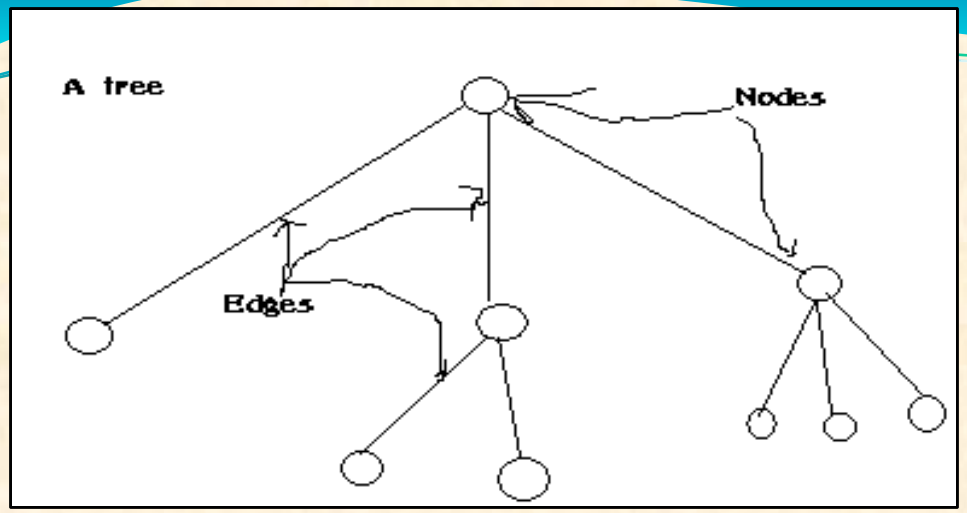
CSC 224: Data Structures

Bsc. (Computer Science)

By: Omieno Kelvin

TREE ADT

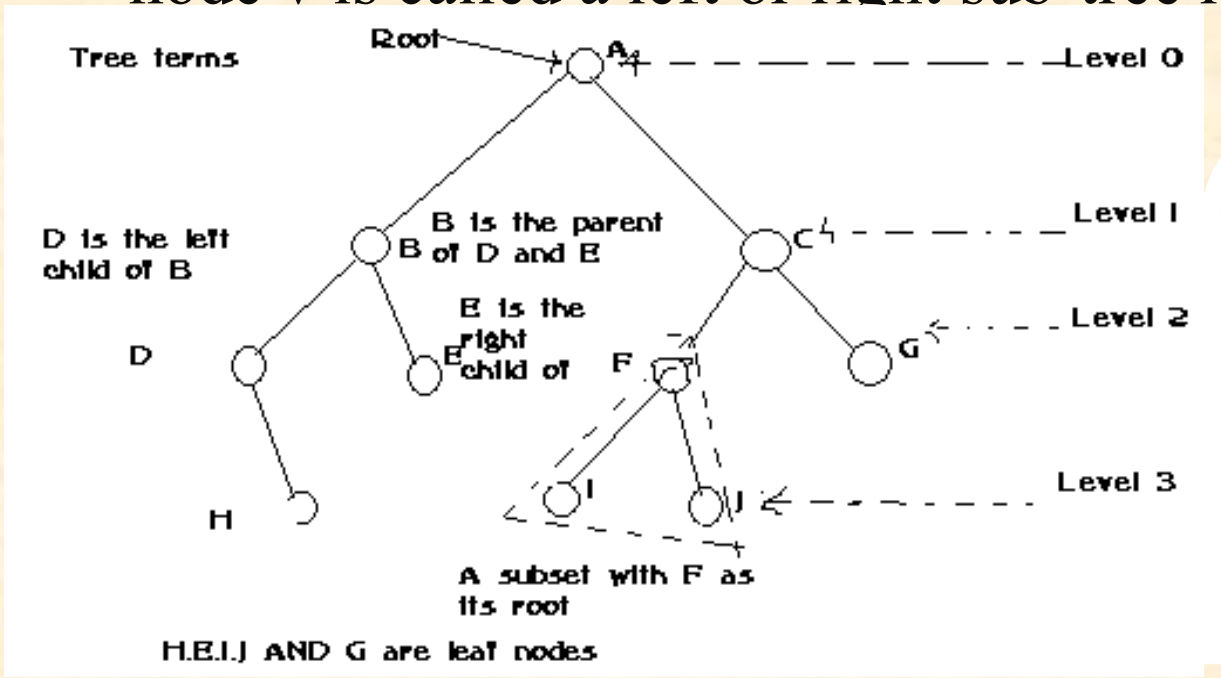
- External node/ leave- do not have children.
- Internal node- has children.
- A tree contain a finite set of elements called nodes or vertices and a finite set of directed arcs connect pairs of nodes.
- If tree is non-empty then one of the nodes (root)- has no incoming arcs but every other node on the tree can be reached by following a unique sequence of consecutive arcs.
- A tree consists of nodes connected by edges.
- A tree is an instance of a graph.

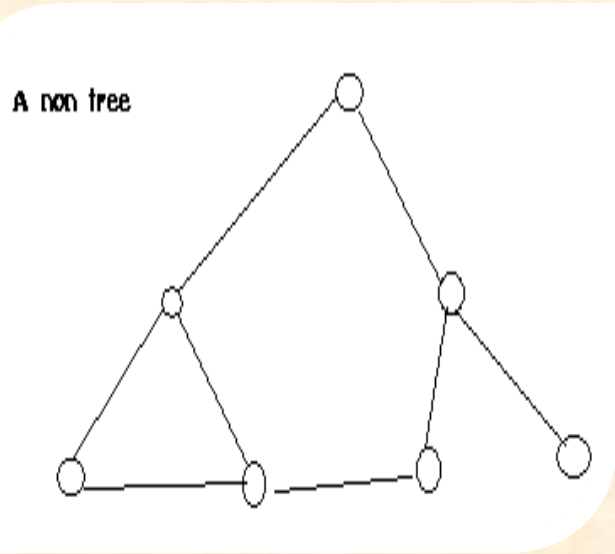


- A node is an ancestor of itself.
- The height of a tree- the no. of paths to the highest node.
- Depth- the no. of arcs from a node.
- If a node u is a parent of node v, then we say that v is a child of u.
- 2 nodes that are children of the same parent are siblings.
- A node is external-if it has no children and internal – if it has one or more children. *External are also called Leaves*.

Binary tree.

- Is an ordered tree in which every node has utmost 2 children.
- A B-tree is proper if @ node has either 0 or 2 children- proper B-tree , every internal node has exactly 2 children.
- The sub-tree rooted at the left or right child in the internal node v is called a left or right sub-tree resp. over v.



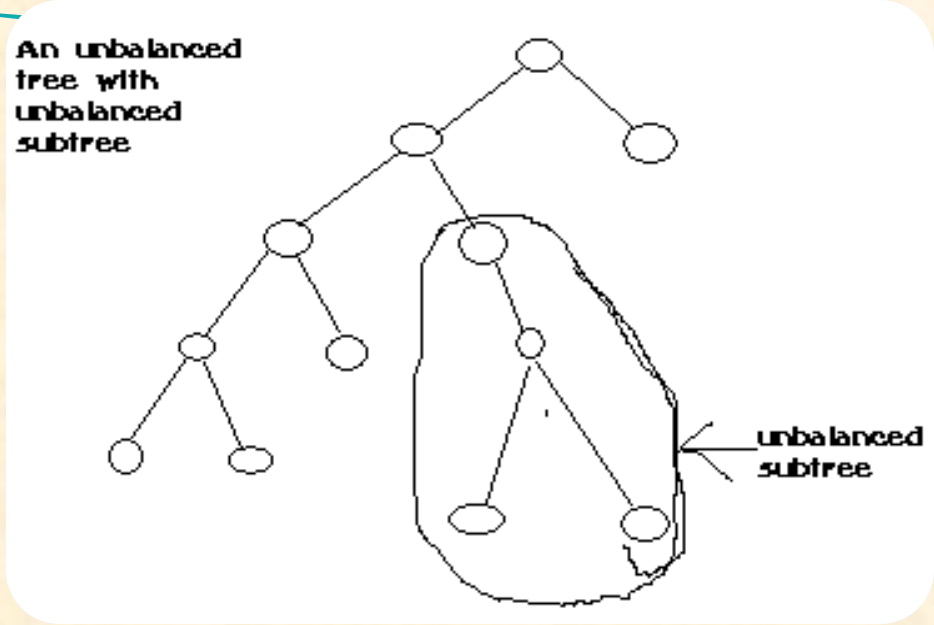


Other terminologies

- Path- sequence of nodes
- Root- node at the top of the tree
- Visiting- a node is visited when program control arrives at the node and performing a operation ; passing over a node on the path from one node-is not necessarily visiting the node.
- Traverse- visit all nodes in some specified manner
- Key- value used to search for an item or perform other operations on the tree

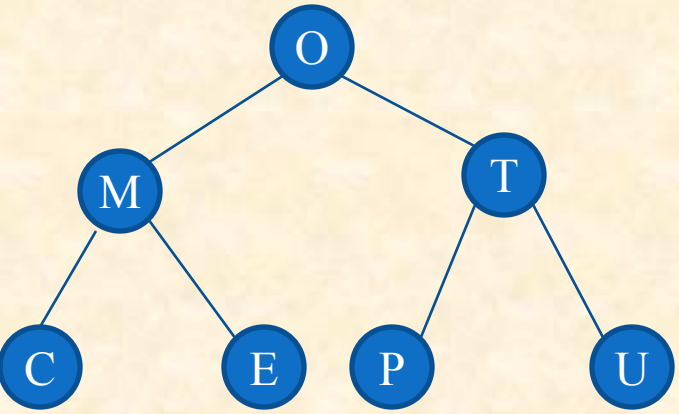
E.g. of tree- The root directory of a given device (designated with the backslash, as in C:\, on many systems) is the tree's root.

- Unbalance tree- they have most of their nodes on one Side of the root or the other.



- Trees become unbalanced because of the order in which the data items are inserted. If these key values are inserted randomly, the tree will be more or less balanced. However, if an ascending sequence (like 11, 18, 33, 42, 65, and so on) or a descending sequence is generated, all the values will be right children (if ascending) or left children (if descending) and the tree will be unbalanced.

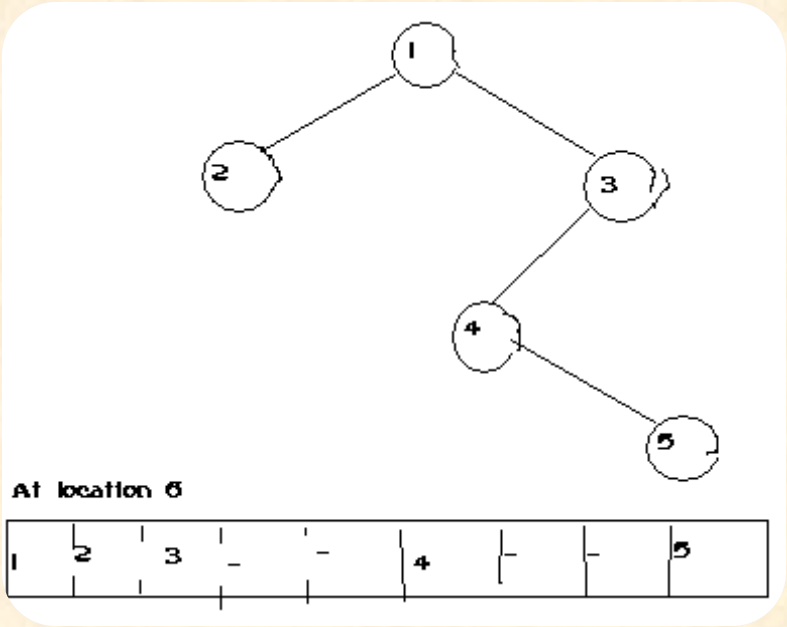
Array representation of B-tree



I	t[i]
0	O
1	M
2	T
3	C
4	E
5	P
6	U

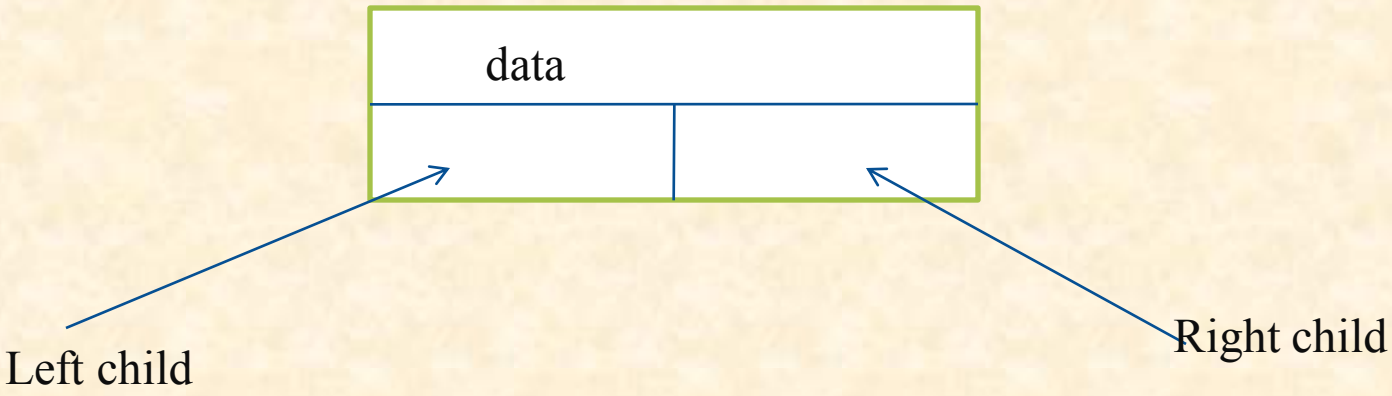
This is a table illustration location of array elements from a Binary tree.

- An array can be used to store B-trees.
- We simply no. the nodes in the tree from the root down numbering the nodes from each level from left to right and store contents of the ith node in the ith location of the array.
- This array-based implementation works very well for complete trees but may be space inefficient for other kinds of trees.



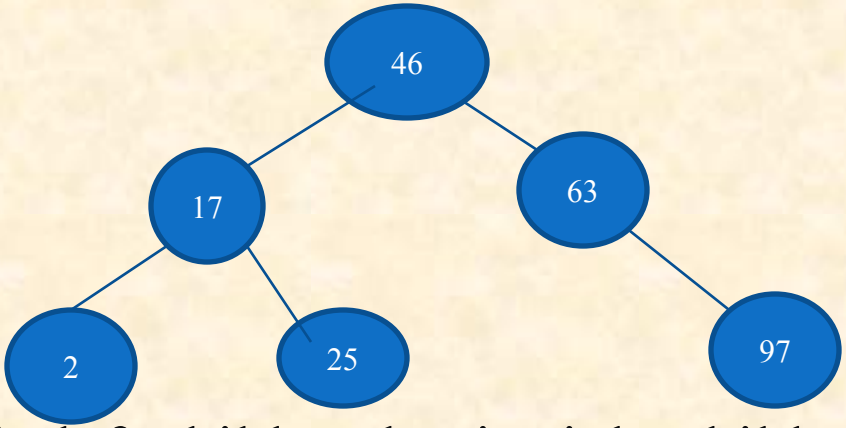
Linked list implementation of B-trees.

- to use space more efficiently, and to provide additional flexibility, we implement B-trees as linked structures in which each node has 2 links- one pointing to the left child of that node and the other pointing to the right child.



Binary search tree(BST)

- A B-tree in which for each node x, value in left child of x is less than value in right child of x.

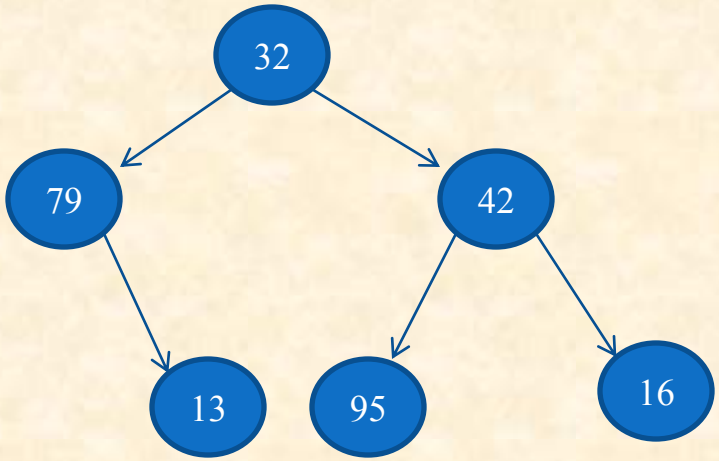


- Value in left child<value in right child
- The simplest form of tree is a **binary tree**. A binary tree consists of :

a) a *node* (called the **root** node) and

b) left and right sub-trees.
Both the sub-trees are themselves binary trees.

- In a B-tree, a node's left child must have a key less than its parent, && a node's right child must have a key have a key greater than or equal to its parent.
- The nodes at the lowest levels of the tree (the ones with no sub-trees) are called **leaves**.
- In an *ordered binary tree*:-
- The keys of all the nodes in the left sub-tree are less than that of the root,
- The keys of all the nodes in the right sub-tree are greater than that of the root,
- The left and right sub-trees are themselves ordered binary trees



- Inorder: LNR:-79,13,32,95,42,16
- Preorder:NLR:-32,79,13,95,42,16
- Postorder:LRN:-13,79,95,16,42,32

Tree Traversals-

- Traversing a tree means visiting each node in a specified order
- There are three simple ways to traverse a tree. They're called *preorder*, *inorder*, and *postorder*.
- An *in order traversal* of a binary search tree will cause all the nodes to be visited in ascending order, based on their key values {C++ code }

```
void inOrder(Node* pLocalRoot)
{
   if(pLocalRoot != NULL)
   {
     inOrder(pLocalRoot->pLeftChild); //left child
     cout << pLocalRoot->iData << " "; //display node
      inOrder(pLocalRoot->pRightChild); //right child
   }
  }
```