***PREPARE REPORT ON NONLINEAR DATA STRUCTURE.***

# What is a non-linear data structure?

## Data Structure

A data structure is a special way of organizing the data elements into a particular form. The arrangement of data in a particular order is very important to access the particular data element in less time easily without putting much effort. For example, in our daily life, when we used to put our clothes in a particular drawer properly, especially in a sequence so that whenever we want to wear a particular dress, we may not have to suffer in finding it out and save our time from wasting.

## Types of Data Structure

The Data structures are categorized on two bases:

1. Primitive data structures
2. Non-Primitive data structure

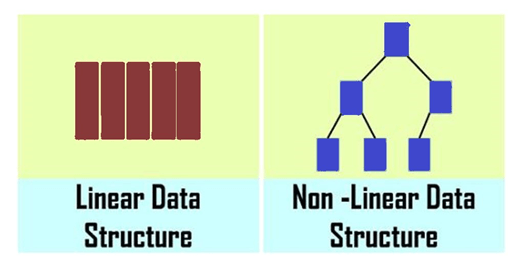
### 1. Primitive data structures

Primitive data structures are nothing but predefined data structures, which are already defined; we do not require giving a particular definition. To derive the non-primitive data structures, we will use these primitive data structures to easily collect a large amount of data. These data structures involve int, float, char, etc. The ' int 'is a data type used to set the type of our data as an integer type; similarly, in this way, we have a float for assigning the type of float to our data used to store the decimal values and so on.

### 2. non-Primitive data structures

The non-primitive data structures are nothing, but the defined data structures used to create particular data structures by using the primitive data structures. It is mainly used to store the collection of elements; it may be of the same data types and may differ depending on the program's need. In non - primitive data structures, we have a concept of Abstract data type. It is a derived data type that the user derives, and the user defines that data type. We must create an abstract data type for using it in many places. There are various non-primitive data structures like an array, linked list, queue, stack, etc.

### ***Types of Non-primitive data structures***



### Linear Data structure:

* The linear data structure is nothing but arranging the data elements linearly one after the other. Here, we cannot arrange the data elements randomly as in the hierarchical order.
* This linear data structure will follow the sequential order of inserting the various data elements. Similarly, in this way, we perform the deletion operation onto the elements. Linear data structures are easy to implement because computer memory is arranged linearly. Examples are **array, stack, queue, linked list, etc**.

Let us discuss some of its types:

**Array:**

* An array is a collection of homogeneous data elements consisting of mainly the same data types.
* An array consists of similar types of data elements present on the contiguous memory locations. Here the word contiguous means consecutive address locations. Suppose our particular array is starting from the address location 1000. Depending on its type, the next element is present at the consecutive memory location with the data type's difference in size.

**Stack:**

* Stack is also one of the important linear data structures based on the LIFO (Last in First Out) principle. Many computer applications and the various strategies used in the operating system and other places are based on the principle of LIFO itself. In this principle, the data element entered last must be popped out first from it, and the element pushed into the stack at the very first time is popped out last. In this approach, we will push the data elements into the stack until it reaches their end limit; after that, we will pop out the corresponding values.

**Queue:**

* A queue is one of the important linear data structures extensively used in various **computer applications**, and also it is based on the **FIFO** (First in First Out) principle. It follows a particular order of data execution for which operations are performed. In this data structure, data enters from one end, i.e., **the REAR** end, and the next data enters the queue after the previous one. Deletion operation is performed from another end, i.e**., FRONT**

**Linked list:**

* The linked list is another major data structure used in various programs, and even many non-linear data structures are implemented using this linked list data structure. As the name suggests, it consists of a link of the node connected by holding the address of the next node. It comes in the portion of the linear data structure because it forms the link-like structure the one data node is connected sequentially with the other node by carrying the address of that node.
* This data is not arranged in a sequential contiguous location as observed in the array. The homogeneous data elements are placed at the contiguous memory location to retrieve data elements is simpler.

Now, let's start with non-linear data structures. Here, we will also discuss the coding part in detail.

## Non-linear data structure

* A non-linear data structure is another important type in which data elements are not arranged sequentially; mainly, data elements are arranged in random order without forming a linear structure.
* Data elements are present at the multilevel, for example, tree.
* In trees, the data elements are arranged in hierarchical form, whereas in graphs, the data elements are arranged in random order, using the edges and vertex.
* Multiple runs are required to traverse through all the elements completely. Traversing in a single run is impossible to traverse the whole data structure.
* Each element can have multiple paths to reach another element.
* The data structure where data items are not organized sequentially is called [a non-linear data structure](https://theknowshares.com/computerscience/datastructure/nonlinear-data-structures/). In other words, data elements of the non-linear data structure could be connected to more than one element to reflect a special relationship among them.

Let us discuss some of its types:

**Trees** and **Graphs** are the types of non-linear data structures.

**Tree:**

* The tree is a non-linear data structure that is comprised of various nodes. The nodes in the tree data structure are arranged in hierarchical order.
* It consists of a root node corresponding to its various child nodes, present at the next level. The tree grows on a level basis, and root nodes have limited child nodes depending on the order of the tree.
* For example, in the binary tree, the order of the root node is 2, which means it can have at most 2 children per node, not more than it.
* The non-linear data structure cannot be implemented directly, and it is implemented using the linear data structure like an array and linked list.
* The tree itself is a very broad data structure and is divided into various categories like **Binary tree, Binary search tree, AVL trees, Heap, max Heap, min-heap**, etc.
* All the types of trees mentioned above differ based on their properties.

**Graph**

* A graph is a non-linear data structure with a finite number of vertices and edges, and these edges are used to connect the vertices.
* The graph itself is categorized based on some properties; if we talk about a complete graph, it consists of the vertex set, and each vertex is connected to the other vertexes having an edge between them.
* The vertices store the data elements, while the edges represent the relationship between the vertices.
* Graphs are very important in various fields; the network system is represented using graph theory and its principles in computer networks.
* Even in Maps, we consider every location a vertex, and the path derived between two locations is considered edges.
* The graph representation's main motive is to find the minimum distance between two vertexes via a minimum edge weight.

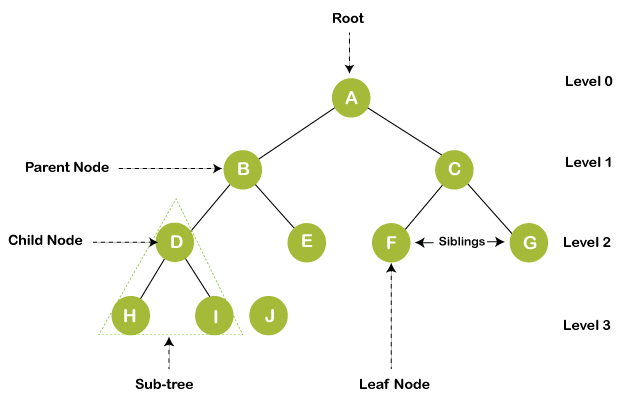
## Properties of Non-linear data structures

* It is used to store the data elements combined whenever they are not present in the contiguous memory locations.
* It is an efficient way of organizing and properly holding the data.
* It reduces the wastage of memory space by providing sufficient memory to every data element.
* Unlike in an array, we have to define the size of the array, and subsequent memory space is allocated to that array; if we don't want to store the elements till the range of the array, then the remaining memory gets wasted.
* So, to overcome this factor, we will use the non-linear data structure and have multiple options to traverse from one node to another.
* Data is stored randomly in memory.
* It is comparatively difficult to implement.
* Multiple levels are involved.
* Memory utilization is effective.

## What is Tree Data structure?

**Trees:** Unlike Arrays, Stack, Linked Lists, and queues, which are linear data structures, trees are hierarchical.

* A tree data structure is a collection of objects or entities known as nodes linked together to represent or simulate hierarchy.
* This data is not arranged in a sequential contiguous location like as we have observed in an array, the homogeneous data elements are placed at the contiguous memory location so that the retrieval of data elements is simpler.
* A tree data structure is **non-linear** because it does not store sequentially. It is a hierarchical structure as elements in a Tree are arranged in multiple levels.
* The topmost node in the Tree data structure is known as a **root node.** Each node contains some data, and data can be of any type. The node contains the employee's name in the tree structure, so the data type would be a string.
* Each node contains some data and the link or reference of other nodes that can be called children.



### Terminologies used in tree:

1. **Root node:** The tree consists of a root node; it is the starting node of the tree from which any tree grows. The root node is present at level 0 in any tree. Depending on the order of the tree, it can hold many child nodes. For example, if in a tree the order is 3, then it can have at most three child nodes, and minimum, it can have 0 child nodes.
2. **Child node:** The child node is the node that comes after the root node, which has a parent node and has some ancestors. It is the node present at the next level of its parent node. For example, if any node is present at level 5, it is sure that its parent node is present at level 4, just above its level.
3. **Edge:** Edges are nothing but the link between the parent node, and the children node is called the link or edge connectivity between two nodes.
4. **Siblings:** The nodes having the same parent node are called siblings of each other. Same ancestors are not siblings, and only the parent node must be the same, defined as siblings. For example, if node A has two child nodes, B and C, B and C are called siblings.
5. **Leaf node:** The leaf node is the node that does not have any child nodes. It is the termination point of any tree. It is also known as External nodes.
6. **Internal nodes:** Internal nodes are non-leaf nodes, having at least one child node with child nodes.
7. **Degree of a node:** The degree of a node is defined as the number of children nodes. The degree of the leaf node is always 0 because it does not have any children, and the internal node always has at least one degree, as it contains at least one child node.
8. **Height of the tree:** The tree's height is defined as the distance of the root node to the leaf node present at the last level is called the height of the tree. In other words, height is the maximum level up to which the tree is extended.