SNS COLLEGE OF TECHNOLOGY

Coimbatore-35

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE NAME : 23ITT201 DATA STRUCTURES

II YEAR/ III SEMESTER

UNIT – IV MULTIWAY SEARCH TREES AND GRAPH

Topic: *B TREE*

B tree

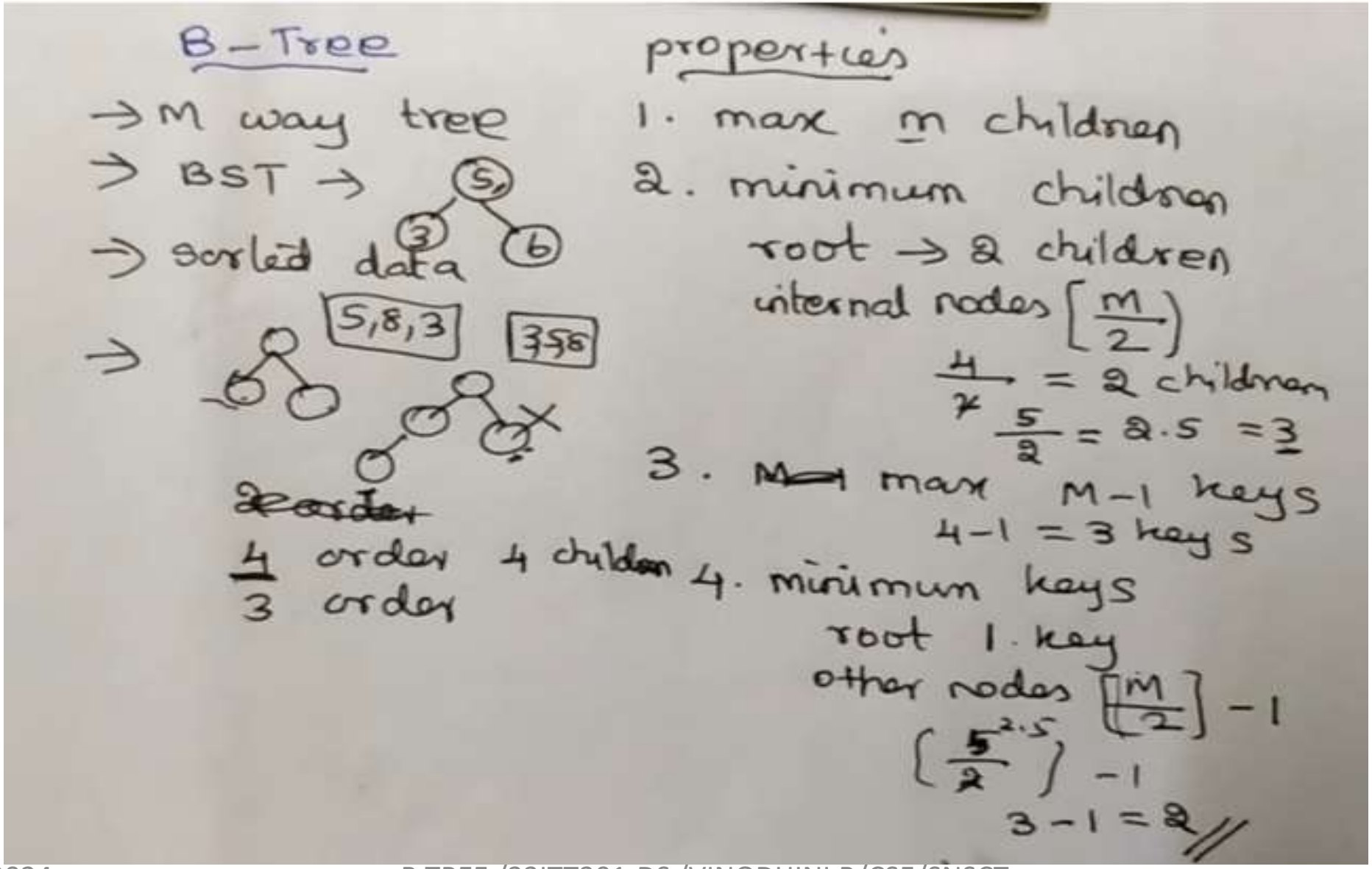
- ***Height Balanced m-way Search Tree***

- B-Tree is a self-balanced search tree with multiple keys in every node and more than two children for every node.
- One of the main reason of using B tree is its capability to store large number of keys in a single node and large key values by keeping the height of the tree relatively small.
- While performing some operations on B Tree, any property of B Tree may violate such as number of minimum children a node can have. To maintain the properties of B Tree, the tree may split or join.
- Operations
  - Searching
  - Insertion
  - Deletion

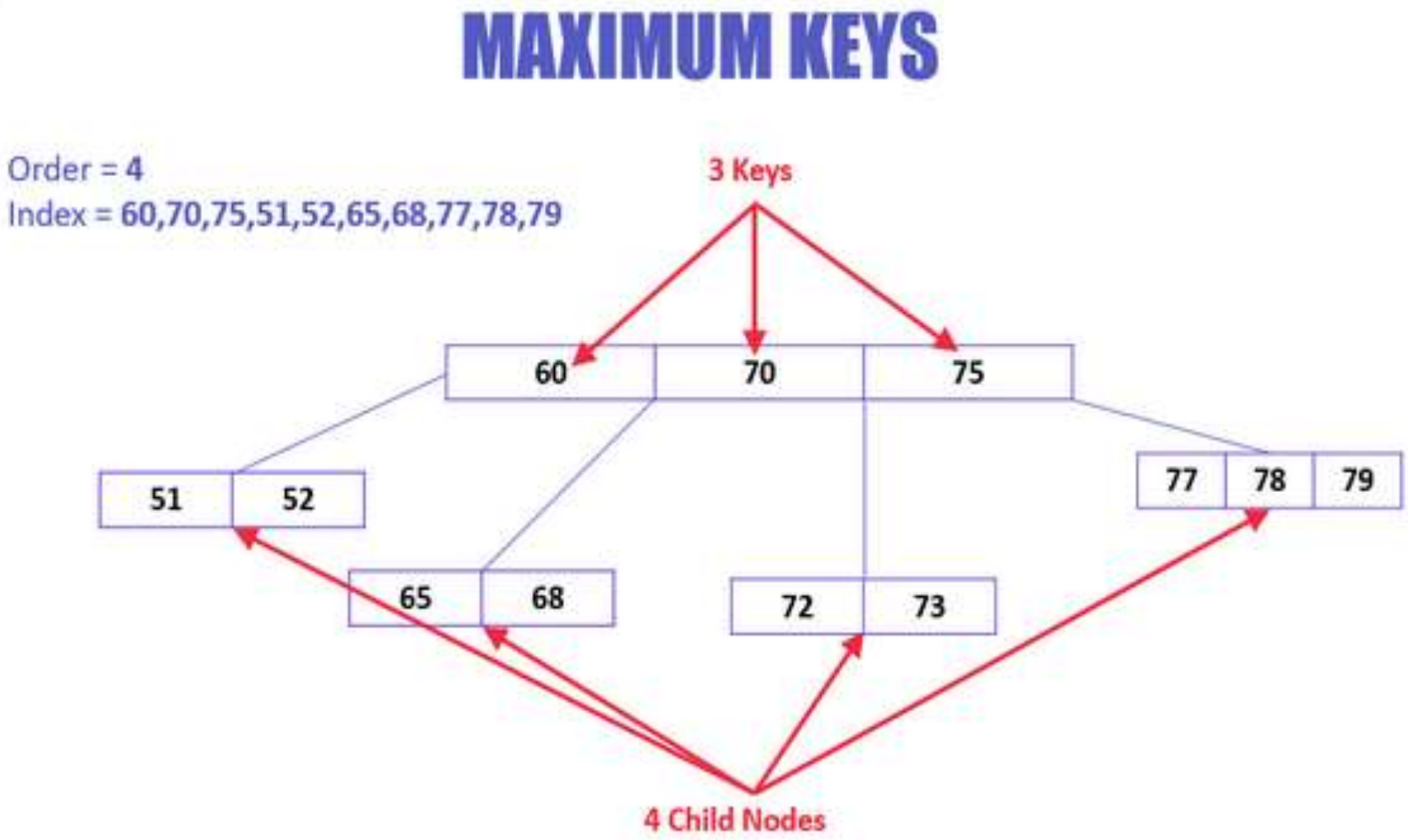
B tree

B-Tree of Order m Properties	
Property #1	All the **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.
Property #3	- All non leaf nodes except root (i.e. all internal nodes) must have at least **m/2** children.
Property #4	If the root node is a non leaf node, then it must have **at least 2** children.
Property #5	A non leaf node with **n-1** keys must have **n** number of children
Property #6	All the **key values within a node** must be in **Ascending Order**

B tree

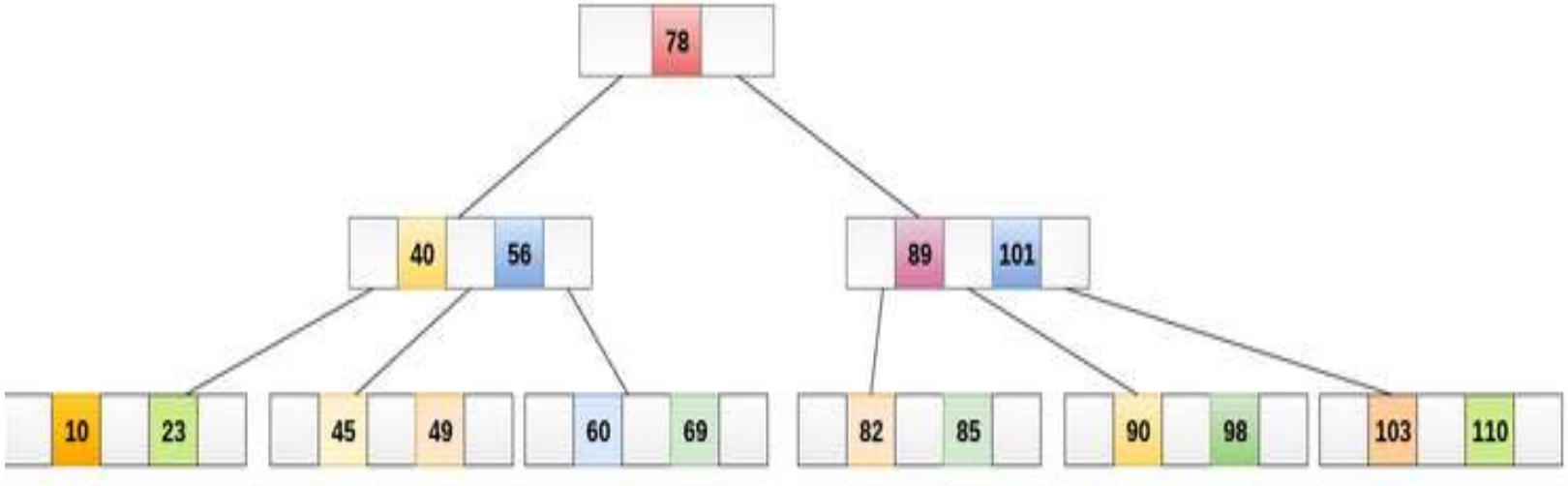

B-Tree
properties
→ M way tree
→ BST →
5
3
6
→ sorted data
→
5,8,3
3,5,6
4 order 4 children
3 order
1. max m children
2. minimum children
root → 2 children
internal nodes ⌈M/2⌉
4/2 = 2 children
5/2 = 2.5 = 3
3. max M-1 keys
4-1 = 3 keys
4. minimum keys
root 1 key
other nodes ⌈M/2⌉ - 1
⌈5/2⌉ - 1
3 - 1 = 2

B tree Example



B tree Searching

Searching in B Trees is similar to that in Binary search tree. For example, if we search for an item 49 in the following B Tree. The process will something like following :
Compare item 49 with root node 78. since 49 < 78 hence, move to its left sub-tree.
Since, 40<49<56, traverse right sub-tree of 40.
49>45, move to right. Compare 49 match found, return.
Searching in a B tree depends upon the height of the tree. The search algorithm takes O(log n) time to search any element in a B tree.



B tree Insertion

Insertions are done at the leaf node level. The following algorithm needs to be followed in order to insert an item into B Tree.

Traverse the B Tree in order to find the appropriate leaf node at which the node can be inserted.

If the leaf node contain less than m-1 keys then insert the element in the increasing order.

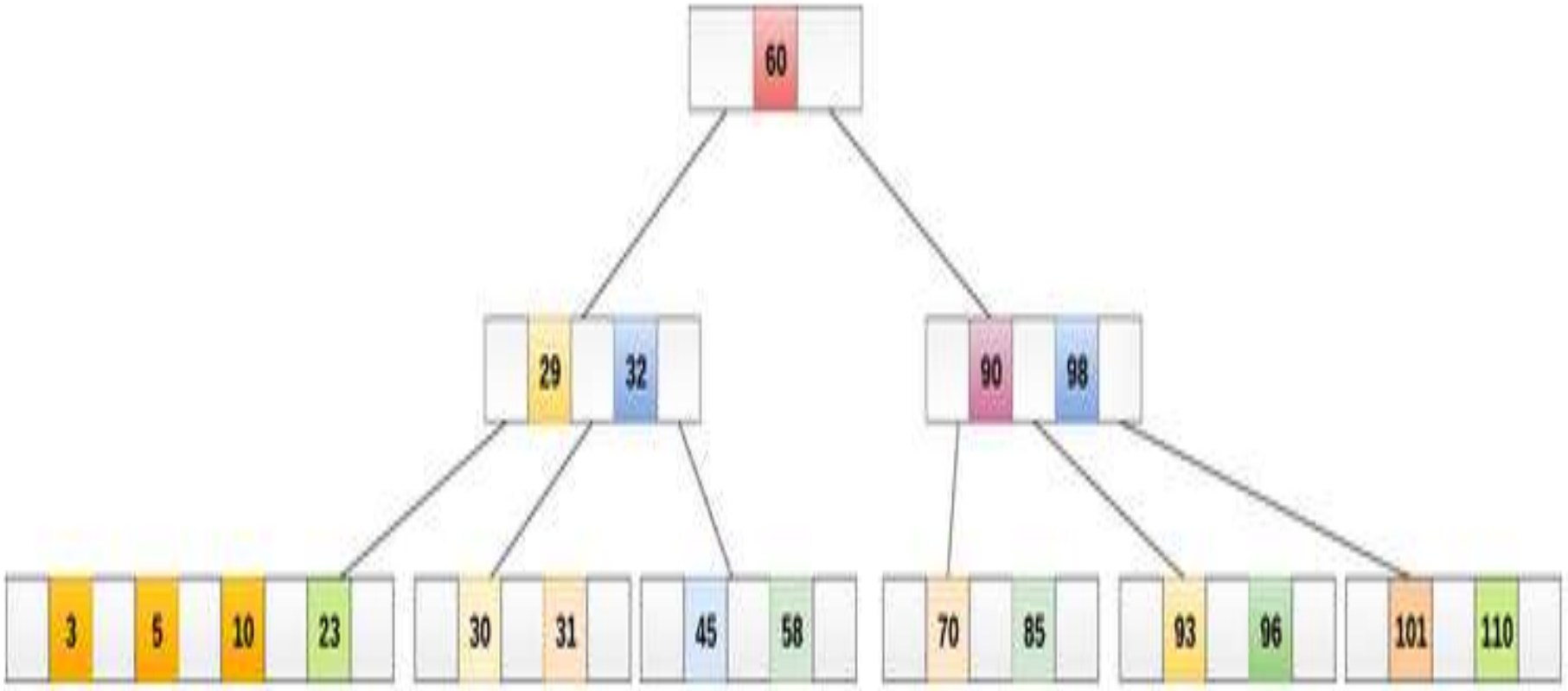
Else, if the leaf node contains m-1 keys, then follow the following steps.

- Insert the new element in the increasing order of elements.
- Split the node into the two nodes at the median.
- Push the median element upto its parent node.
- If the parent node also contain m-1 number of keys, then split it too by following the same steps.

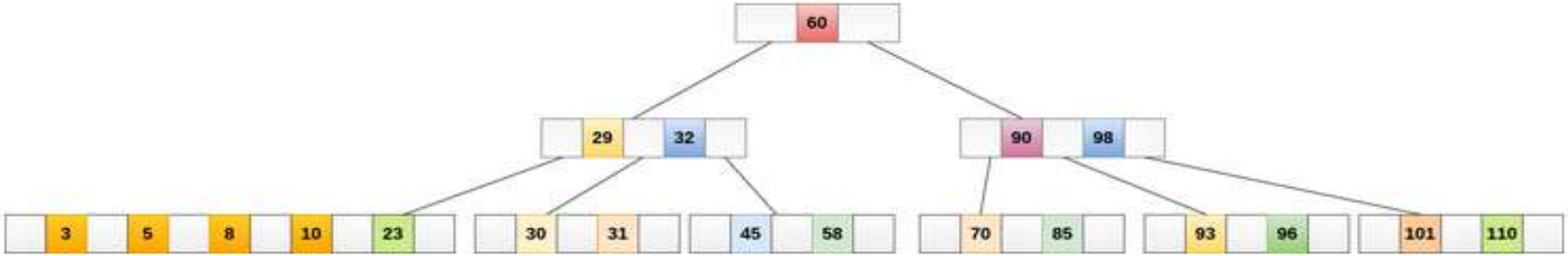
The node has reached the max number of keys
The node will split, and the middle key will become the root node of the rest two nodes.
In case of even number of keys, the middle node will be selected by left bias or right bias.

B tree Insertion

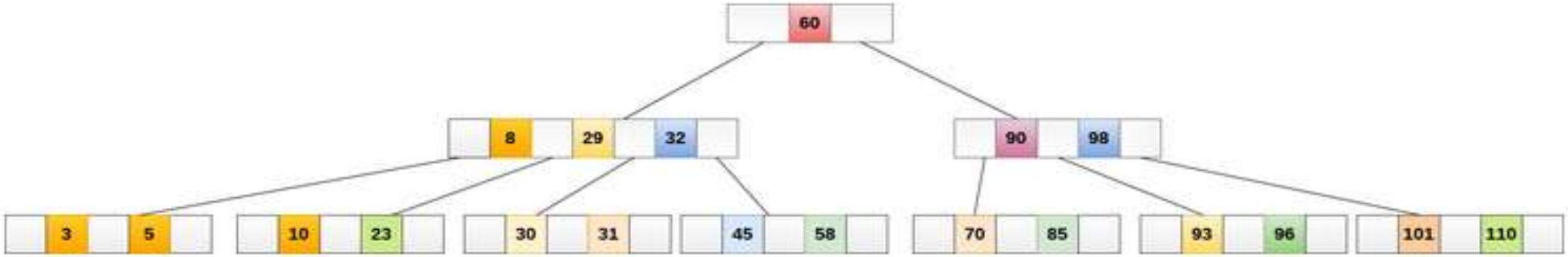
Insert the node 8 into the B Tree of order 5 shown in the following image.



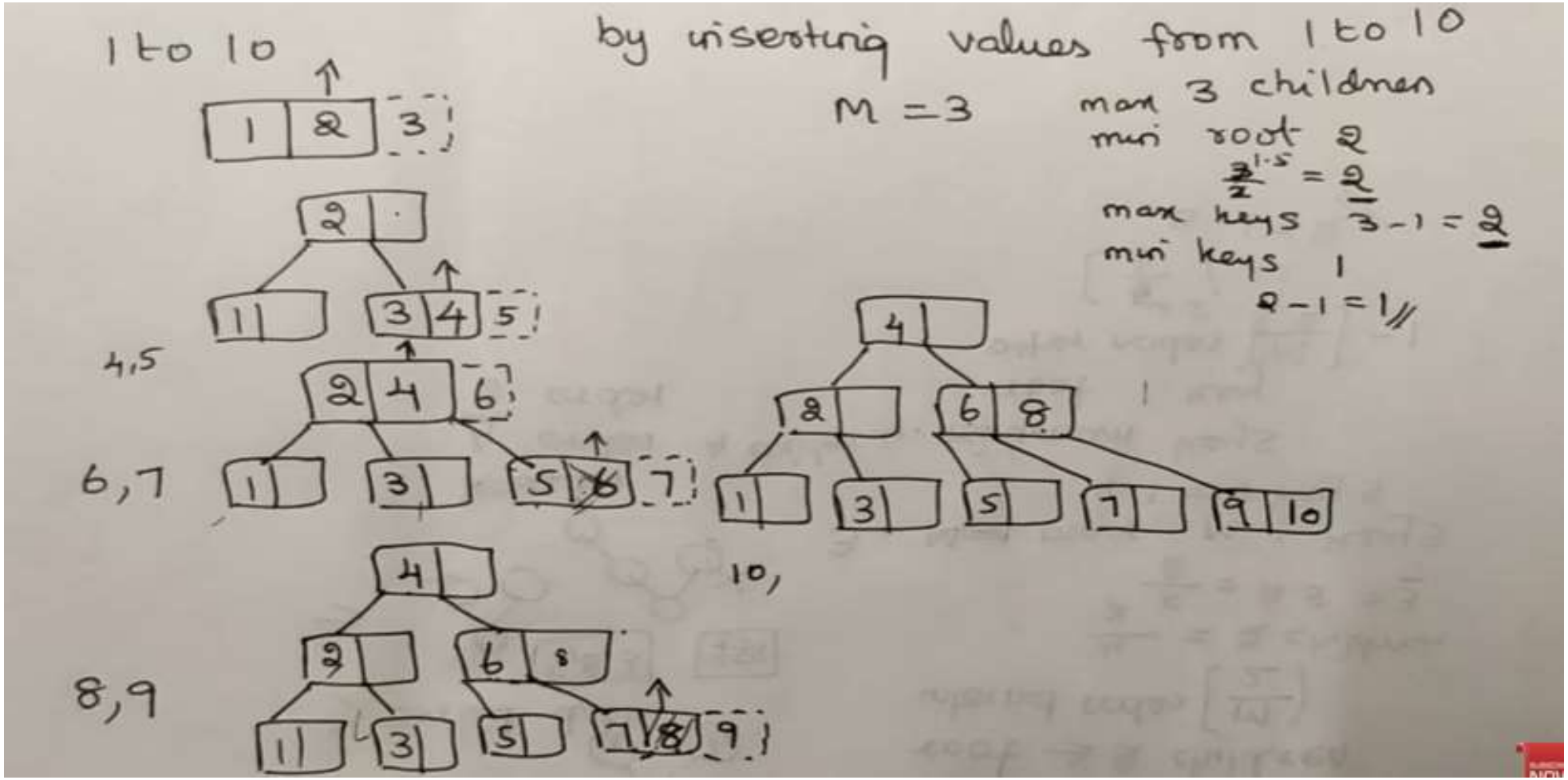
B tree Insertion



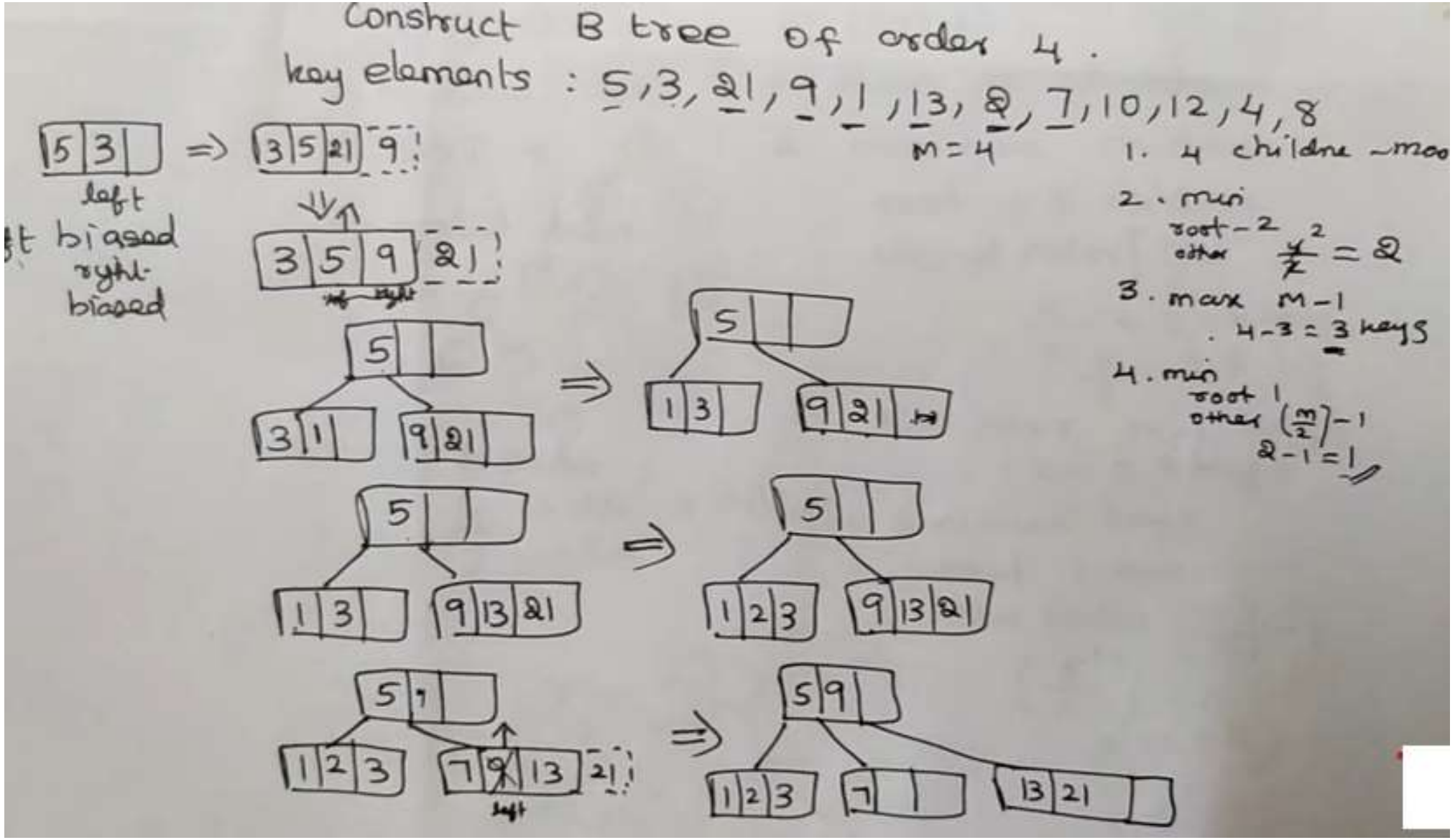
The node, now contain 5 keys which is greater than (5 -1 = 4) keys. Therefore split the node from the median i.e. 8 and push it up to its parent node shown as follows.

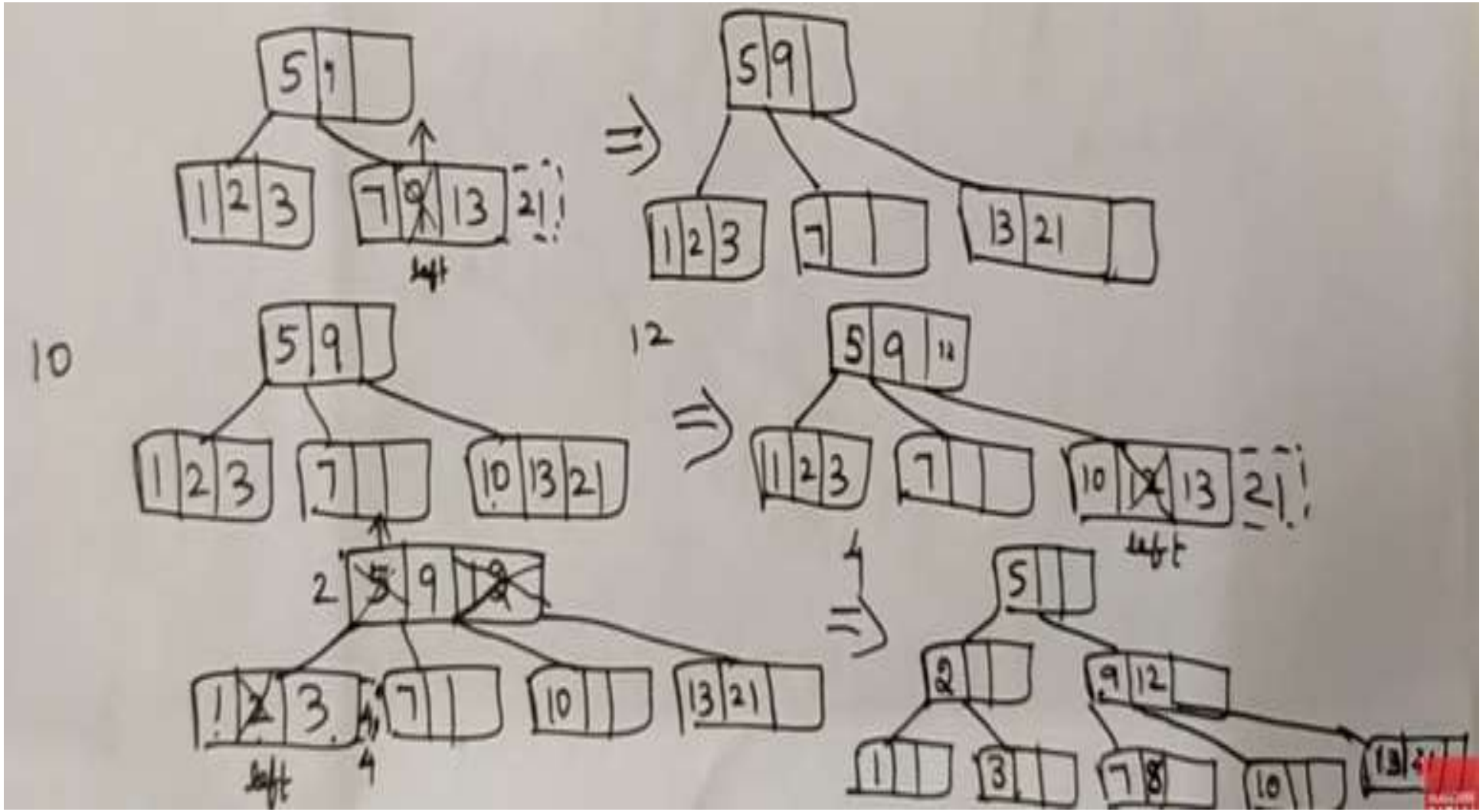


B tree Insertion



B tree Insertion




10
12
left
left

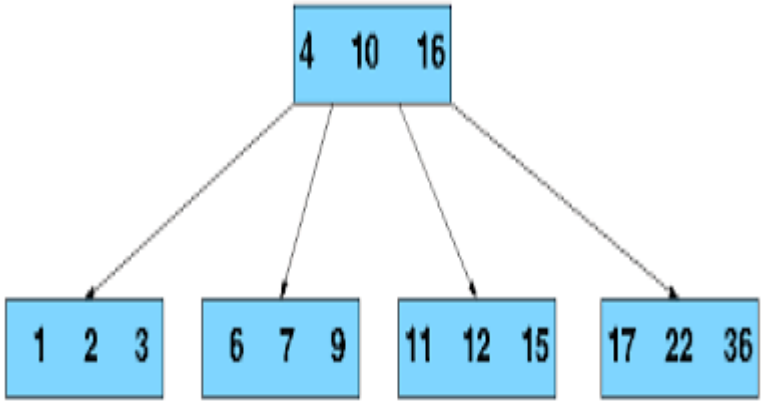
B tree Deletion

Deletion is also performed at the leaf nodes. The node which is to be deleted can either be a leaf node or an internal node. Following algorithm needs to be followed in order to delete a node from a B tree.

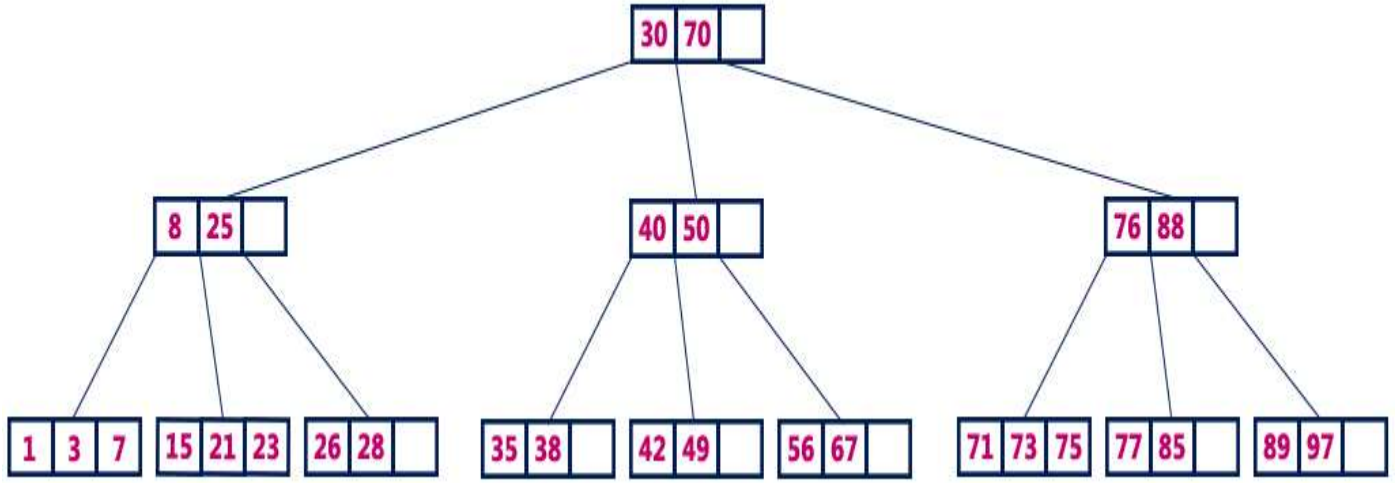
- Locate the leaf node.
- If there are more than m/2 keys in the leaf node then delete the desired key from the node.
- If the leaf node doesn't contain m/2 keys then complete the keys by taking the element from right or left sibling.
    - If the left sibling contains more than m/2 elements then push its largest element up to its parent and move the intervening element down to the node where the key is deleted.
    - If the right sibling contains more than m/2 elements then push its smallest element up to the parent and move intervening element down to the node where the key is deleted.
- If neither of the sibling contain more than m/2 elements then create a new leaf node by joining two leaf nodes and the intervening element of the parent node.
- If parent is left with less than m/2 nodes then, apply the above process on the parent too.

If the the node which is to be deleted is an internal node, then replace the node with its in-order successor or predecessor. Since, successor or predecessor will always be on the leaf node hence, the process will be similar as the node is being deleted from the leaf node.

B tree



B-Tree of Order 4



Assessment

1. Special node in tree structure which has many child nodes and one parent node is called----------

2. A B-tree of order 4 is built from scratch by 10 successive insertions. What is the maximum number of node splitting operations that may take place?

3
4
5
6

3. Search value which is present on each node in search tree diagram of order 'p' is as
p – 1
p – 2
p + 1
p+2

References

1. M. A. Weiss, “Data Structures and Algorithm Analysis in C”, Pearson Education, 2nd Edition, 2002.
2. A. V. Aho, J. E. Hopcroft and J. D. Ullman, “Data Structures and Algorithms”, Pearson Education, 2nd Edition, 2007
3. Ashok Kamthane, " Data Structures Using C ", Pearson Education, 2nd Edition, 2012.
4. Sahni Horowitz, “Fundamentals of Data Structures in C”Universities Press; Second edition 2008

Thank You