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

III YEAR/ III SEMESTER

Topic: *B+ tree*

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

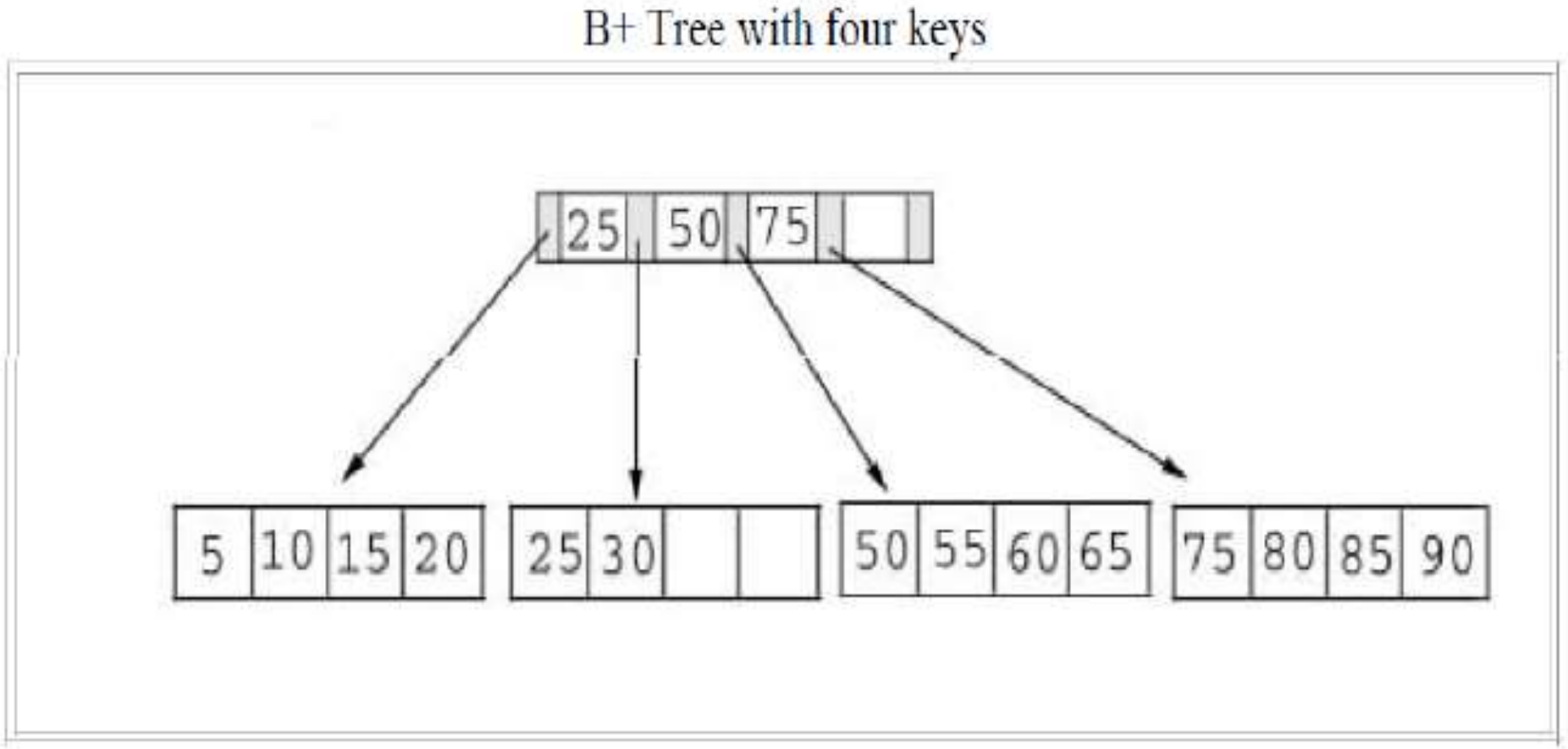
- B+ Tree Combines feature of ISAM(indexed sequential access method) and B Trees
- It contains index pages and data pages . The data pages always appear as leaf nodes in the tree
- The root node and intermediate nodes are always index pages . These features are similar to ISAM.Unlike ISAM, overflow pages are not used in B+ trees

B tree

B+ Trees and B Trees use a "fill factor" to control the growth and the shrinkage. A 50% fill factor would be the minimum for any B+ or B tree. As our example we use the smallest page structure. This means that our B+ tree conforms to the following guidelines.

Number of Keys/page	4
Number of Pointers/page	5
Fill Factor	50%
Minimum Keys in each page	2

B+ tree


B+ Tree with four keys
25
50
75
5
10
15
20
25
30
50
55
60
65
75
80
85
90

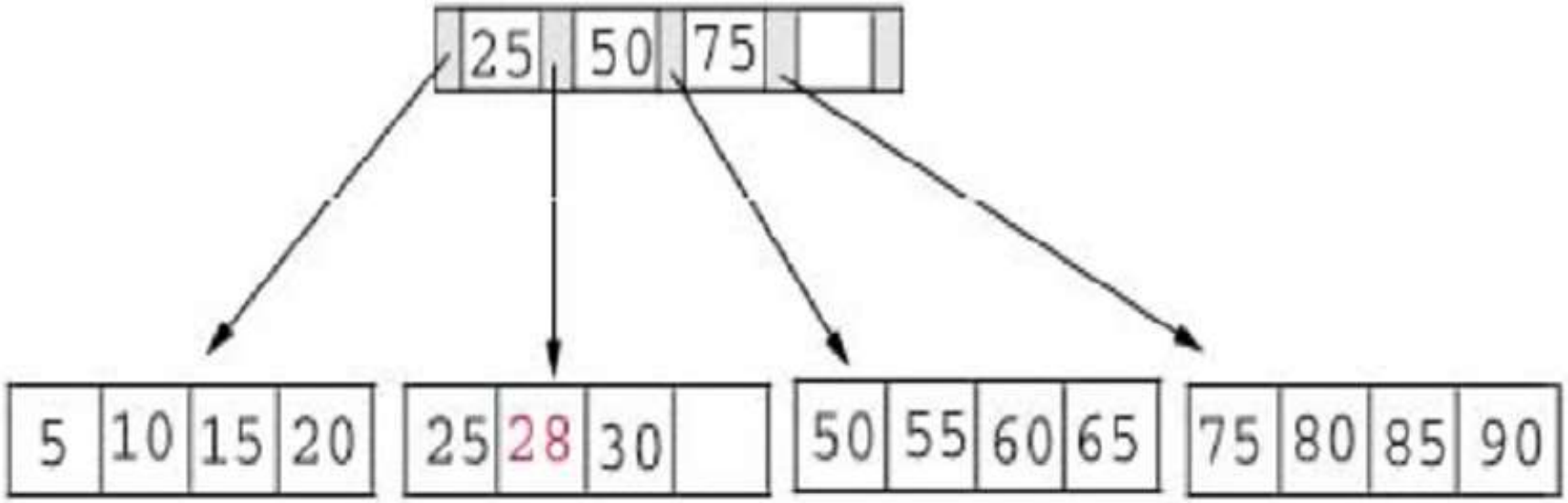
B+ tree

The `insert` algorithm for B+ Trees

Leaf Page Full	Index Page FULL	Action
NO	NO	Place the record in sorted position in the appropriate leaf page
YES	NO	1. Split the leaf page 2. Place Middle Key in the index page in sorted order. 3. Left leaf page contains records with keys below the middle key. 4. Right leaf page contains records with keys equal to or greater than the middle key.
YES	YES	1. Split the leaf page. 2. Records with keys < middle key go to the left leaf page. 3. Records with keys >= middle key go to the right leaf page. 4. Split the index page. 5. Keys < middle key go to the left index page. 6. Keys > middle key go to the right index page. 7. The middle key goes to the next (higher level) index. IF the next level index page is full, continue splitting the index pages.

B+ tree

Add record with key 28



B+ tree

Adding a record when the leaf page is full but the index page is not

Next, we're going to insert a record with a key value of 70 into our B+ tree. This record should go in the leaf page containing 50, 55, 60, and 65. Unfortunately this page is full. This means that we must split the page as follows:

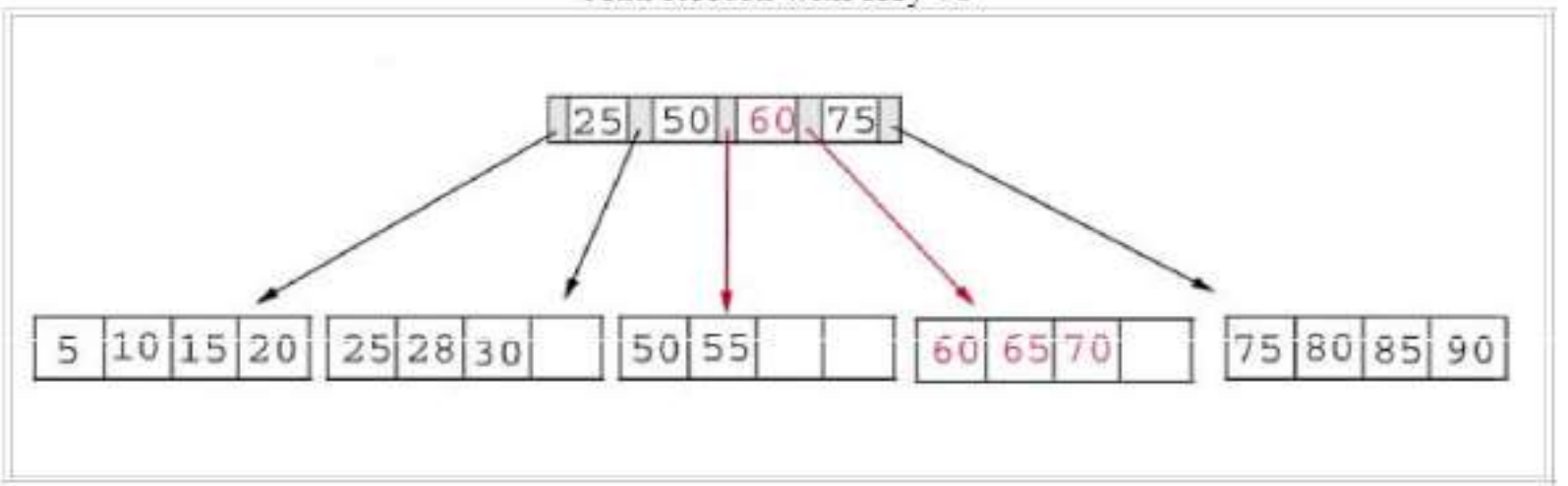
Left Leaf Page	Right Leaf Page
50 55	60 65 70

The middle key of 60 is placed in the index page between 50 and 75.

The following table shows the B+ tree after the addition of 70.

B+ tree

Add Record with Key 70



Adding a record when both the leaf page and the index page are full

As our last example, we're going to add a record containing a key value of 95 to our B+ tree. This record belongs in the page containing 75, 80, 85, and 90. Since this page is full we split it into two pages:

Left Leaf Page	Right Leaf Page

B+ tree

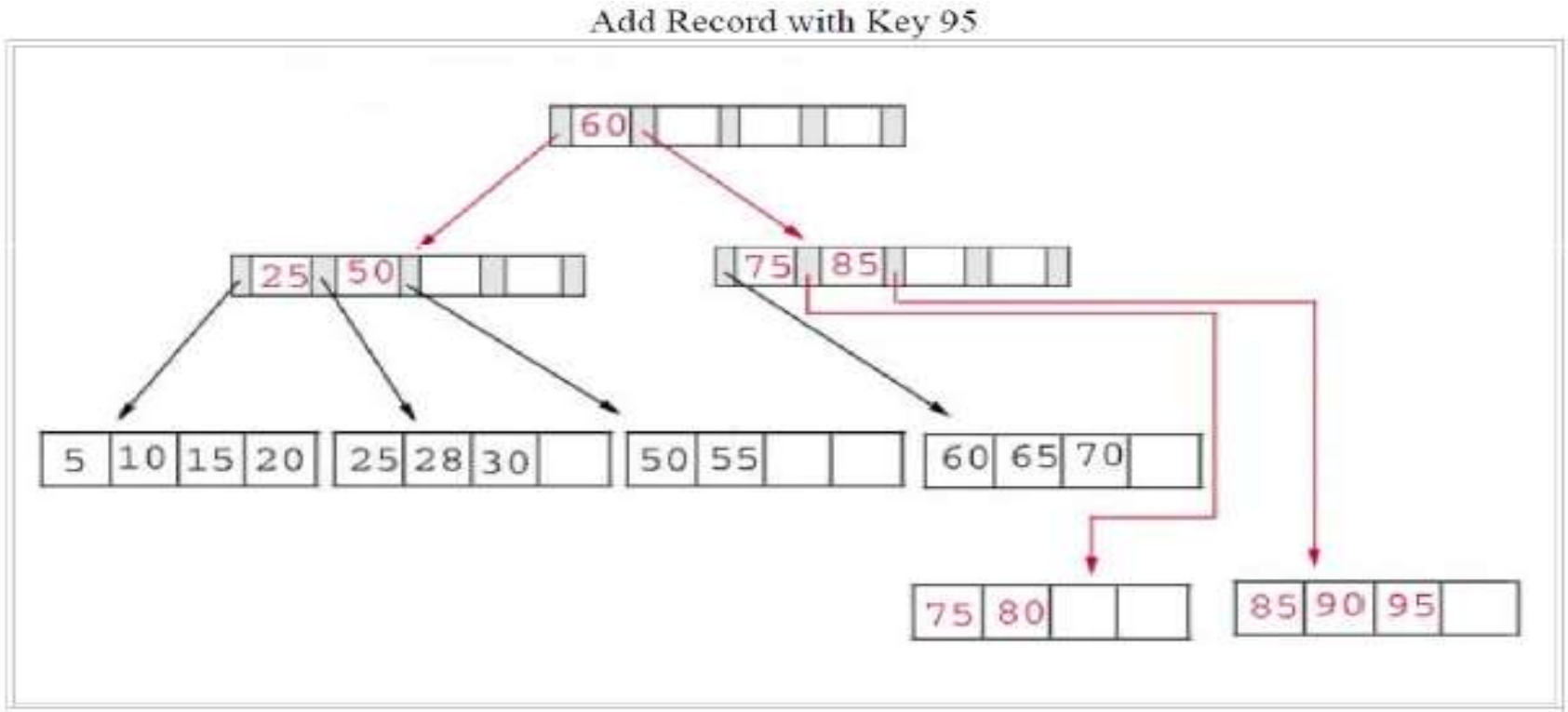
75 80	85 90 95

The middle key ,85,rises to the index page. Unfortunately the index page is also full, so we split the index page

Left Index Page	Right Index Page	New Index Page
25 50	75 85	60

B+ tree

The following figure illustrates the addition of the record containing 95 to the B + Tree



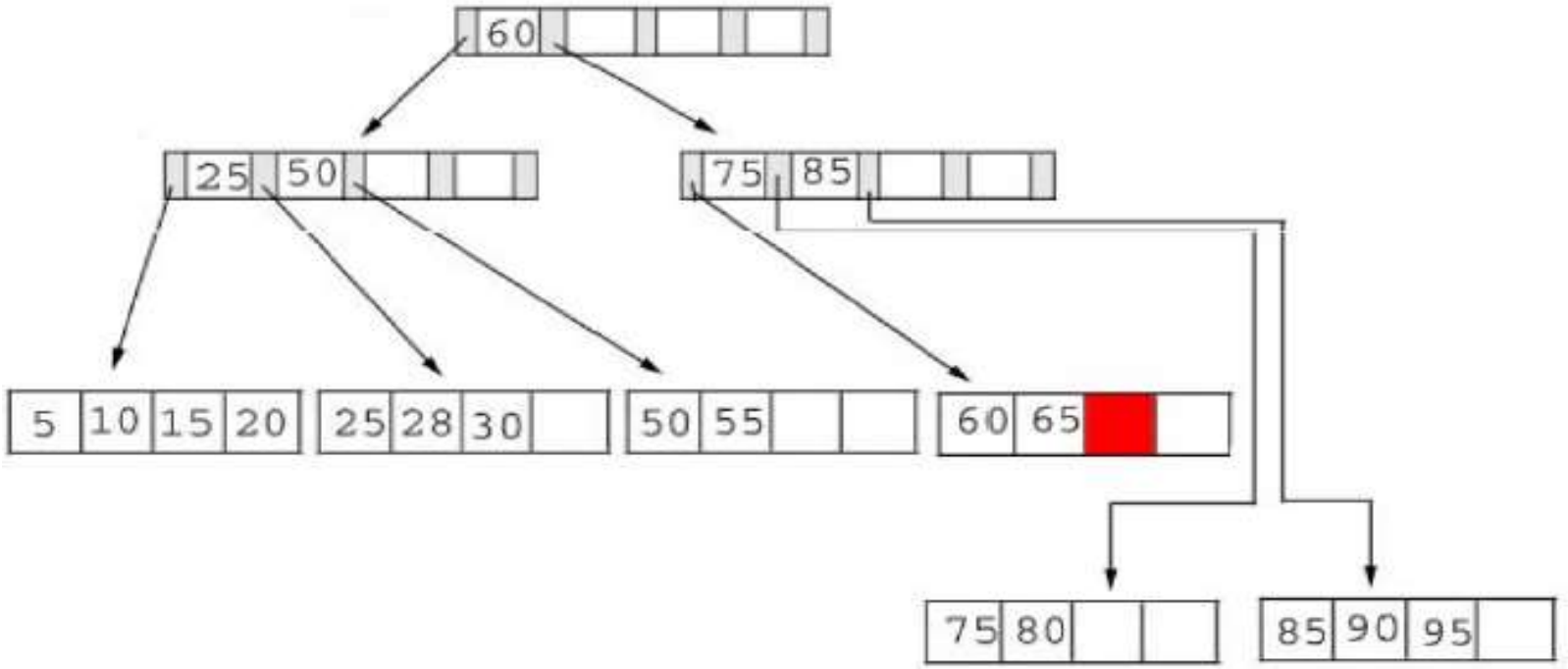
B+ tree

The delete algorithm for the B + Tree

Leaf Page Below Fill Factor	Index Page Below Fill Factor	Action
NO	NO	Delete the record from the leaf page. Arrange keys in ascending order to fill void. If the key of the deleted record appears in the index page, use the next key to replace it.
YES	NO	Combine the leaf page and its sibling. Change the index page to reflect the change.
YES	YES	1. Combine the leaf page and its sibling. 2. Adjust the index page to reflect the change. 3. Combine the index page with its sibling. Continue combining index pages until you reach a page with the correct fill factor or you reach the root page.

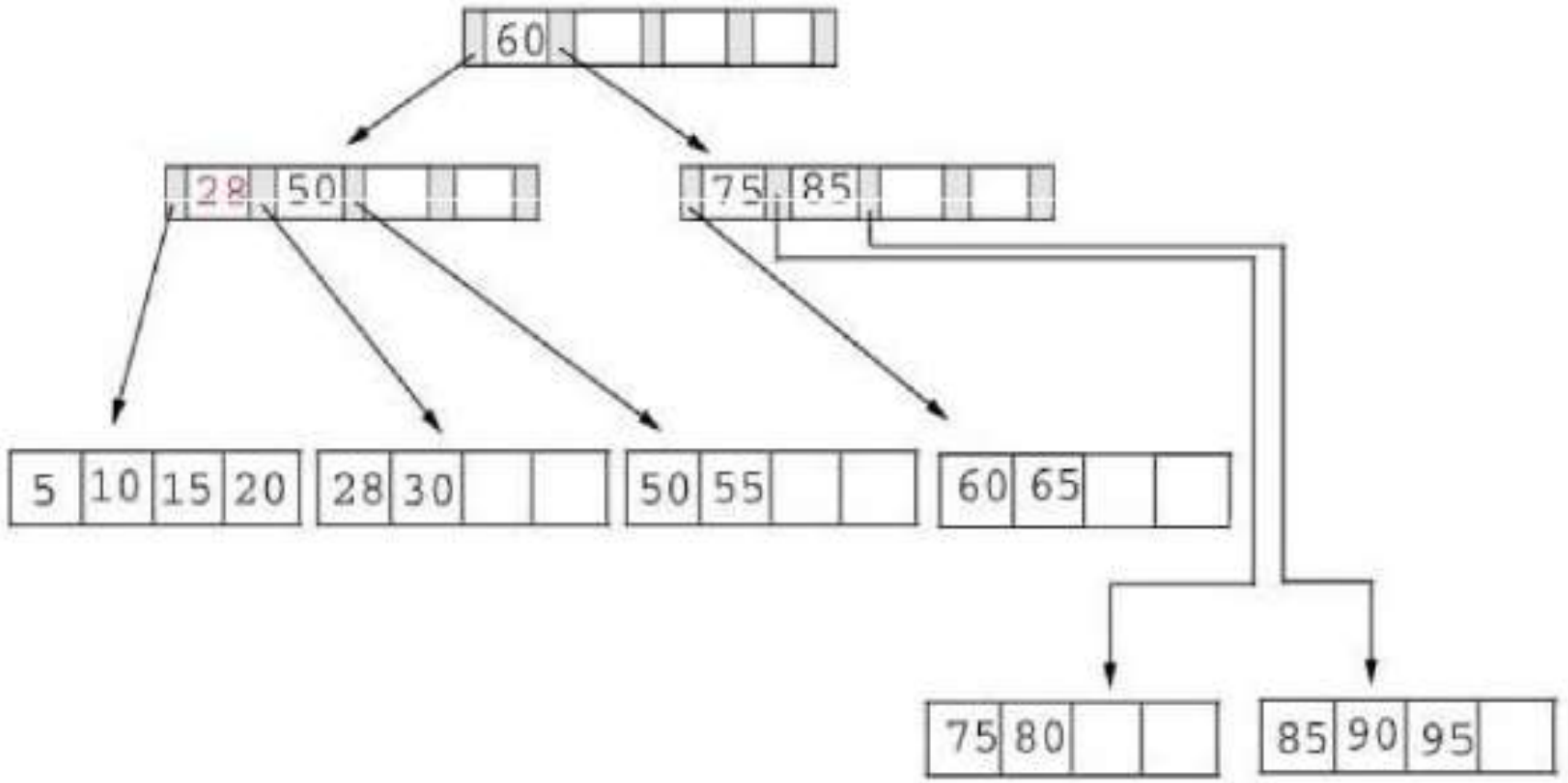
B+ tree

Delete Record with key 70



B+ tree

Delete Record with key 25



Assessment

1. Consider a B+-tree in which the maximum number of keys in a node is 5.
What is the minimum number of keys in any non-root node?

A. 1
B. 2
C. 3
D. 4

2. B+ trees are preferred to binary trees in databases because

A. Disk capacities are greater than memory capacities
B. Disk access is much slower than memory access
C. Disk data transfer rates are much less than memory data transfer rates
D. Disks are more reliable than memory

3. ---------- are the leaf nodes in a B+ tree

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