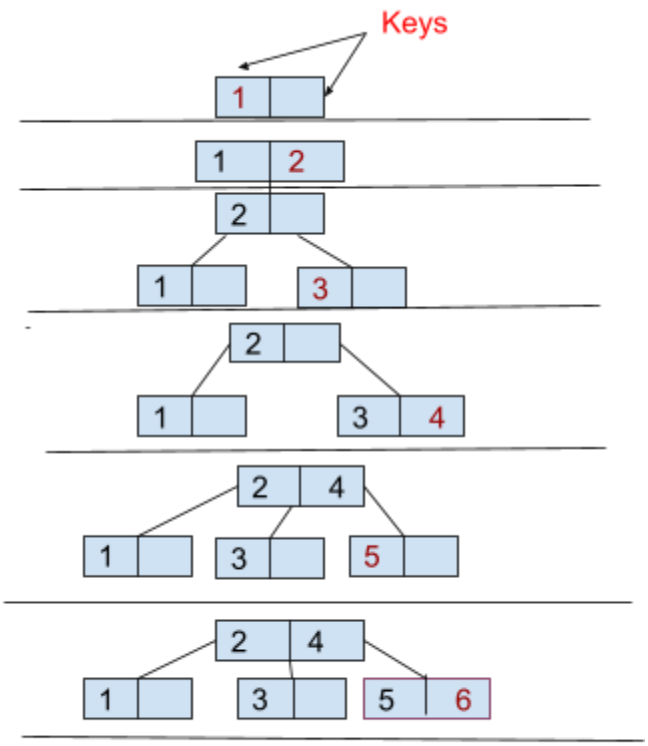
B Tree
(Dynamic multilevel index)

★ B tree is a type of tree in which data is stored in a disk.
★ A generalization of BST, in which a node can have more than one key and more than two children
★ B tree is an m-order tree which stores multiple elements in each node, i.e. every node has maximum ‘m’ children & minimum children in leaf, root and internal node can have 0, 2 & (m/2) respectively.
★ Each node can store at most m-1 elements which are called as keys. Minimum keys in the root node and all other nodes can have 1 and (m/2)-1 respectively.
★ It is a balanced tree.
★ Leaf nodes should be at the same level.
★ It stores the element in ascending order.
★ Each node stores some values
★ Let's have an example, suppose the order is of 3(m) then each node can store 2 (m-1) keys.

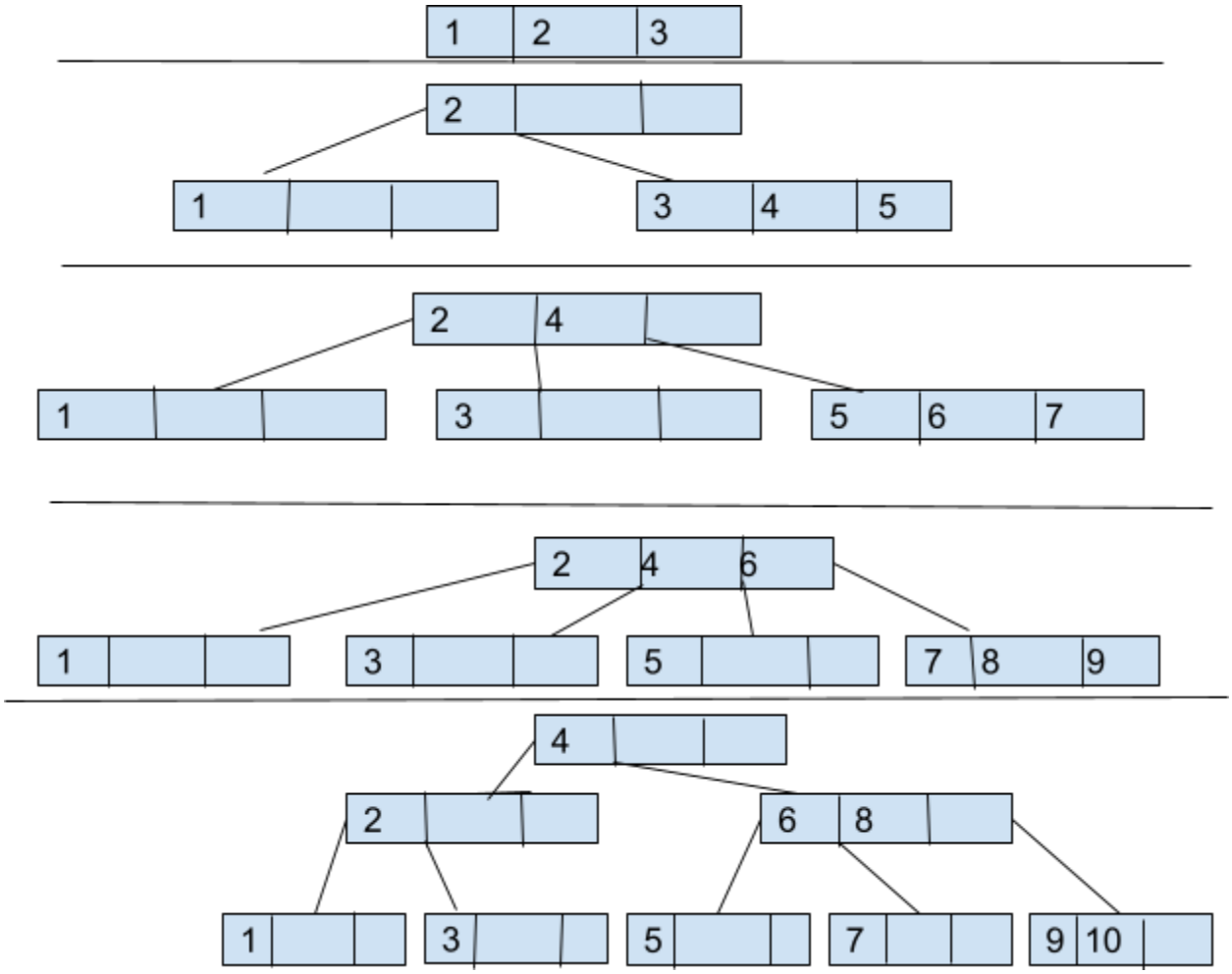


So here a node can have a maximum of 2 values, when the third one will approach, the middle value will go up and store values in the upper key.

INSERTION IN B-TREE

- ➔ Insert the following keys into B-tree, if the order of B-tree is 4(m) and keys 3(m-1).
  1,2,3,4,5,6,7,8,9,10
- ➔ Insertions are performed at leaf node level.
- ➔ Insert the elements in increasing order.
- ➔ Split the nodes into two in the median.
- ➔ Insert the values in their parent node.
- ➔ If the parent node is occupied by its keys, then split the parent node into two.

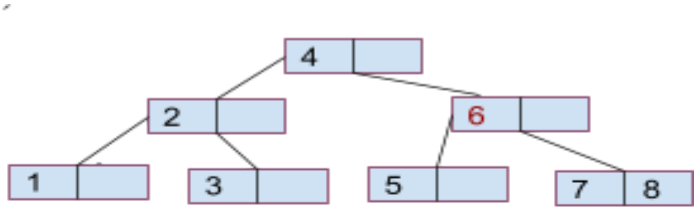
Maximum keys= m-1=3
Minimum keys =(m/2)-1 = 1



SEARCHING

Searching is the same as we do in BST.
If we want to search item 5 in the b tree.

1. Compare item 5 with root node 4. since 5> 4 hence, move to its left sub-tree.
2. Since, 2<5<6, traverse right sub-tree of 40
3. 5>4, move to the right. Compare 5.
4. Match is found.

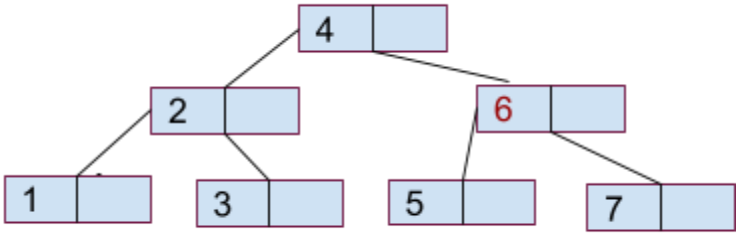


DELETION IN B-TREE

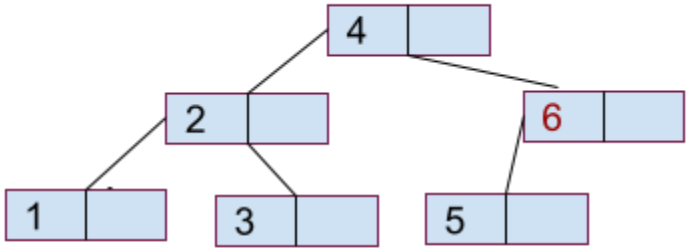
Deletion is also performed at leaf node level.
Nodes to be deleted may be leaf nodes or may be internal nodes.

Example 1

Delete the node 7 from the B Tree of order 3 shown in the following figure.



7 is present in the right child of element 6. Delete it.



Now, 5 is the only element which is left in the child node of 6,leaf nodes should be at the same level where it violates ,elements in its left and right subtree are also not

sufficient therefore, merge it with the left child and intervening element of parent i.e. 6, and then rearrange the above nodes as well.