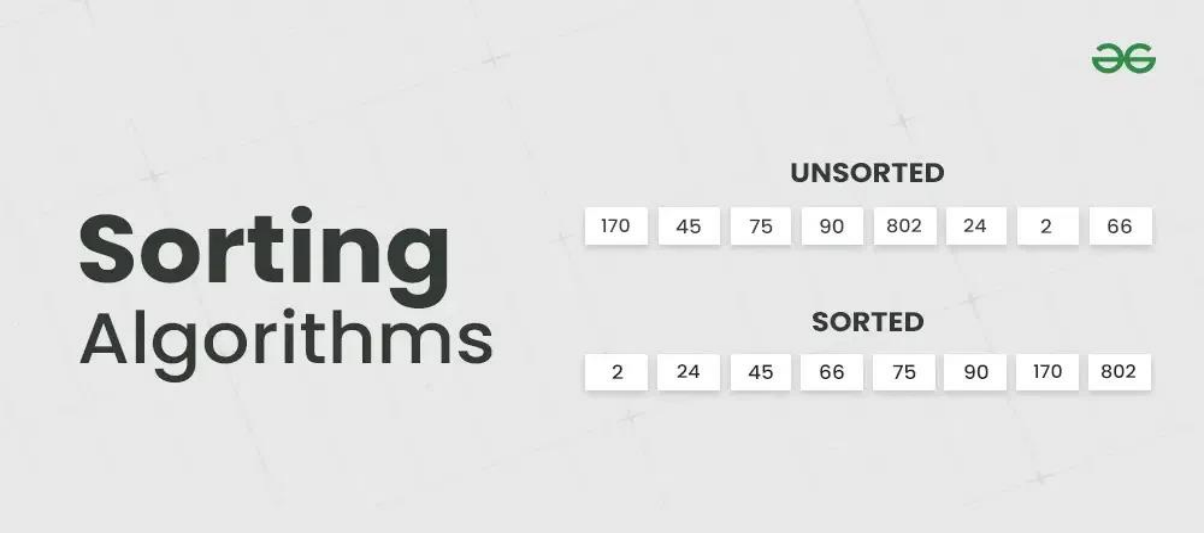
Sorting Algorithms

A **Sorting Algorithm** is used to rearrange a given array or list of elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of elements in the respective data structure.

For Example: The below list of characters is sorted in increasing order of their ASCII values. That is, the character with a lesser ASCII value will be placed first than the character with a higher ASCII value.



Types of Sorting Algorithm

1. Insertion Sort
2. Bubble Sort
3. Selection Sort
4. Quick Sort
5. Merge Sort
6. Heap Sort
7. Counting sort
8. Radix Sort
9. Shell Sort
10. Bucket Sort Etc..

Searching Algorithms

Searching algorithms are essential tools in computer science used to locate specific items within a collection of data. These algorithms are

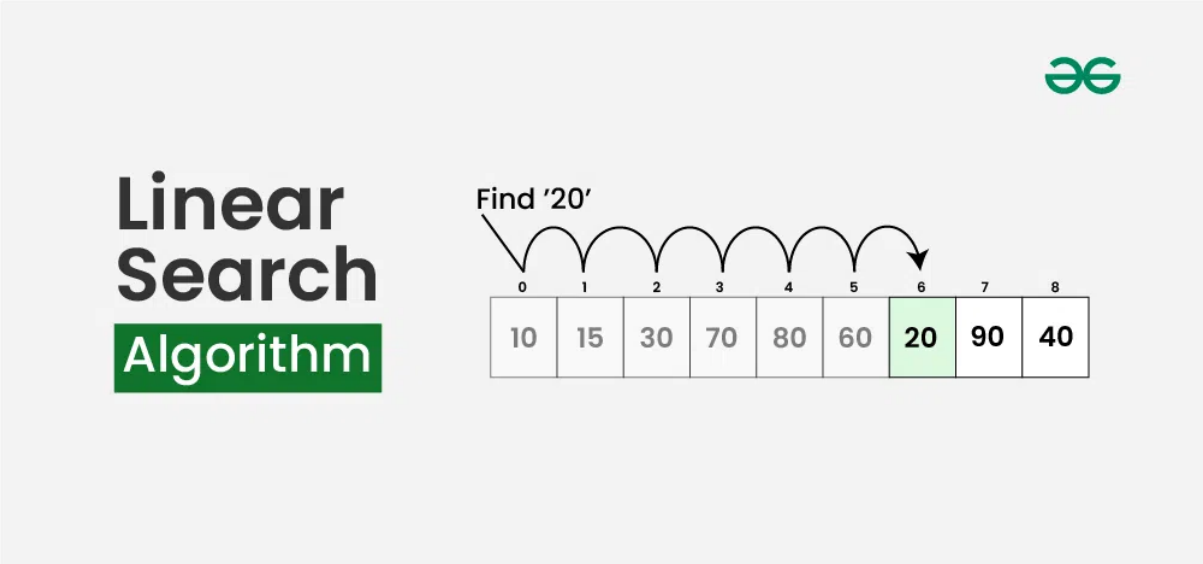
designed to efficiently navigate through data structures to find the desired information, making them fundamental in various applications such as **databases, web search engines**, and more.

Types of Searching algorithms

1. Linear or Sequential Search
2. Binary Search

Linear Search Algorithm

The linear search algorithm is defined as a sequential search algorithm that starts at one end and goes through each element of a list until the desired element is found; otherwise, the search continues till the end of the dataset. In this article, we will learn about the basics of the linear search algorithm, its applications, advantages, disadvantages, and more to provide a deep understanding of linear search.



Tree Data Structure

A **tree data structure** is a hierarchical structure that is used to represent and organize data in a way that is easy to navigate and search. It is a collection of nodes that are connected by edges and has a hierarchical relationship between the nodes. The topmost node of the tree is called the root, and the nodes below it are called the child nodes. Each node can have

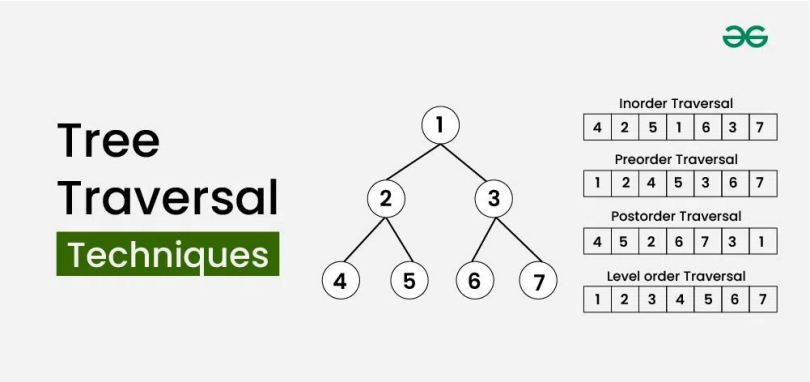
multiple child nodes, and these child nodes can also have their own child nodes, forming a recursive structure.

Types of Tree Data Structure

- **Binary tree:** In a binary tree, each node can have a maximum of two children linked to it. Some common types of binary trees include full binary trees, complete binary trees, balanced binary trees, and degenerate or pathological binary trees.
- **Ternary Tree:** A Ternary Tree is a tree data structure in which each node has at most three child nodes, usually distinguished as “left”, “mid” and “right”.
- **N-ary Tree or Generic Tree:** Generic trees are a collection of nodes where each node is a data structure that consists of records and a list of references to its children(duplicate references are not allowed). Unlike the linked list, each node stores the address of multiple nodes.

Tree Traversal Techniques

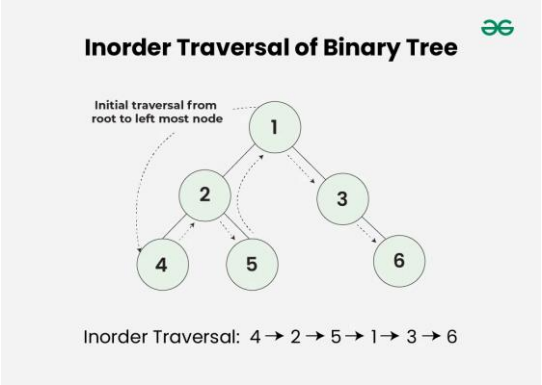
Tree Traversal techniques include various ways to visit all the nodes of the tree. Unlike linear data structures (Array, Linked List, Queues, Stacks, etc) which have only one logical way to traverse them, trees can be traversed in different ways. In this article, we will discuss about all the tree traversal techniques along with their uses.



There are different types of tree traversals, such as **pre-order, in-order, post-order, and level-order.**

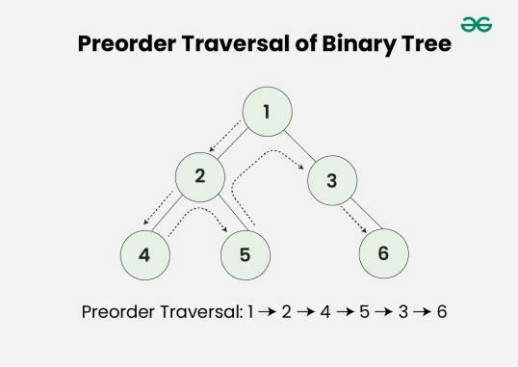
In-Order Traversal

In-order traversal visits the node in the order: Left -> Root -> Right



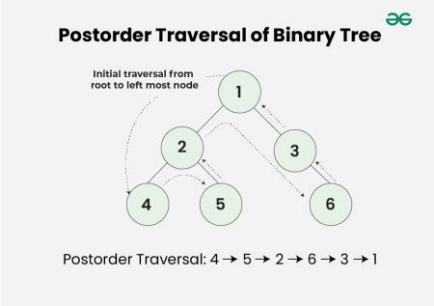
Preorder traversal

Preorder traversal visits the node in the order: **Root -> Left -> Right**



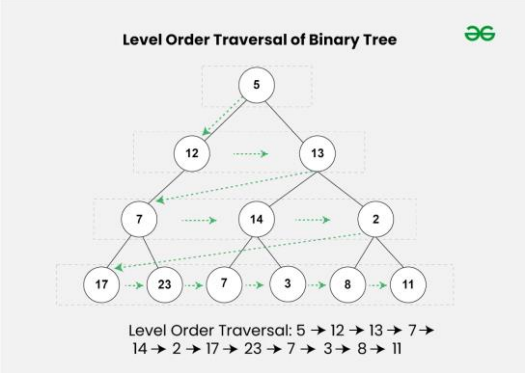
Post-Order Traversal

Postorder traversal visits the node in the order: **Left -> Right -> Root**



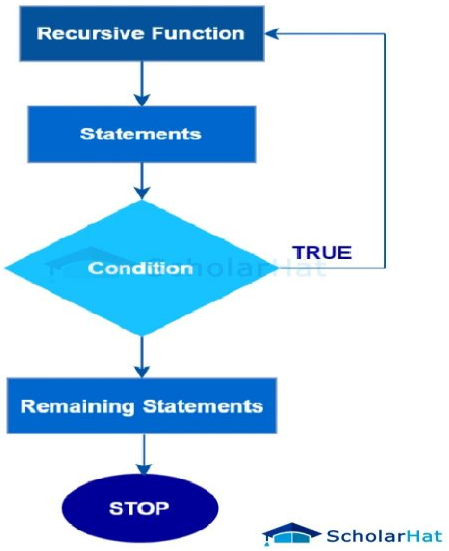
Level Order Traversal

Level Order Traversal visits all nodes present in the same level completely before visiting the next level.



Recursion

Recursion is the process in which a function calls itself again and again. It entails decomposing a challenging issue into more manageable issues and then solving each one again. There must be a terminating condition to stop such recursive calls. Recursion may also be called the alternative to iteration. Recursion provides us with an elegant way to solve complex problems, by breaking them down into smaller problems and with fewer lines of code than iteration.



Recursive Function

A recursive function is a function that calls itself one or more times within its body. A recursive function solves a particular problem by calling a copy of itself and solving smaller subproblems of the original problems. Many more recursive calls can be generated as and when required. It is necessary to have a terminating condition or a base case in recursion, otherwise, these calls may go endlessly leading to an infinite loop of recursive calls and call stack overflow.

The recursive function uses the LIFO (LAST IN FIRST OUT) structure just like the **stack data structure**. A recursion tree is a diagram of the function calls connected by pointed (up or down) arrows to depict the order in which the calls were made.

Syntax to Declare a Recursive Function

```
recursionfunction()

{

recursionfunction(); //calling self function

}
```