# Data Structures

**Intro**

* Data Structure can be defined as the group of data elements which provides an efficient way of storing and organising data in the computer so that it can be used efficiently.
* There are two algorithms to perform searching, Linear Search and Binary Search.
* Pointer is used to points the address of the value stored anywhere in the computer memory. To obtain the value stored at the location is known as dereferencing the pointer.
* A structure is a composite data type that defines a grouped list of variables that are to be placed under one name in a block of memory.

**Data:** Data can be defined as an elementary value or the collection of values, for example, student's name and its id are the data about the student.

**Group Items:** Data items which have subordinate data items are called Group item, for example, name of a student can have first name and the last name.

**Record:** Record can be defined as the collection of various data items, for example, if we talk about the student entity, then its name, address, course and marks can be grouped together to form the record for the student.

**File:** A File is a collection of various records of one type of entity, for example, if there are 60 students in the class, then there will be 20 records in the related file where each record contains the data about each employee.

**Attribute and Entity:** An entity represents the class of certain objects. it contains various attributes. Each attribute represents the particular property of that entity.

**Field:** Field is a single elementary unit of information representing the attribute of an entity.

**Data Structure Classification**

DS Introduction

**Linear Data Structures:** if all of its elements are arranged in the linear order. In linear data structures, the elements are stored in non-hierarchical way where each element has the successors and predecessors except the first and last element.

**Non-Linear Data Structures:** This data structure does not form a sequence i.e. each item or element is connected with two or more other items in a non-linear arrangement. The data elements are not arranged in sequential structure.

### **Operations on data structure**

1) **Traversing**

2) **Insertion**

3) **Deletion**

4) **Searching**

5) **Sorting**

6) **Merging**

**Arrays:** An array is a collection of similar type of data items and each data item is called an element of the array. The data type of the element may be any valid data type like char, int, float or double.

The elements of array share the same variable name but each one carries a different index number known as subscript. The array can be one dimensional, two dimensional or multidimensional.

The individual elements of the array age are:

age[0], age[1], age[2], age[3],??? age[98], age[99].

**Linked List:** Linked list is a linear data structure which is used to maintain a list in the memory. It can be seen as the collection of nodes stored at non-contiguous memory locations. Each node of the list contains a pointer to its adjacent node.

**Stack:** Stack is a linear list in which insertion and deletions are allowed only at one end, called **top**.

A stack is an abstract data type (ADT), can be implemented in most of the programming languages. It is named as stack because it behaves like a real-world stack, for example: - piles of plates or deck of cards etc.

**Queue:** Queue is a linear list in which elements can be inserted only at one end called **rear** and deleted only at the other end called **front**.

It is an abstract data structure, similar to stack. Queue is opened at both end therefore it follows First-In-First-Out (FIFO) methodology for storing the data items.

**Non Linear Data Structures:** This data structure does not form a sequence i.e. each item or element is connected with two or more other items in a non-linear arrangement. The data elements are not arranged in sequential structure.

Types of Non Linear Data Structures are given below:

**Trees:** Trees are multilevel data structures with a hierarchical relationship among its elements known as nodes. The bottommost nodes in the herierchy are called **leaf node** while the topmost node is called **root node**. Each node contains pointers to point adjacent nodes.

Tree data structure is based on the parent-child relationship among the nodes. Each node in the tree can have more than one children except the leaf nodes whereas each node can have atmost one parent except the root node. Trees can be classfied into many categories which will be discussed later in this tutorial.

**Graphs:** Graphs can be defined as the pictorial representation of the set of elements (represented by vertices) connected by the links known as edges. A graph is different from tree in the sense that a graph can have cycle while the tree can not have the one.

# Algorithms

Algorithm is finite set of logic or instructions, written in order for accomplish the certain predefined task.

The major categories of algorithms are given below:

* **Sort**
* **Search**
* **Delete**
* **Insert**
* **Update**

The performance of algorithm is measured on the basis of following properties:

* **Time complexity:** It is a way of representing the amount of time needed by a program to run to the completion.
* **Space complexity:** It is the amount of memory space required by an algorithm, during a course of its execution.

Usually the time required by an algorithm comes under three types:

**Worst case:** It defines the input for which the algorithm takes the huge time.

**Average case:** It takes average time for the program execution.

**Best case:** It defines the input for which the algorithm takes the lowest time.

# Searching

## Linear Search

Linear search is the simplest search algorithm and often called sequential search. In this type of searching, we simply traverse the list completely and match each element of the list with the item whose location is to be found. If the match found then location of the item is returned otherwise the algorithm return NULL.

Linear search is mostly used to search an unordered list in which the items are not sorted.

## Binary Search

Binary search is the search technique which works efficiently on the sorted lists. Hence, in order to search an element into some list by using binary search technique, we must ensure that the list is sorted

Binary search follows divide and conquer approach in which, the list is divided into two halves and the item is compared with the middle element of the list. If the match is found then, the location of middle element is returned otherwise, we search into either of the halves depending upon the result produced through the match.

* A linear search scans one item at a time, without jumping to any item. In contrast, binary search cuts down your search to half as soon as you find the middle of a sorted list.
* Linear search does the sequential access whereas Binary search access data randomly.
* Binary search is better and quite faster than linear search.
* Linear search uses sequential approach. But, binary search implements divide and conquer approach.
* Linear search is quick and easy to use, but there is no need for any ordered elements. Where binary search algorithm is tricky, and elements are necessarily arranged in order.
* Linear search can be implemented in an array as well as in linked list, but binary search can't be implemented directly on linked list.
* Binary search is efficient for the larger array. If the amount of data is small, then linear search is preferable because this searching process is fast when data is small.

# Sorting

# Bubble Sort: Each element of the array is compared with its adjacent element.

1. In Pass 1, A[0] is compared with A[1], A[1] is compared with A[2], A[2] is compared with A[3] and so on. At the end of pass 1, the largest element of the list is placed at the highest index of the list.
2. In Pass 2, A[0] is compared with A[1], A[1] is compared with A[2] and so on. At the end of Pass 2 the second largest element of the list is placed at the second highest index of the list.
3. In pass n-1, A[0] is compared with A[1], A[1] is compared with A[2] and so on. At the end of this pass. The smallest element of the list is placed at the first index of the list.

# Bucket Sort:

Bucket sort is also known as bin sort. It works by distributing the element into the array also called buckets. Buckets are sorted individually by using different sorting algorithm.

# COMB SORT:

Comb Sort is the advance form of Bubble Sort. Bubble Sort compares all the adjacent values while comb sort removes all the turtle values or small values near the end of the list.

Factors affecting comb sort are:

* It improves on bubble sort by using gap of size more than 1.
* Gap starts with large value and shrinks by the factor of 1.3.
* Gap shrinks till value reaches 1.

# Counting Sort:

It is a sorting technique based on the keys i.e. objects are collected according to keys which are small integers. Counting sort calculates the number of occurrences of objects and stores its key values. New array is formed by adding previous key elements and assigning to objects.

# Heap Sort:

Heap sort processes the elements by creating the min heap or max heap using the elements of the given array. Min heap or max heap represents the ordering of the array in which root element represents the minimum or maximum element of the array. At each step, the root element of the heap gets deleted and stored into the sorted array and the heap will again be heapified.

The heap sort basically recursively performs two main operations.

* Build a heap H, using the elements of ARR.
* Repeatedly delete the root element of the heap formed in phase 1.

# Insertion Sort:

Insertion sort is the simple sorting algorithm which is commonly used in the daily lives while ordering a deck of cards. In this algorithm, we insert each element onto its proper place in the sorted array. This is less efficient than the other sort algorithms like quick sort, merge sort, etc.

# Technique

Consider an array A whose elements are to be sorted. Initially, A[0] is the only element on the sorted set. In pass 1, A[1] is placed at its proper index in the array.

In pass 2, A[2] is placed at its proper index in the array. Likewise, in pass n-1, A[n-1] is placed at its proper index into the array.

To insert an element A[k] to its proper index, we must compare it with all other elements i.e. A[k-1], A[k-2], and so on until we find an element A[j] such that, A[j]<=A[k].

All the elements from A[k-1] to A[j] need to be shifted and A[k] will be moved to A[j+1].

# Merge sort

Merge sort is the algorithm which follows divide and conquer approach. Consider an array A of n number of elements. The algorithm processes the elements in 3 steps.

1. If A Contains 0 or 1 elements then it is already sorted, otherwise, Divide A into two sub-array of equal number of elements.
2. Conquer means sort the two sub-arrays recursively using the merge sort.
3. Combine the sub-arrays to form a single final sorted array maintaining the ordering of the array.

# Quick Sort

Quick sort is the widely used sorting algorithm that makes n log n comparisons in average case for sorting of an array of n elements. This algorithm follows divide and conquer approach.

# Selection Sort

In selection sort, the smallest value among the unsorted elements of the array is selected in every pass and inserted to its appropriate position into the array.

First, find the smallest element of the array and place it on the first position. Then, find the second smallest element of the array and place it on the second position. The process continues until we get the sorted array.

The array with n elements is sorted by using n-1 pass of selection sort algorithm.

* In 1st pass, smallest element of the array is to be found along with its index **pos**. then, swap A[0] and A[pos]. Thus A[0] is sorted, we now have n -1 elements which are to be sorted.
* In 2nd pas, position pos of the smallest element present in the sub-array A[n-1] is found. Then, swap, A[1] and A[pos]. Thus A[0] and A[1] are sorted, we now left with n-2 unsorted elements.
* In n-1th pass, position pos of the smaller element between A[n-1] and A[n-2] is to be found. Then, swap, A[pos] and A[n-1].

Therefore, by following the above explained process, the elements A[0], A[1], A[2],...., A[n-1] are sorted.