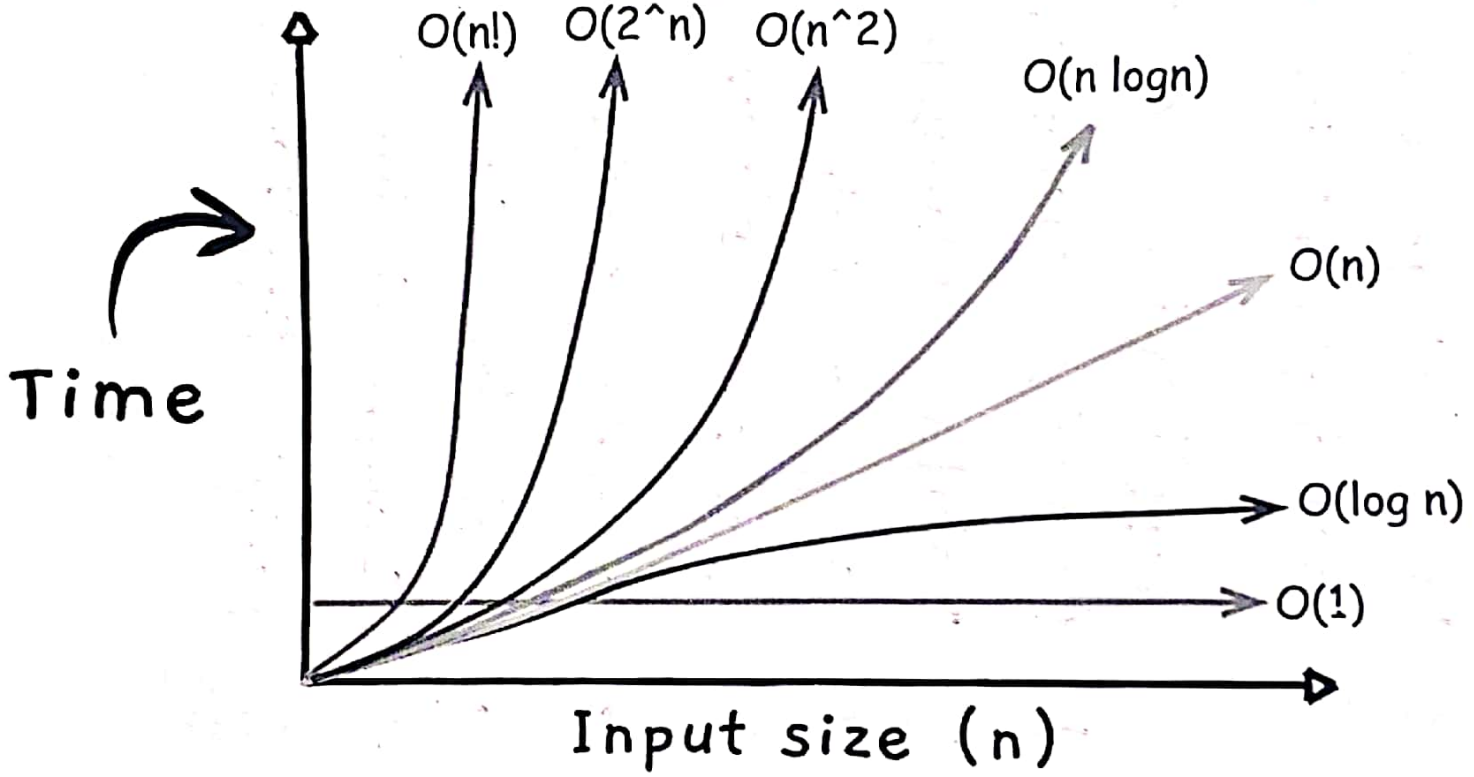
Advantages of Data Structure

1] Efficient data management
2] Faster Processing
3] Better memory usage
4] Improved program performance
5] Easy implementation of algorithms

Applications of Data Structure

1] Operating systems.
2] Database Management Systems.
3] Computer Networks.
4] Artificial Intelligence.
5] Compiler Design.
6] Web browser.
7] Search Engines.
8] Mobile Applications.

BIG-O NOTATION EXPLAINED



Basic Operations 2-Marks

(i) Traversing: Visiting every element in the structure exactly once to process or display it.

(ii) Insertion: Adding a new data element at a specific position or at the end.

(iii) Deletion: Removing an existing element from the structure.

Big oh Notation (Big o notation)

Big o notation is a mathematical notation used in computer science to describe the performance or complexity of an algorithm

Specially, it characterizes the worst-case scenario by describing the execution time or space required in terms of the input size (n)

Common Big o Time complexities:

Notation	Name
$O(1)$	Constant Time
$O(\log n)$	Logarithmic Time
$O(n)$	Linear Time
$O(n \log n)$	Linearithmic Time
$O(n^2)$	Quadratic Time
$O(2^n)$	Exponential Time

Why do we need data structures?
4-Marks

1] Good organization: They provide a neat, Logical way to arrange information.

2] Showing Relationships: They help us see how different pieces of data relate to one another

3] Handling Growth: Software needs to handle data that grows and shrinks over time. Data structures make this dynamic change possible.

4] Solving Tough Problems: Complex problems — like finding the shortest route on a map (Graphs) or searching through hierarchies (Trees) → Can not be solved easily without them.

Abstract Data types (ADTs) 4 Marks

An abstract data type (ADT) is a formal mathematical model that defines a collection of data objects alongside the operational behaviours permitted on those objects.

An ADT focuses entirely on what operations are performed, completely hiding the implementation details of how they are executed. This paradigm is known as Data Abstraction.

(refer Page 2-3)

Explain Linear & Non Linear Structure

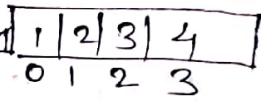
4/6-Marks

Linear Data Structures: In a linear data structure, elements are organized sequentially in a single level sequence. Each Element (Except the first and Last) is connected to a unique predecessor and a unique Successor.

eg. Stack, queue, Linked list array

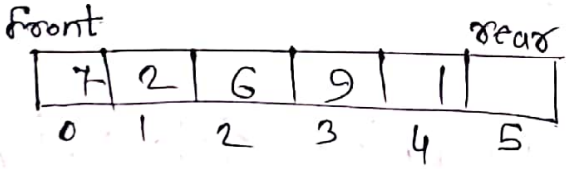
1] Array (Static): A fixed-size list of items stored right next to each other in memory. All items must be of the same Type,

ex. int arr1[4] = {1,2,3,4}; arr1



2] Queue: ~~This~~ (Dynamic): follows the FIFO (first in, first out) rule. Just like a real life line, new items enter at the back (Rear) and leave from the front (front).

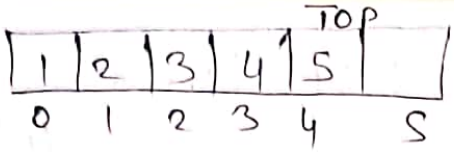
ex.



3] Stack: A stack is a linear data structure in which all insertions (adding elements) and deletions (removing elements) are performed at only one end called the Top.

It operates strictly on the LIFO (Last-in, first-out) principle, meaning the last element added to the stack is the first one to be removed

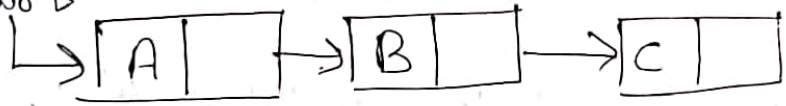
ex.



4] Linked List: A List of items (called nodes) that are not stored next to each other in memory. Instead, they are scattered, and each item points to the next one using a pointer.

Node Structure:
Data field: Holds the actual value
Link field: Holds the address of the next item.

ex. Start



Non-Linear Data Structure: Items are not in a straight line. Instead, they are arranged in levels or interconnected networks.

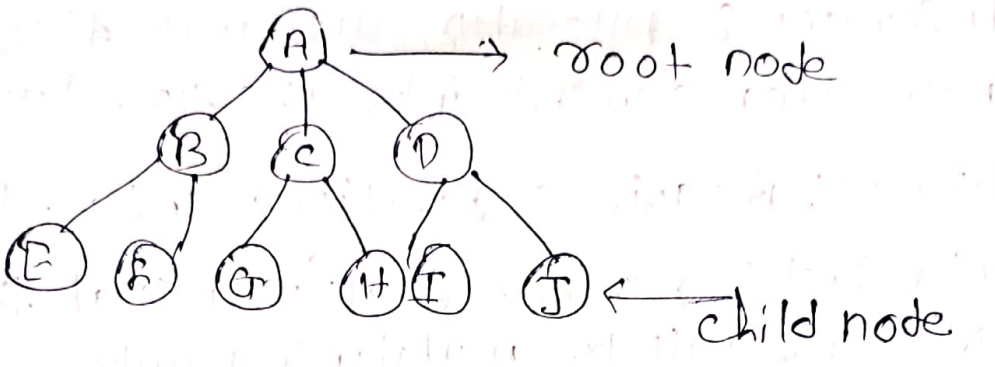
It is multilevel data structure
eg. Tree, Graph.

1] Tree: A Tree organizes data using a Parent-child relationship. It looks like an upside-down tree, starting with a single item at the top and branching downwards

(i) Root: The Top most element (The Starting point)

(ii) Child Nodes: The elements that branch out below a parent node.

Ex.

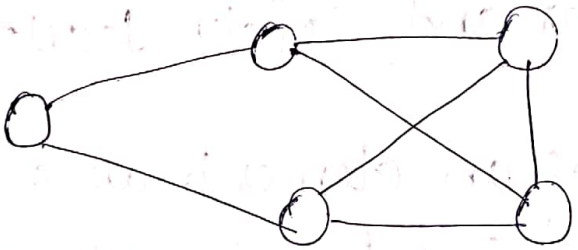


ii] Graph (Network structure): A graph is a collection of points connected by lines. There is no top or bottom; it is a free-form Network.

Vertices (Nodes): The points where data is stored

Edges: The lines or links that connect the points

ex.



Operations on Data Structure: 4-Marks

(i) Traversal: Iterating through the structure to access every single data elements exactly once for processing, searching, or rendering

Array implementation: Looping sequentially through indices 0 to n-1

Linked list implementation: Dereferencing Pointers sequentially from the head node until a null pointer is reached

ii) Insertion: Allocating and introducing a new data element into the structure.

Array: Requires positional index targeting

Linked List: performed efficiently at the head middle or tail by mutating pointer addresses.

Stack/Queue: Executed via standardized Specialized functions: push() for Stacks and enqueue() for Queues.

Deletion: Removing an existing target element from the structure to eliminate redundant data and safely deallocate system memory.

Array: Removes an element at a specific index

Linked List: detaches a targeb node and bridges its predecessor to its successor.

Stack/Queue: Executed via the pop() stack and dequeue() (queue) primitives

2015/26

2015/26

Question Bank

1] Define Data structure. [2-M] [W-15] [W-16]

2] Define ADT [S-22] [W-22] [2M]

3] Applications of Data structure [W-15,23] [S-23]

4] Advantages of Data structure]

5] Explain Linear Data structure [W-15,23] [S-23] [4M]

6] Explain Linear & Non Linear Data structure [W-18] [S-22] [4M]

7] Need of Data structure in brief. [S-16] [4M]

8] Difference between Linear & non Linear data structure [W-22] [4M]

9] what are the types of data structure [W-16] [W-19] [2M]

10] Describe Operations on data structure

11] Define Big 'O' Notation [W-18] [2M]

12] Give time complexity of any four sorting algorithm [2M]

13] write any four operations that can be Performed on data structure [S-23] [W-23] 2M

14] Define an algorithm [S-16,18] [W-17,18] 2M

15] Explain time and space complexity with example. [S-15][S-22][W-15,16,18,19] [4M]

20/5/26

20/5/26