

Data Structure

A data structure is a **group of data elements that provides the easiest way to store and perform different actions on the data of the computer.**

What Are Arrays in Data Structures?

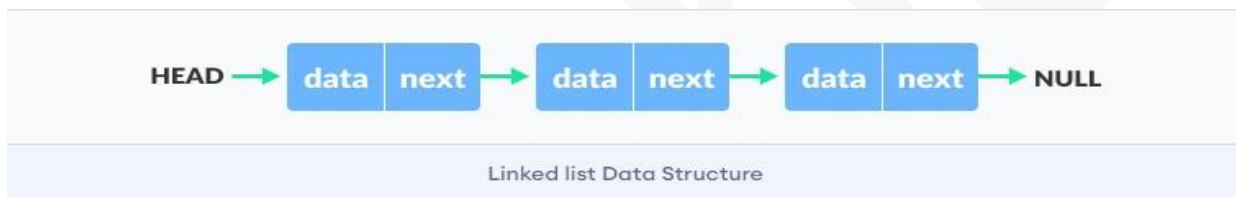
An array is a **linear data structure that collects elements of the same data type and stores them in contiguous and adjacent memory locations.**

Set

A set is a data structure that stores unique elements of the same type in a sorted order. Each value is a key, which means that we access each value using the value itself.

Link List

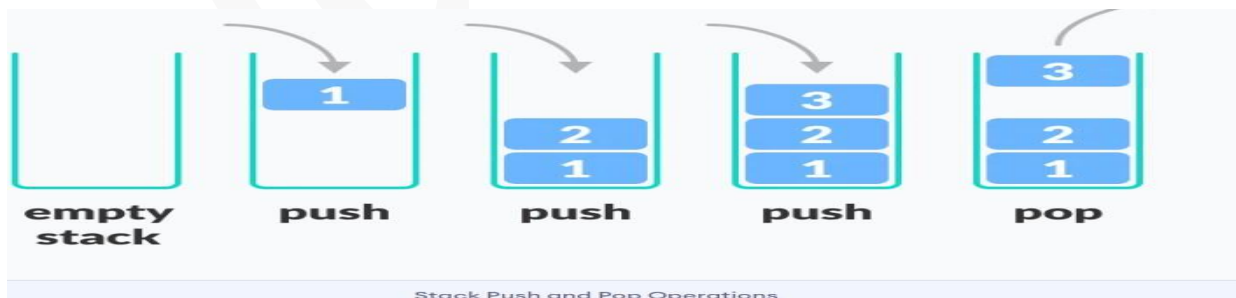
A linked list is a linear data structure that includes a series of connected nodes.



stack

A stack is a linear data structure that follows the principle of **Last In First Out (LIFO)**. This means the last element inserted inside the stack is removed first.

You can think of the stack data structure as the pile of plates on top of another.



queue

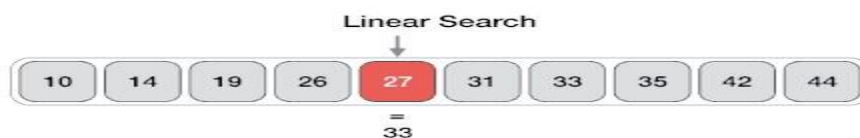
A queue is a useful data structure in programming. It is similar to the ticket queue outside a cinema hall, where the first person entering the queue is the first person who gets the ticket.

Queue follows the **First In First Out (FIFO)** rule - the item that goes in first is the item that comes out first.



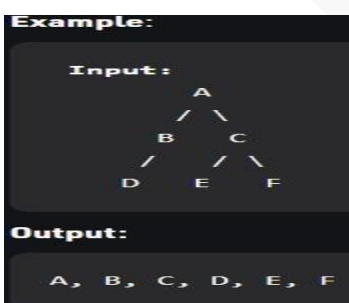
Linear search

Linear search is a very simple search algorithm. In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data collection.



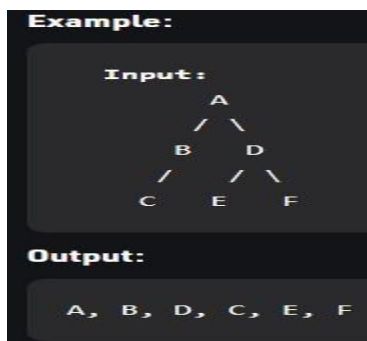
Breadth-First Search:

BFS, Breadth-First Search, is a vertex-based technique for finding the shortest path in the graph. It uses a [Queue data structure](#) that follows first in first out. In BFS, one vertex is selected at a time when it is visited and marked then its adjacent are visited and stored in the queue. It is slower than DFS.



Depth First Search:

DFS, [Depth First Search](#), is an edge-based technique. It uses the [Stack data structure](#) and performs two stages, first visited vertices are pushed into the stack, and second if there are no vertices then visited vertices are popped.



Insertion sort

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

Insertion Sort Execution Example



Bubble sort

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order.

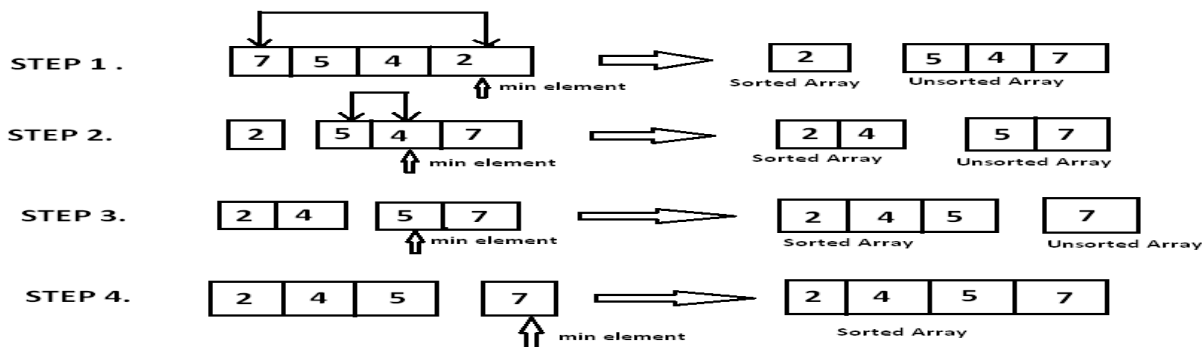
Bubble Sort

First pass	6	2	8	4	10
Next pass	2	6	8	4	10
Next pass	2	6	4	8	10
	2	4	6	8	10

Review complete

Selection sort

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list



The way to write arithmetic expression is known as a notation. An arithmetic expression can be written in three different but equivalent notations, i.e., without changing the essence or output of an expression. These notations are –

- Infix
- Prefix
- Postfix

Infix

Infix notations are normal notations, that are used by us while write different mathematical expressions. The Prefix and Postfix notations are quite different.

Prefix Notation

In this notation, operator is **prefixed** to operands, i.e. operator is written ahead of operands. For example, **+ab**. This is equivalent to its infix notation **a + b**. Prefix notation is also known as **Polish Notation**.

Postfix Notation

This notation style is known as **Reversed Polish Notation**. In this notation style, the operator is **postfixed** to the operands i.e., the operator is written after the operands. For example, **ab+**. This is equivalent to its infix notation **a + b**.

Example

Expression No	Infix Notation	Prefix Notation	Postfix Notation
1	$a + b$	$+ a b$	$a b +$
2	$(a + b) * c$	$* + a b c$	$a b + c *$
3	$a * (b + c)$	$* a + b c$	$a b c + *$
4	$a / b + c / d$	$+ / a b / c d$	$a b / c d / +$
5	$(a + b) * (c + d)$	$* + a b + c d$	$a b + c d + *$
6	$((a + b) * c) - d$	$- * + a b c d$	$a b + c * d -$