Introduction to Data Structures:

Prof. Nilesh Ghavate

What is Data Structure ?

A data structure is a way of organizing and storing data in a computer or memory in a particular format so that it can be accessed and manipulated efficiently.

It provides a systematic way to manage and organize data elements, allowing for easy retrieval, insertion, deletion, and modification of data.

Types of Data Structure:

Linear Data Structures	Non-Linear Data Structures
● Linear data structures have a sequential order, meaning that each element is connected to its previous and next element.	● Non-linear data structures do not have a sequential order, meaning that elements are connected in a more complex way, such as branching or hierarchical relationships.
● Data elements are arranged sequentially in a linear manner.	● Data elements are arranged in a non-sequential or hierarchical manner.
Examples:	Examples:
1. Arrays: Elements stored in contiguous memory locations with a fixed size.	1. Trees: A hierarchical data structure with nodes connected by edges, with a top node called the root. Common types include binary trees, AVL trees, etc.
2. Linked Lists: A collection of elements (nodes), where each node contains data and a reference	2. Graphs: A collection of nodes connected by edges, where each edge can have different

Types of Data Structure:

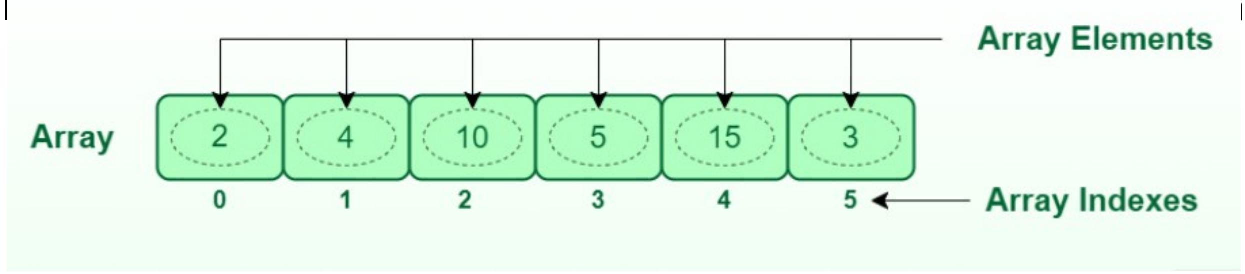
Linear Data Structures	Non-Linear Data Structures
3. Stacks: A Last-In-First-Out (LIFO) data structure where elements are added and removed from the same end.	3. Hash Tables: A data structure that uses hash functions to efficiently store and retrieve data based on key-value pairs.
4. Queues: A First-In-First-Out (FIFO) data structure where elements are added at the rear and removed from the front.	
5. Strings: A sequence of characters stored in a linear order.	

1. Array

The array is a type of data structure that **stores elements of the same type**. These are the most basic and fundamental data structures.

Data stored in each position of an array is given a positive value called the **index** of the element. The index helps in identifying the location of the elements in an array.

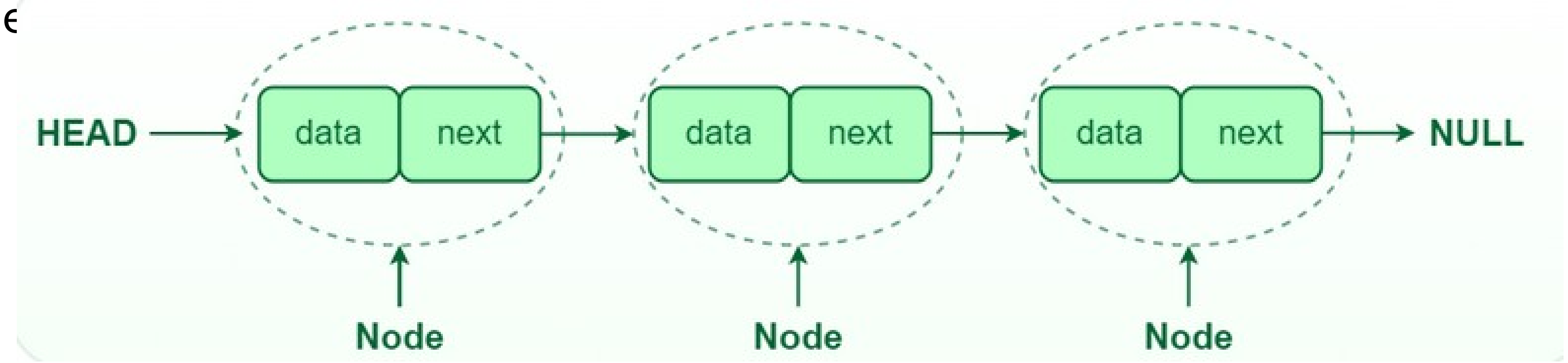
The index val y.



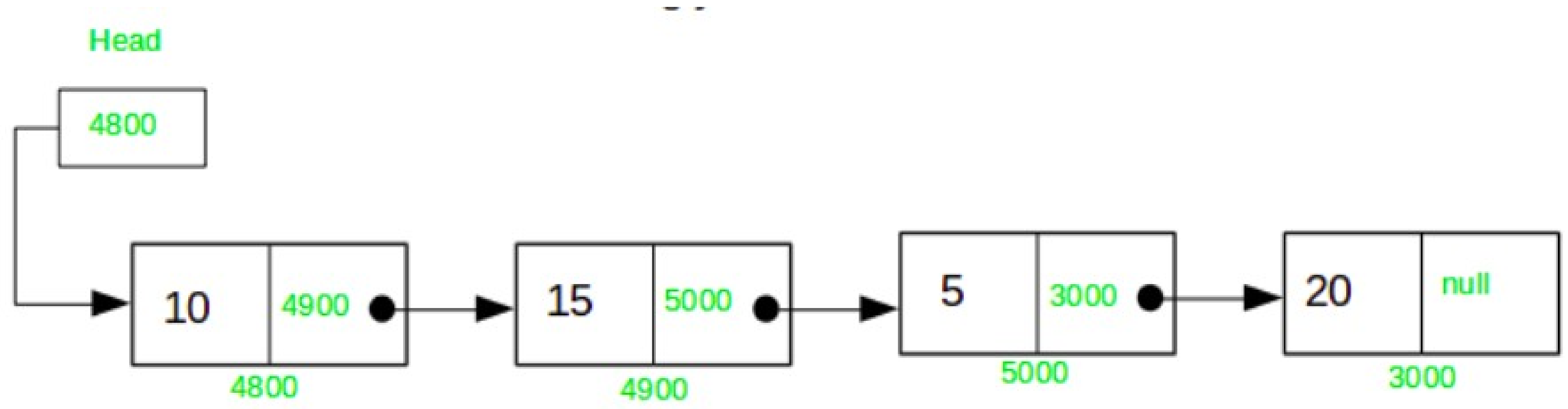
2. Linked List

Linked lists are the types where the data is stored in the form of nodes which consist of an element of data and a pointer. The use of the pointer is that it points or directs to the node which is next to the element in the sequence.

The data stored in a linked list might be of any form, strings, numbers, or characters. Both sorted and unsorted data can be stored in a linked list along with unique



2. Linked List



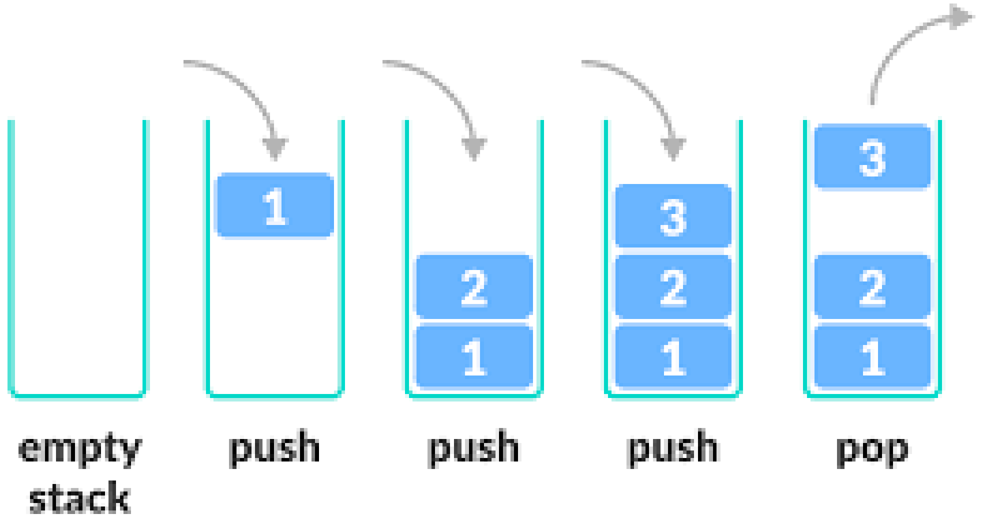
3. Stack

The data structure follows the rule of **LIFO (Last In-First Out)** where the data last added element is removed first.

Push operation is used for adding an element of data on a stack and the pop operation is used for deleting the data from the stack.

This can be explained by the example of books stacked together. In order to access the last book, all the books placed on top of the last book have to be safely removed.

3. Stack


1
2
1
3
2
1
3
2
1
empty stack
push
push
push
pop

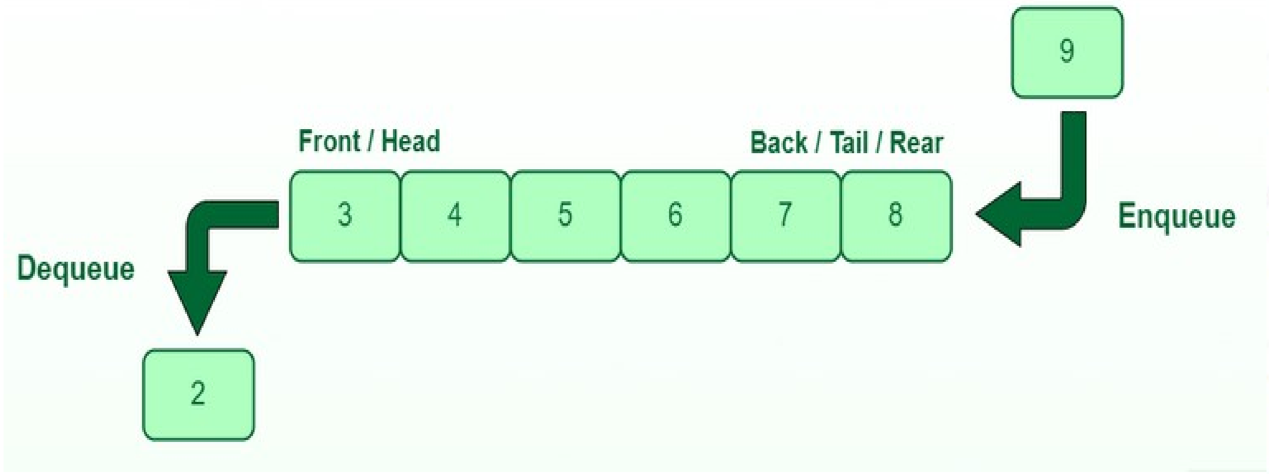
L LAST
I IN
F FIRST
O OUT

4. Queue

This structure is almost similar to the stack as the data is stored sequentially. The difference is that the queue data structure follows **FIFO which is the rule of First In-First Out** where the first added element is to exit the queue first. Front and rear are the two terms to be used in a queue.

Enqueue is the insertion operation and **dequeue** is the deletion operation. The former is performed at the end of the queue and the latter is performed at the start end. The data structure might be explained with the example of people queuing up to ride a bus. The first person in the line will get the chance to exit the queue while the last person will be the last to exit.

4. Queue


9
Front / Head
Back / Tail / Rear
3
4
5
6
7
8
Enqueue
Dequeue
2

F FIRST
I IN
F FIRST
O OUT

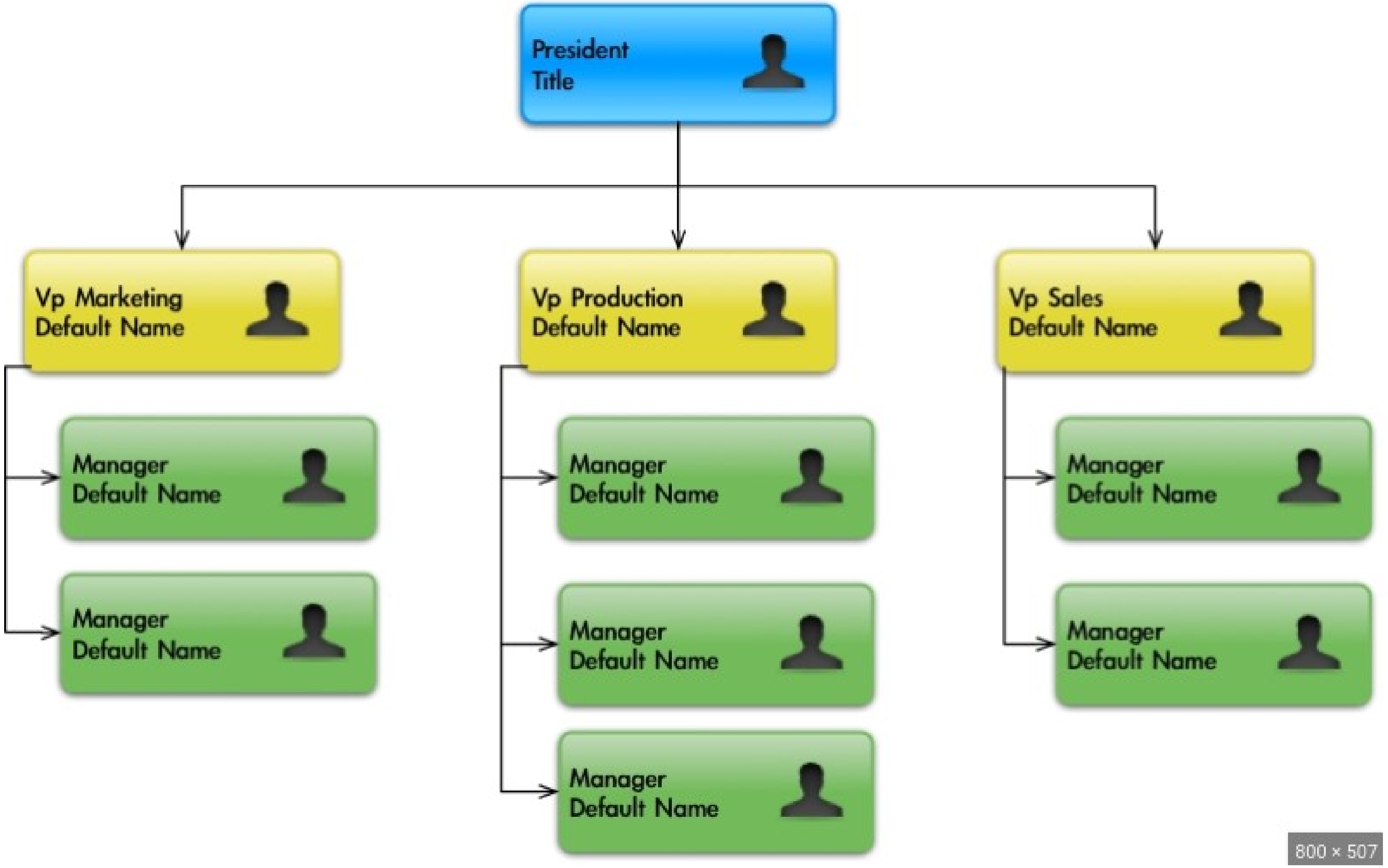
5. Tree

A tree data structure consists of various nodes linked together.

The structure of a tree is hierarchical that forms a relationship like that of the parent and a child. The structure of the tree is formed in a way that there is one connection for **every parent-child node** relationship.

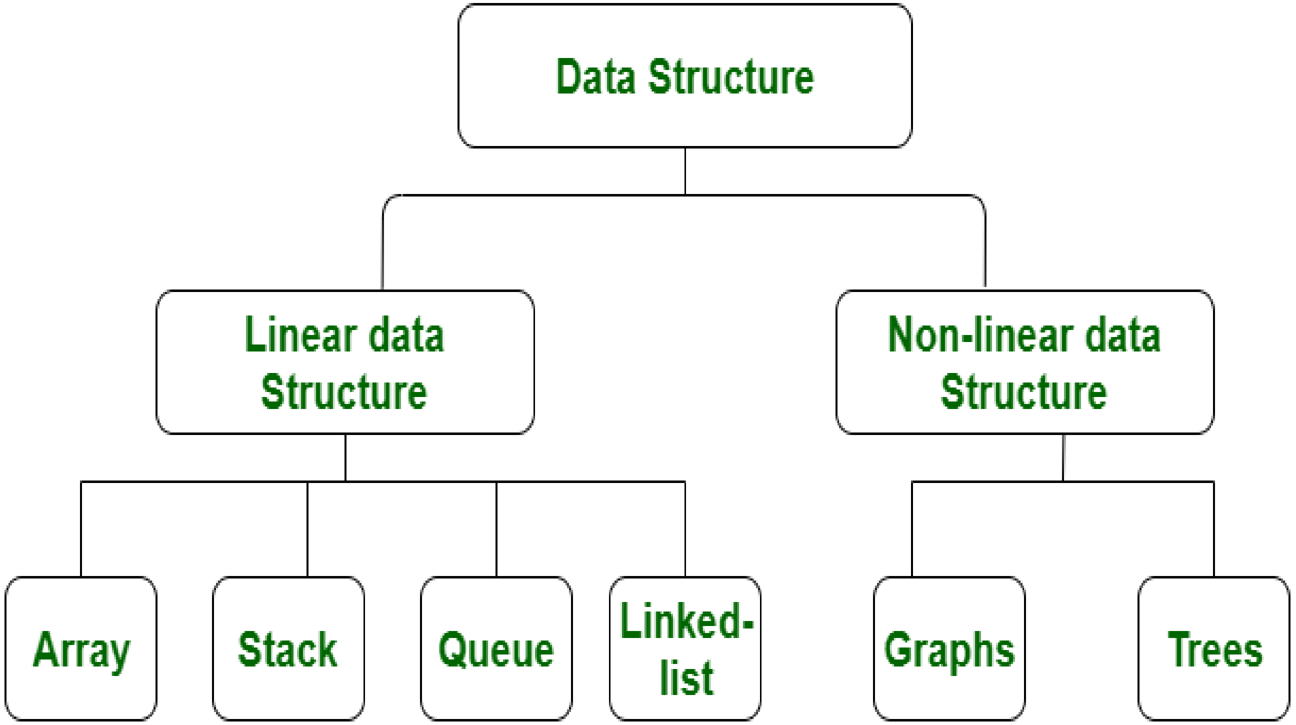
Only one path should exist between the root to a node in the tree. Various types of trees are present based on their structures like **AVL tree, binary tree, binary search tree,** etc.

5. Tree



President
Title
Vp Marketing
Default Name
Vp Production
Default Name
Vp Sales
Default Name
Manager
Default Name
Manager
Default Name
Manager
Default Name
Manager
Default Name
Manager
Default Name
Manager
Default Name
Manager
Default Name
800 × 507

5. Tree


Data Structure
Linear data Structure
Non-linear data Structure
Array
Stack
Queue
Linked-list
Graphs
Trees

6. Graph

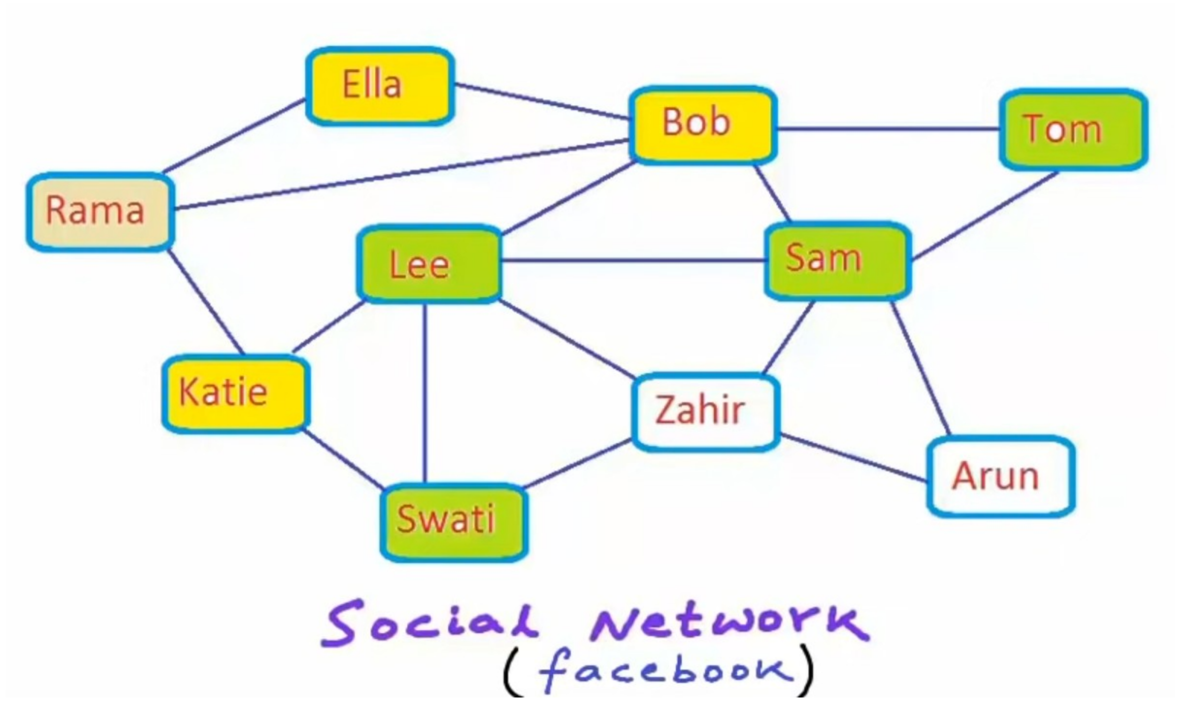
Graphs are those types of non-linear data structures which consist of a definite quantity of vertices and edges.

The vertices or the nodes are involved in storing data and the edges show the vertices relationship.

The difference between a graph to a tree is that in a graph there are no specific rules for the connection of nodes.

Real-life problems like social networks, telephone networks, etc. can be represented through the graphs.

6. Graph


Ella
Bob
Tom
Rama
Lee
Sam
Katie
Zahir
Arun
Swati
Social Network
(facebook)

Can you suggest some friends to Rama?

Static and Dynamic Data Structures

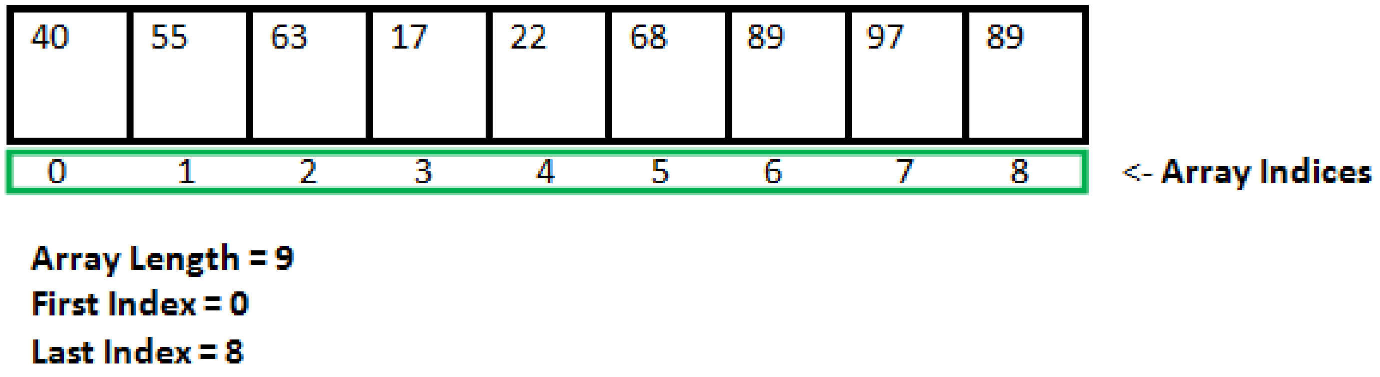
Static Data Structures	Dynamic Data Structures
● Size is fixed and determined at compile time.	● Size can be changed during runtime as needed.
● Memory allocation is done at compile time.	● Memory allocation is done at runtime.
● Requires explicit initialization and deallocation of memory.	● Memory allocation and deallocation are handled automatically.

What is Static Data Structures

In Static data structure the **size of the structure is fixed.**

The content of the data structure can be modified but **without changing the memory space** allocated to it.

Example of Static Data Structures: **Array**

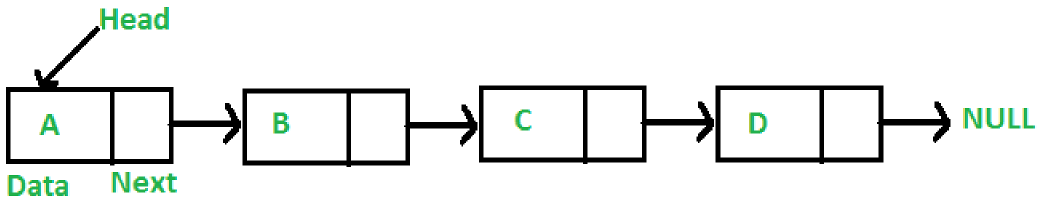


What is Dynamic Data Structures

In Dynamic data structure the **size of the structure in not fixed** and **can be modified** during the operations performed on it.

Dynamic data structures are designed to facilitate change of data structures in the run time.

Example of Dynamic Data Structures: **Linked List**



Concept of Stack and Queue. Array Implementation of Stack and Queue, Circular Queue, Double Ended Queue, Priority Queue.