B Tree

B-tree is a special type of self-balancing search tree in which each node can contain more than one key and can have more than two children. It is a generalized form of the binary search tree.

number of keys in a node and number of children for a node depends on the order of B-Tree.

It is also known as a height-balanced m-way tree.

Properties of B tree

Property #1 - All leaf nodes must be at same level.

Property #2 - All nodes except root must have at least [m/2]-1 keys and maximum of m-1 keys.

If m = 4 then
Min keys = 4/2-1 =1
Max keys = 4-1 = 3

Property #3 - All non leaf nodes(internal node) except root (i.e. all internal nodes) must have at least m/2(ceiling) children.

Property #4 - If the root node is a non leaf node, then it must have atleast 2 children.

Property #5 - A non leaf node with n-1 keys must have n number of children.

Property #6 - All the key values in a node must be in Ascending Order.

Property #7- The left subtree of the node will have lesser values than the right side of the subtree.

Operations on a B-Tree

The following operations are performed on a B-Tree.

1. Search
2. Insertion
3. Deletion

Insertion Operation in B-Tree

In a B-Tree, a new element must be added only at the leaf node. That means, the new keyValue is always attached to the leaf node only.

insertion operation

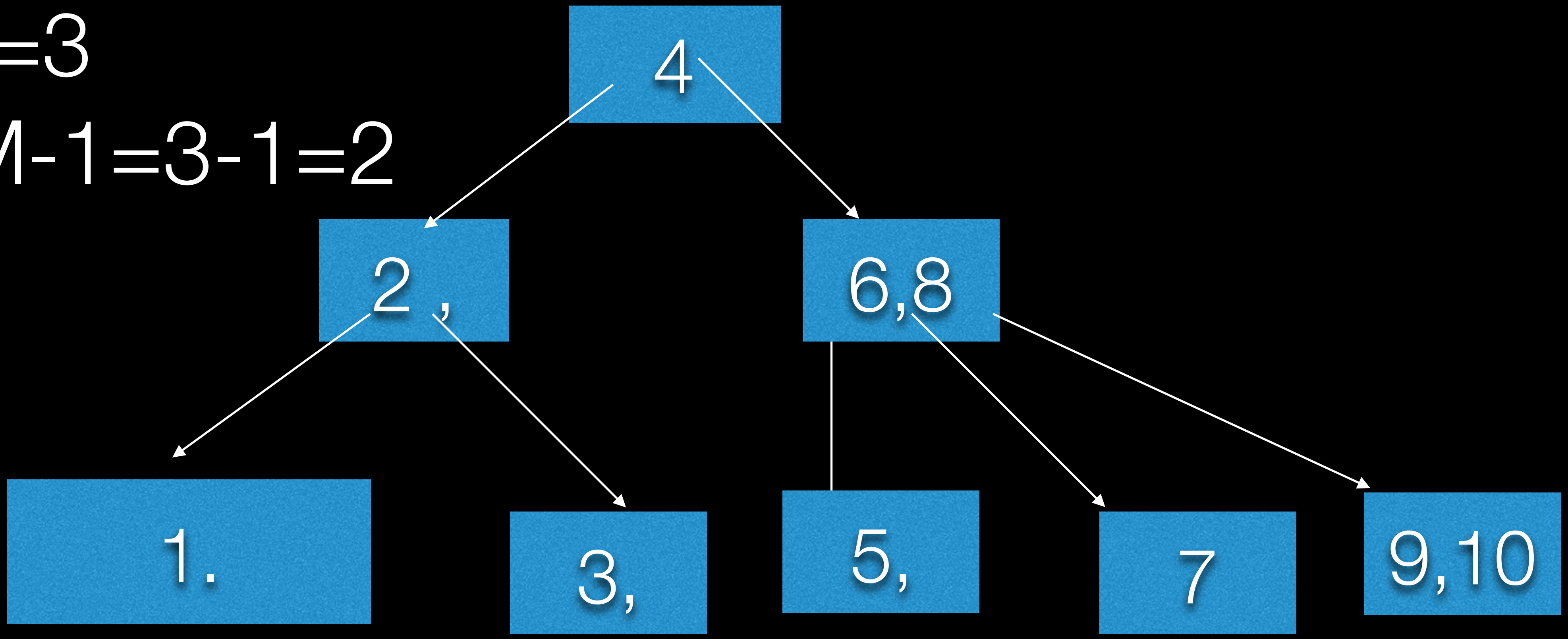
- Step 1 - Check whether tree is Empty.
- Step 2 - If tree is Empty, then create a new node with new key value and insert it into the tree as a root node.
- Step 3 - If tree is Not Empty, then find the suitable leaf node to which the new key value is added using Binary Search Tree logic.
- Step 4 - If that leaf node has empty position, add the new key value to that leaf node in ascending order of key value within the node.
- Step 5 - If that leaf node is already full, split that leaf node by sending middle value to its parent node. Repeat the same until the sending value is fixed into a node.
- Step 6 - If the splitting is performed at root node then the middle value becomes new root node for the tree and the height of the tree is increased by one

Construct a **B-Tree of Order 3** by inserting numbers from 1 to 10.

1,2,3,4,5,6,7,8,9,10

M=3

Max key M-1=3-1=2



Search operation

Step 1 - Read the search element from the user.

Step 2 - Compare the search element with first key value of root node in the tree.

Step 3 - If both are matched, then display "Given node is found!!!”

Step 4 - If both are not matched, then check whether search element is smaller or larger than that key value.

Step 5 - If search element is smaller, then continue the search process in left subtree.

Step 6 - If search element is larger, then compare the search element with next key value in the same node and repeat steps 3, 4, 5 and 6 until we find the exact match or until the search element is compared with last key value in the leaf node.

Step 7 - If the last key value in the leaf node is also not matched then display "Element is not found".