# Relation of Algorithm and data structure

# Search Algorithms

On each day, we search for something in our day-to-day life. Similarly, with the case of computer, huge data is stored in a computer that whenever the user asks for any data then the computer searches for that data in the memory and provides that data to the user. There are mainly two techniques available to search the data in an array:

1. Linear search
2. Binary search
3. **Linear Search:** Linear search is a very simple algorithm that starts searching for an element or a value from the beginning of an array until the required element is not found. It compares the element to be searched with all the elements in an array, if the match is found, then it returns the index of the element else it returns -1. This algorithm can be implemented on the unsorted list.
4. **Binary Search:** A Binary algorithm is the simplest algorithm that searches the element very quickly. It is used to search the element from the sorted list. The elements must be stored in sequential order or the sorted manner to implement the binary algorithm. Binary search cannot be implemented if the elements are stored in a random manner. It is used to find the middle element of the list.

# Sorting Algorithms

Sorting algorithms are used to rearrange the elements in an array or a given data structure either in an ascending or descending order. The comparison operator decides the new order of the elements.

**Why do we need a sorting algorithm?**

1. An efficient sorting algorithm is required for optimizing the efficiency of other algorithms like binary search algorithm as a binary search algorithm requires an array to be sorted in a particular order, mainly in ascending order.
2. It produces information in a sorted order, which is a human-readable format.
3. Searching a particular element in a sorted list is faster than the unsorted list.

Selection Sort:

Bubble Sort

Insertion Sort

Merge Sort

Quick Sort

Heap Sort

Counting Sort

# Traversal Algorithms

Traversing in Data Structure means systematically visiting every element of it. Traversing is a process in which each element of a data structure is accessed. Accessing an element of data structure means visiting every element at least once.

Traversing is performed to display every element of data structure or to perform any operation on its element.

Traversing is also known as iterating over the data structure.

## **Tree Traversal**

Traversal is a process to visit all the nodes of a tree. In tree all nodes are connected via edges. In tree traversal we always start from the root (head) node.

1. Depth First Search or DFS

* In-order Traversal
* Pre-order Traversal
* Post-order Traversal

1. Level Order Traversal or Breadth First Search or BFS
2. Boundary Traversal
3. Diagonal Traversal
4. **Pre-Order Traversal (current-left-right):**

* Firstly root is visited
* Then the left subtree is visited
* Then finally the right subtree is visited.



**A → B → D → E → C → F → G**

1. **In-Order Traversal (left-current-right):**

* Firstly left subtree is visited
* Then the root is visited
* The right subtree is visited.



**D → B → E → A → F → C → G**

1. **Post-Order Traversal (left-right-current):**

* Firstly left subtree is visited
* Then the right subtree is visited.
* And at last, the root is visited.



**D → E → B → F → G → C → A**

## Graph Traversal Algorithm

The process of visiting or updating each vertex in a graph is known as graph traversal. The sequence in which they visit the vertices is used to classify such traversals. Graph traversal is a subset of tree traversal.

There are two techniques to implement a graph traversal algorithm:

1. Breadth-first search
2. Depth-first search

### Depth-first search

### Breadth-first search

BFS is the most commonly used approach. It is a recursive algorithm to search all the vertices of a tree or graph data structure. BFS puts every vertex of the graph into two categories - visited and non-visited. It selects a single node in a graph and, after that, visits all the nodes adjacent to the selected node.