Indexing

B+ Tree

B+ Tree Properties

- Every Tree has an order of “n”
- All nodes on the left have smaller values than a key and all nodes on the right have greater than or equal values than a key.

Each node has the following properties	**Max**	**Min**
Children	root/internal: n Leaf: 0	Root: 2 *(if root is not leaf as well)* Internal: ⌈n/2⌉
Keys/Values	n-1	Root: 1 Leaf and Internal: ⌈n/2⌉-1 *[Note: for simplicity leaf and internal is considered same in some resources, otherwise leaf min is ⌈(n-1)/2⌉]*

B+ Tree (Insertion)

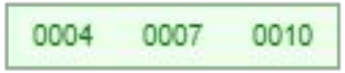
Insertion Algorithm:

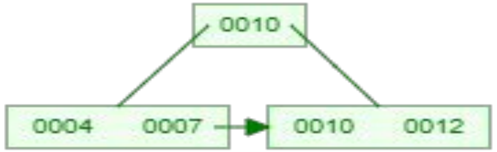
1. **Find the right spot (leaf node):** Start from the top (root) and move down the tree, following the correct path (based on the key you want to insert), until you reach a leaf node.

2. **Insert if node has less than “max” keys:** If that leaf node has space (less than the max number of keys allowed), just insert the value in sorted order.

B+ Tree (Insertion)

3. **Split if node has “max” keys:** If the leaf is already full, split it into half. Put first half on a new left leaf and the second half on a new right leaf. Send up the lowest key from the new right node to the parent



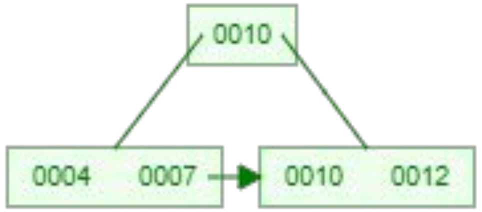


4. **Repeat the split upwards if needed**: If the parent now becomes full too, split it as well and push a key further up. If a non-leaf node is split, the lowest value on the right will only be moved up and not copied to the new right leaf. This might continue up to the root. If the root splits, a new root will be created and the tree height will increase.

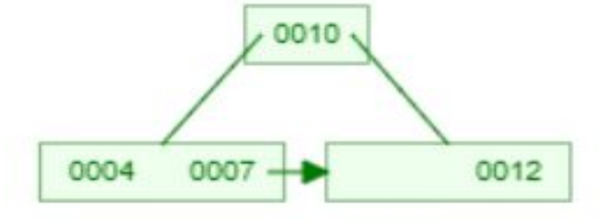
B+ Tree (Insertion)

Difference between splitting a leaf node vs a non-leaf(root/internal) node [for n= 4]

Leaf Node



Non-Leaf Node



B+ Tree (Insertion)

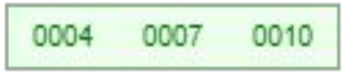
Insert the following in a B+ tree of order 4 according to the given sequence:

4, 10, 7, 12, 1, 5, 2, 25, 15, 9, 18, 20

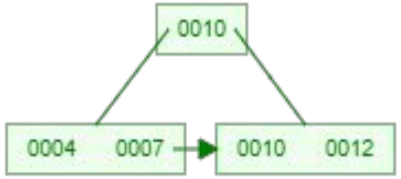
**Use the following B+ tree simulator (https://www.cs.usfca.edu/~galles/visualization/BPlusTree.html)

Here, max keys in a node = 4 - 1 = 3

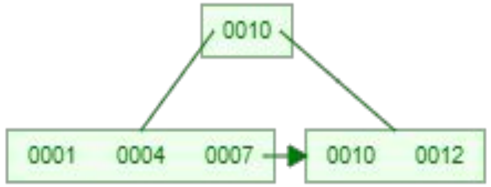
1. Inserting 4, 10, 7



2. Inserting 12

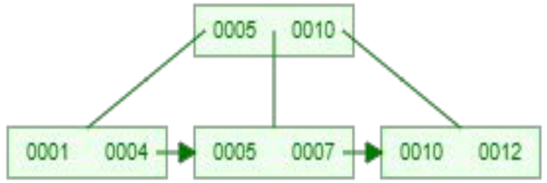


3. Inserting 1

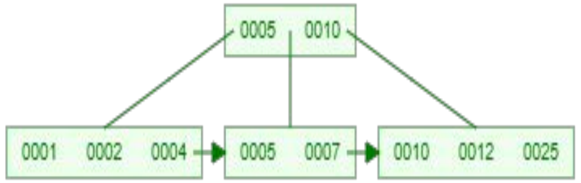


B+ Tree (Insertion)

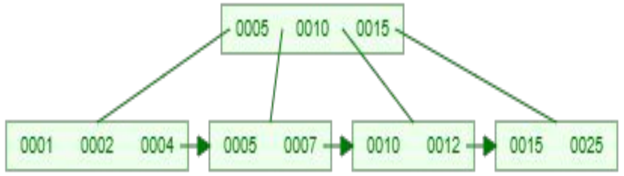
4. Inserting 5



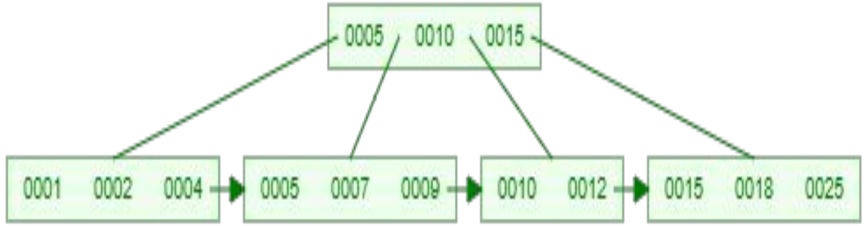
5. Inserting 2, 25



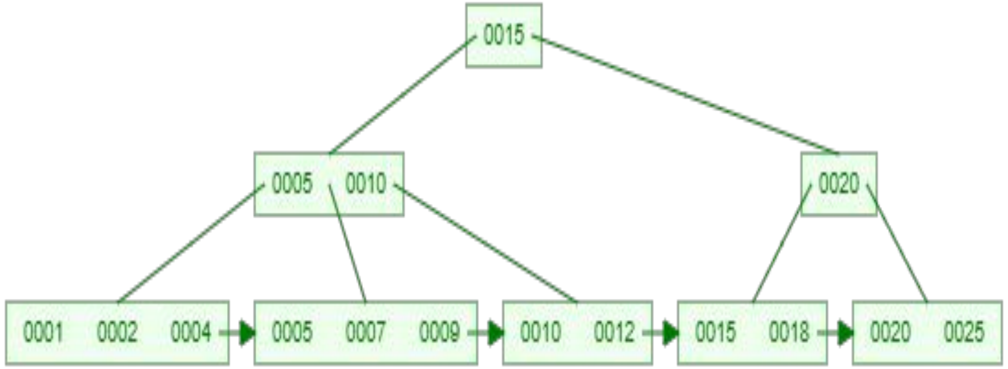
6.Inserting 15



7. Inserting 9, 18



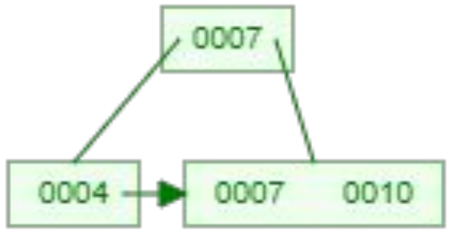
8. Inserting 20

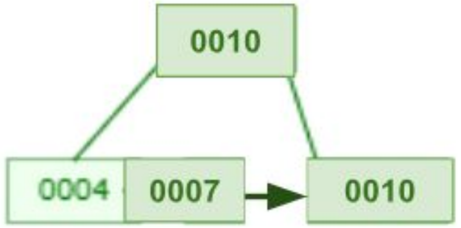


B+ Tree (Insertion)

- If the keys have String values, then sorting should be done according to the “dictionary” order.
- If order of B+ tree is odd (e.g. n=3), then create a right-biased or left-biased tree. Maintain the same bias for the entire tree.

INSERTING 4, 10 , 7





B+ Tree (Deletion)

CASE 1 (No. of Keys > “Min” Keys allowed)

CASE 2 (No. of Keys = “Min” Keys allowed, Sibling Nodes have > “Min” Keys allowed)

CASE 3 (No. of Keys = “Min” Keys allowed, Sibling Nodes also do not have “Min” Keys allowed)

CASE 4 (Internal Nodes: No. of Keys = “Min” Keys allowed)

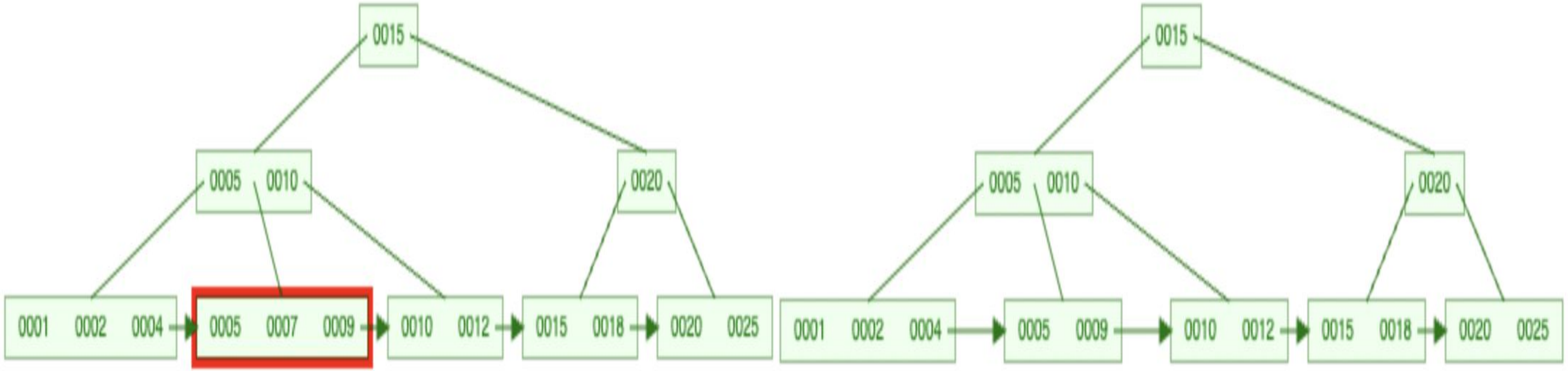
CASE 5 (Shrinking of Tree)

B+ Tree (Deletion)

CASE 1 (No. of Keys > "Min" Keys allowed)

For n = 4, leaf/internal node min = ⌈n/2⌉-1 = ⌈4/2⌉-1 = 1

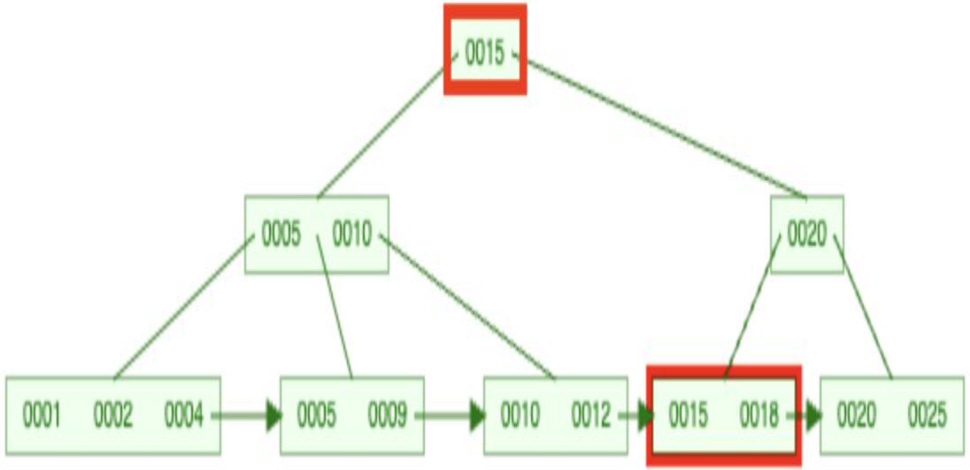
Example 1: Delete 7 -> Find 7 (in leaf node and internal nodes) and remove

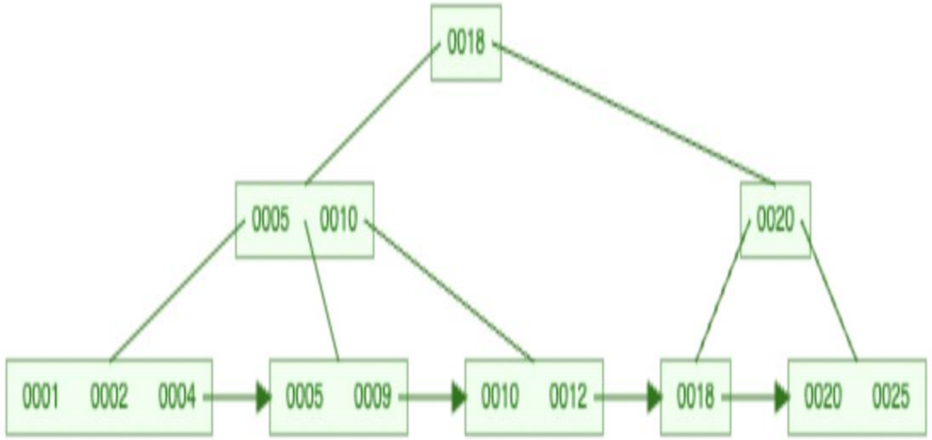


B+ Tree (Deletion)

CASE 1 (No. of Keys > “Min” Keys allowed)

Example 2: Delete 15 -> Find 15 (in leaf node and internal nodes) and remove. If removed from internal nodes, then replace with the lowest key from right subtree.

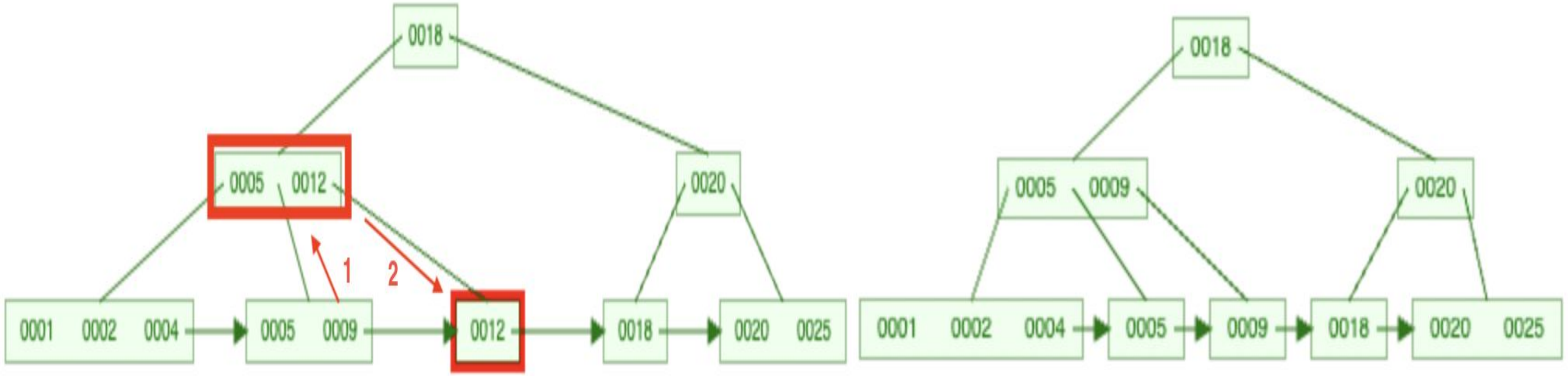




B+ Tree (Deletion)

CASE 2 (No. of Keys = “Min” Keys allowed, Sibling Nodes have > “Min” Keys allowed)

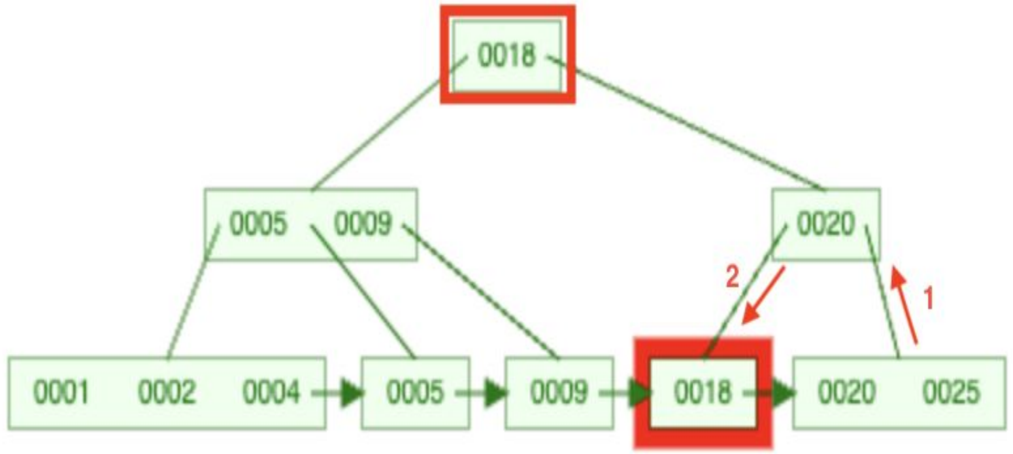
Example 1: Delete 12 -> Find key, **borrow** from left/right sibling through the parent. For internal nodes/parents, copy lowest key from right subtree.

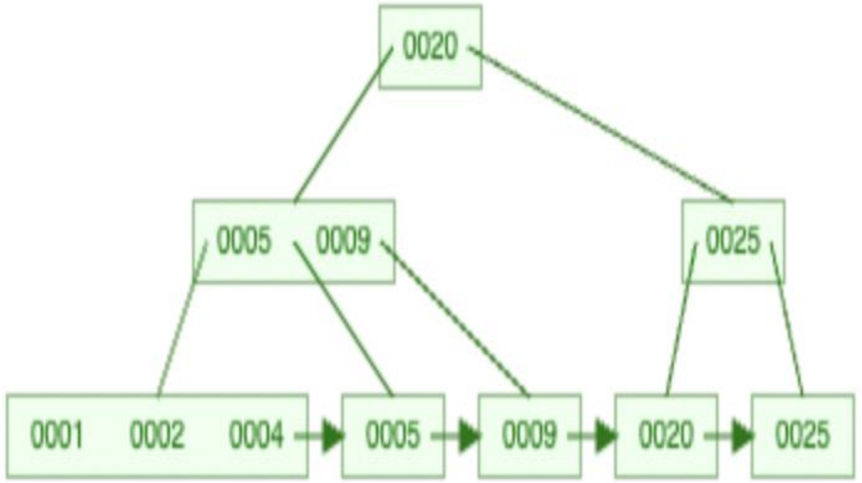


B+ Tree (Deletion)

CASE 2 (No. of Keys = “Min” Keys allowed, Sibling Nodes have > “Min” Keys allowed)

Example 2: Delete 18 -> Find key, **borrow** from left/right sibling through the parent. For internal nodes/parents, copy lowest key from right subtree.

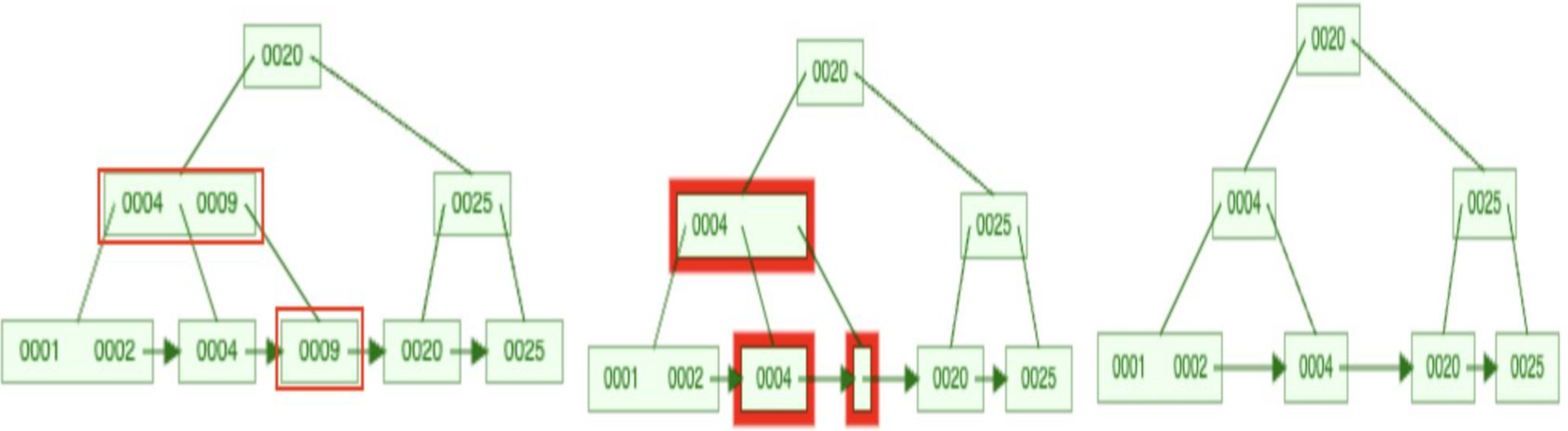




B+ Tree (Deletion)

CASE 3 (No. of Keys = “Min” Keys allowed, Sibling Nodes also have only “Min” Keys allowed, Parent has greater than min allowed)

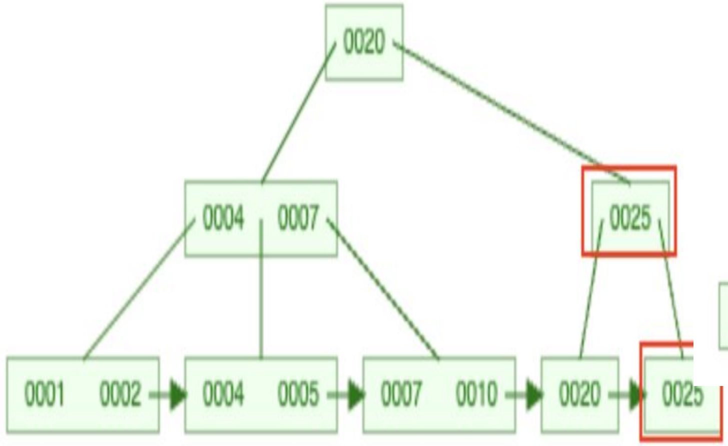
Example 1: Delete 9 -> Find key, **Merge** with sibling, remove parent key (if Parent Node no.of keys> “Min” keys allowed)

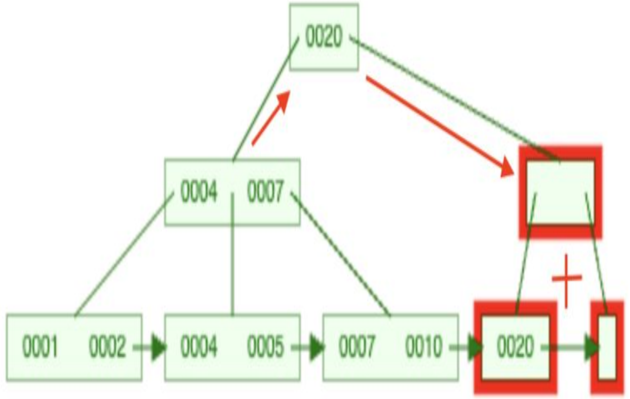


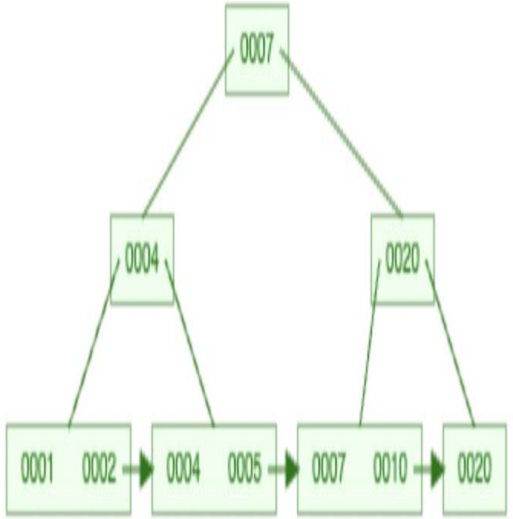
B+ Tree (Deletion)

CASE 4 (Internal Nodes: No. of Keys = “Min” Keys allowed)

Example 1: Delete 25-> Find 20 in leaf and use case 3 since case 2 not possible. Now Parent has less than “min” no. of keys. Parent **borrows from/merges with** sibling and adjust children according to B+ properties. If internal nodes merge, they merge with sibling node and parent key.



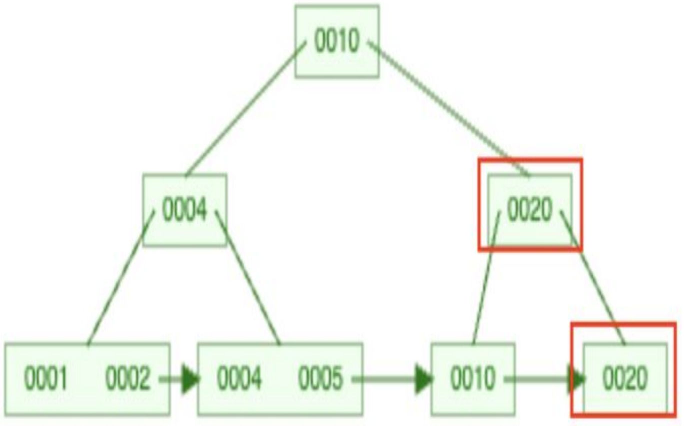


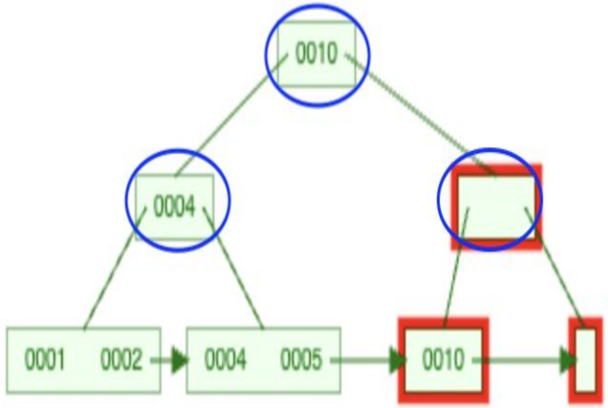


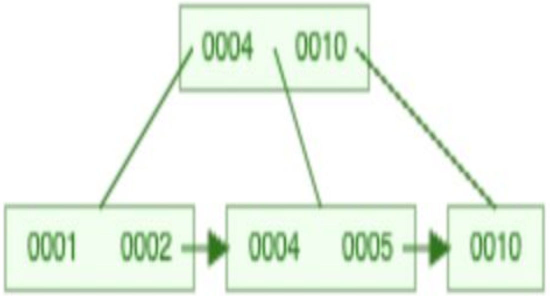
B+ Tree (Deletion)

CASE 5 (Shrinking of Tree: Same conditions as case 4)

Example 1: Delete 20-> Find 20, use case 3 for leaf. For internal nodes, merge with sibling and parent, tree height will shrink if parent node has 1 key. Adjust children according to B+ tree properties.







Summary

CASE 1:

- Node has > min keys

1a -> key is only in leaf

- simply remove the key from leaf

1b -> key is in leaf + internal node

- remove the key from leaf, replace the key in internal node with smallest value from right sub-tree

CASE 2:

- Node has = min keys
- at least 1 sibling has > min keys

Then Borrow from sibling

2a -> borrow from left

- pass the value to parent and the leaf node.

2a -> borrow from right

- pass the value to leaf node, replace parent with smallest value in the right sub-tree

Summary

CASE 3:

- Node has = min keys
- all siblings have = min keys
- parent node has > min keys

Then merge with one sibling, remove the parent key

CASE 4:

- Node has = min keys
- all siblings have = min keys
- parent node has = min keys

Merge with one sibling, remove parent key, then follow 4a/4b depending on case.

4a -> Parent node falls under case 2 conditions

- borrow key from sibling: sibling passes value to it's parent, parent passes it's value to node.

4b -> Parent node falls under case 3 conditions

- merge with sibling node and parent key

CASE 5 (Special Case):

- Exact same condition as Case 4 + Case 4b
- After merging of leaf and parent nodes, height of tree shrinks