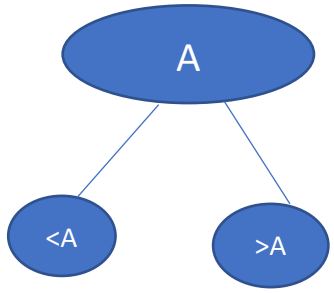
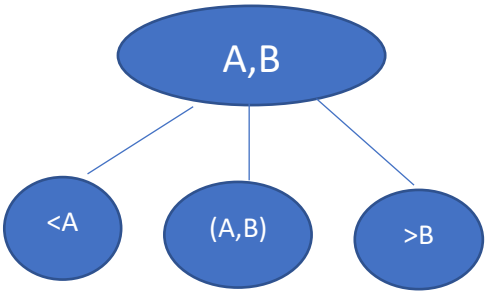
2-3 TREES

A **2-3 Tree** is a tree data structure where every node with children has either two children and one data element or three children and two data elements. A node with 2 children is called a **2-NODE** and a node with 3 children is called a **3-NODE**. A **4-NODE**, with three data elements, may be temporarily created during manipulation of the tree but is never persistently stored in the tree. It is a special type of **B-Tree** with order 3.

2-Node:



3-Node:



PROPERTIES OF 2-3 TREES:

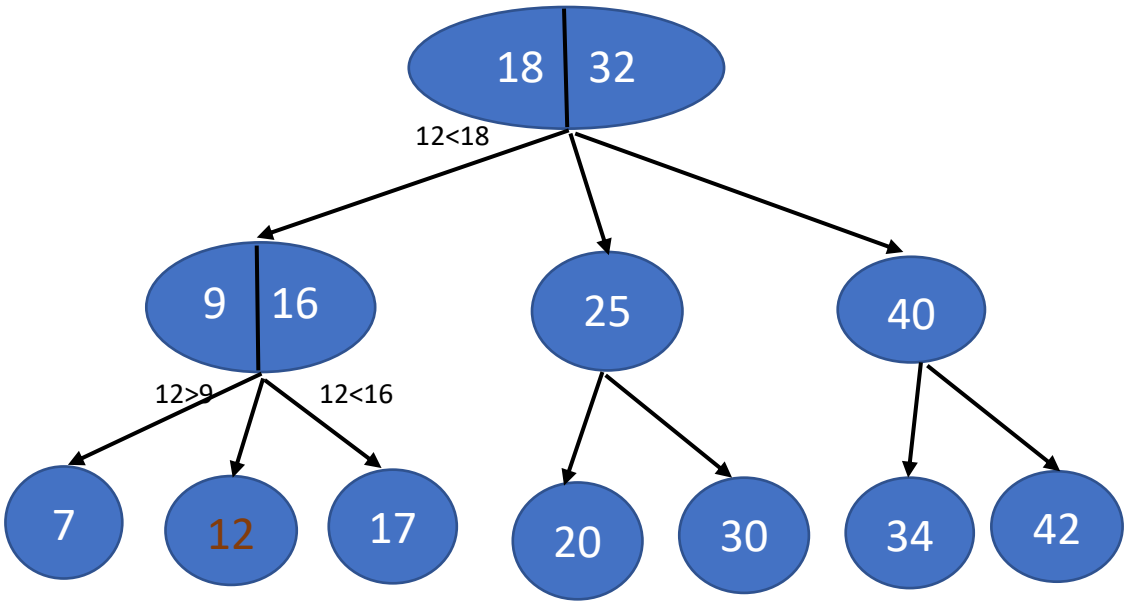
- Each node has one or two keys
- All leaves are at the same level
- Each internal node has 1 key and ***2*** children or 2 keys and ***3*** children.
- Data is stored in sorted order.
- It is a balanced tree.
- Always insertion is done at leaf.
- Search operations are fast and efficient in a 2-3 tree.

Operations in a 2-3 Tree:-

1. Searching in a 2-3 tree

- To search an element in a 2-3 tree.
- If tree is empty then return False
- If we reach root node then also return False means the element is not found.
- If the element is less than the left data part, then search the left subtree
- If the element is greater than left data and more than right data, search the middle subtree.
- If the element is greater than the right data part, then search right subtree.
- It is similar to the binary search tree operation.

SEARCH FOR 12:



2. Inserting a node in a 2-3 tree

- To insert an element in a 2-3 tree.
- If the tree is empty then add that element as the root of the 2-3 tree.
- Search for the right position of that element and add it as a leaf node.

- If there is a leaf node with one data part, add that element with and leaf node becomes a 2 - Node.
- If the leaf node has two data parts, add that element and Split temporary 3-nodes and move data to parent nodes according to the sorting order.
-

3. Deletion of a node in a 2-3 tree

- To delete an element in a 2-3 tree.
- If the tree is empty return false.
- Search for the position of that element and delete it, then adjust the tree.
- If element is part of 3 node delete that element and adjust left value and middle value. Also adjust node’s ancestor’s left and middle value if necessary.
- If element is part of 2 node then adjust and split the tree in a recursive manner arranging nodes in sorted order.