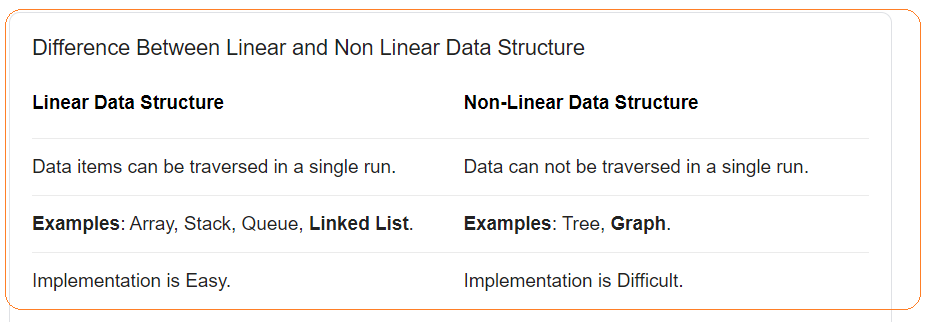
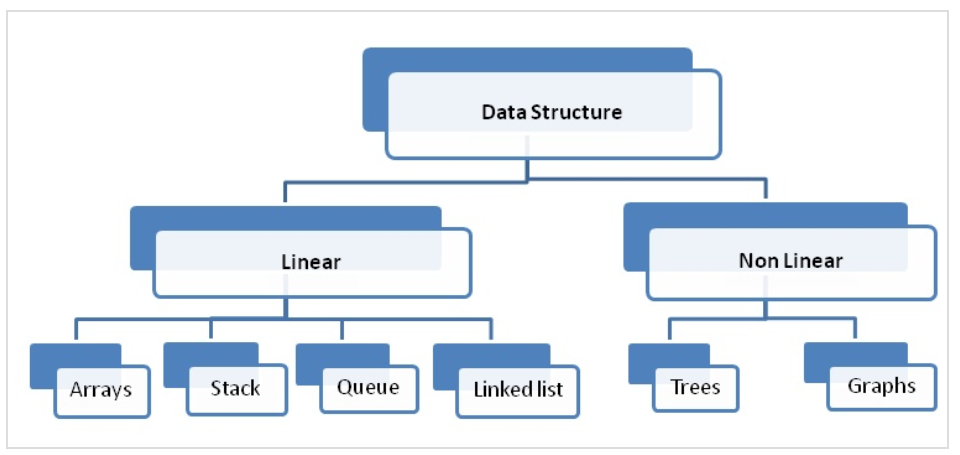
# LINEAR DATASTRUCTURES vs NON LINEAR





# LINKED LIST

Linked List is a linear data structure where each element is a separate object.

“+”:

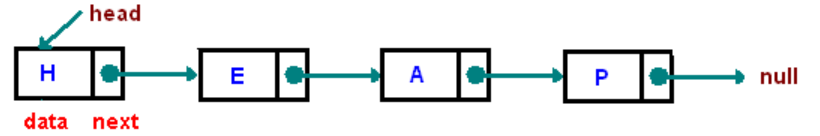
“-“:

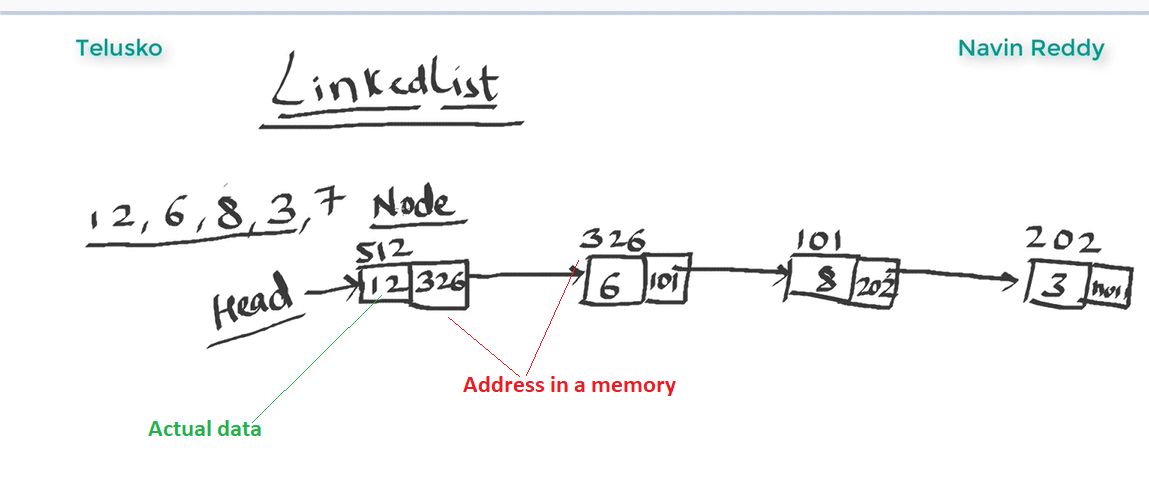
* One disadvantage of a linked list against an array is that it does not allow direct access to the individual elements. If you want to access a particular item then you have to start at the head and follow the references until you get to that item.
* Another disadvantage is that a linked list uses more memory compare with an array - we extra 4 bytes (on 32-bit CPU) to store a reference to the next node.

**Node** - Each element (we will call it a node) of a list is comprising of two items - the data and a reference to the next node. The last node has a reference to null.

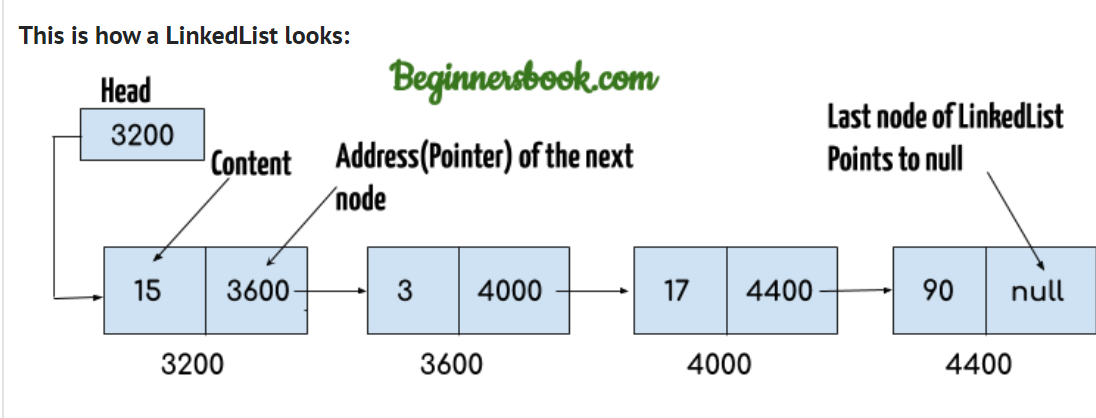
**Head** - The entry point into a linked list is called the head of the list. It should be noted that head is not a separate node, but the reference to the first node. If the list is empty then the head is a null reference.

The number of nodes in a list is not fixed and can grow and shrink on demand. The number of nodes in a list is not fixed and can grow and shrink on demand.





**Head** of the LinkedList only contains the Address of the **First element** of the List.

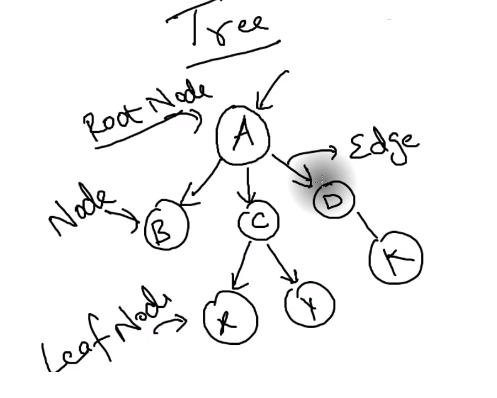


Arrays disadvantages:

Arrays are expensive to add new elements

# TREE

Tree – actually an instance of more general category called [graph]



**Leaf Node** – node without child

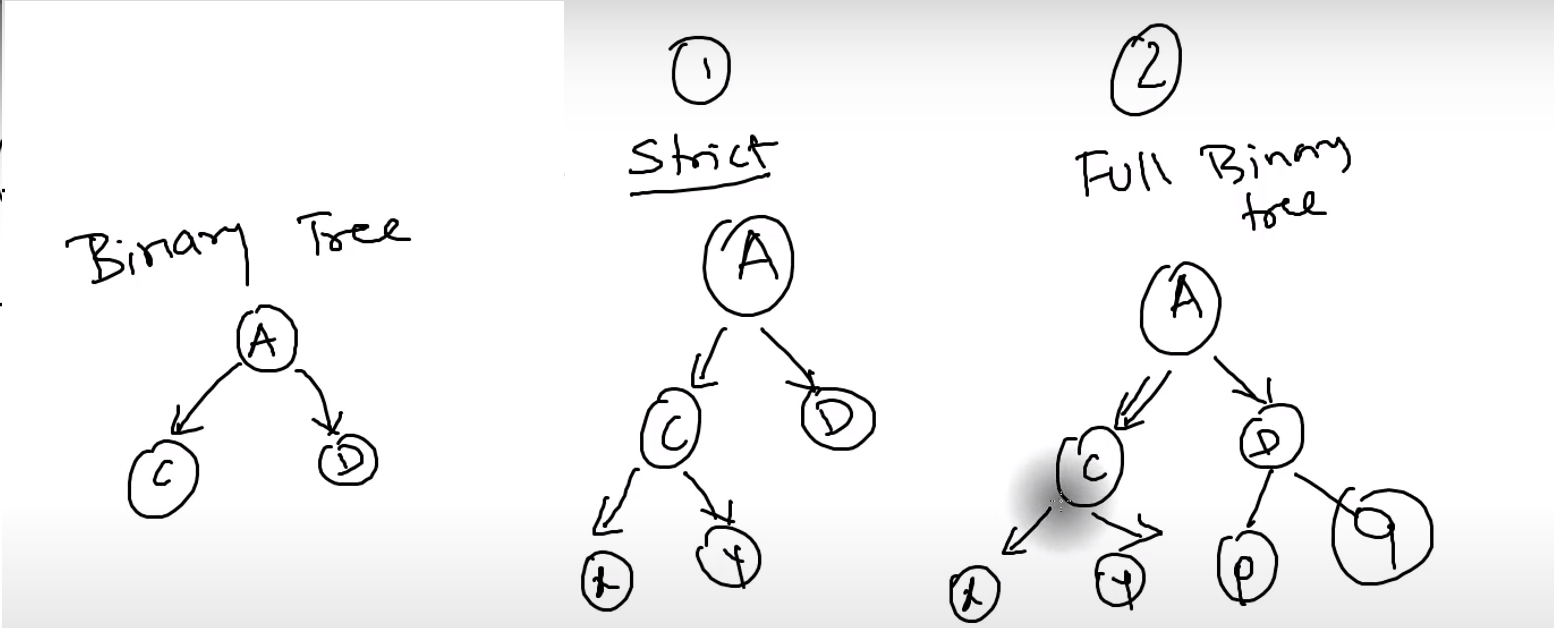
**Edge** – connection between nodes

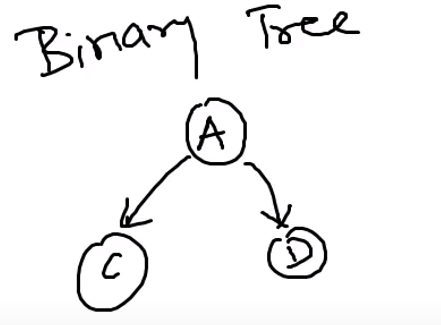
**Traversing** – means visit all nodes in a specified order

**Binary Tree** – is a tree where each node can have not more than 2 child

**Strict Binary Tree** – when node has 2 or 0 child

**Full Binary Tree** – when all children are on the same level



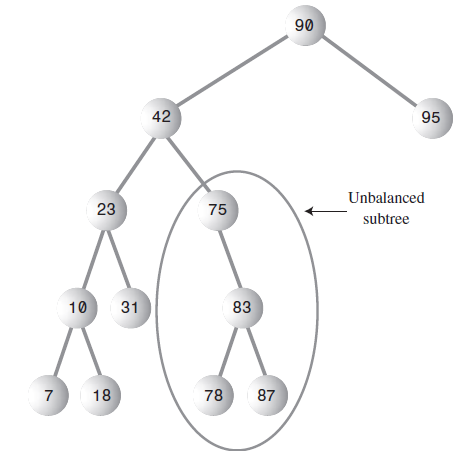


# BINARY TREE

Finding an element in [linkedlist] is not easy task. You need stat from a head to search it and for Deletion – the same case

**Binary Tree** – is a tree where each node can have not more than 2 child

**Unbalanced Tree** – tree that have most of the nodes on one side



The most common way to store the nodes of Binary Tree is to store nodes in memory and connect them using references to each nodes

Complexity of finding a node depends on number of levels. This is O(logN) time

Node: *[binary tree] and [binary search tree] are different trees*.

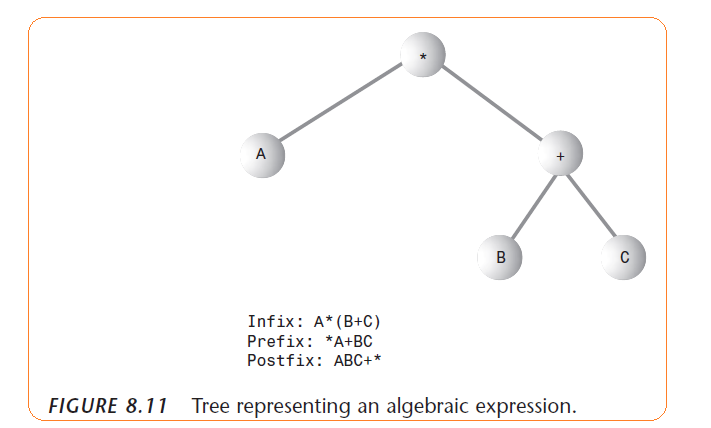
**Traversing** – means visit all nodes in a specified order

There are three ways to traverse tree:

* **inorder** - all nodes visited in ascending order
* **preorder** -
* **postorder** –

note:

* the most common way to traverse is [***inorder***].
* [preorder] and [postorder] can be used in [binary tree] to parse algebraic expressions

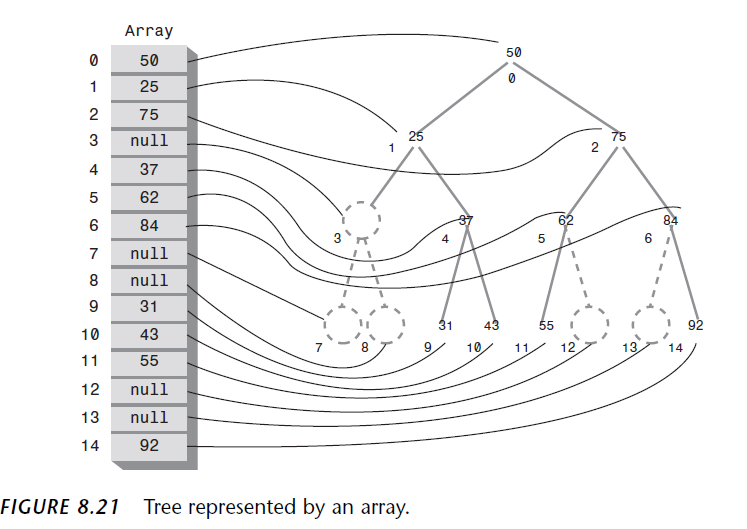


Approaches to traverse tree

* recursive – the simplest way
* iterative –

Representing tree ways:

* Based on [leftChild] and [rightChild]
* Based on [array] – in most situations this way is not efficient



LINKS:

Linked list

<https://www.cs.cmu.edu/~adamchik/15-121/lectures/Linked%20Lists/linked%20lists.html>

* Linear vs Non Linear Data Structures

<https://knowshares.wordpress.com/2016/12/14/linear-vs-non-linear-datastructure/>