DATA STRUCTURES AND ITS APPLICATIONS

Graphs

Saritha
Department of Computer Science & Engineering

DATA STRUCTURES AND ITS APPLICATIONS

B-trees

Saritha

Department of Computer Science & Engineering

Case study: B-tree

Indexing:

- Indexing is a technique used in data structure to access the records quickly from the database file.
- An Index is a small table containing two columns one for primary key and the second column.

B-tree

- A B-tree is a tree data structure that keeps data in its node in sorted order and performs the operations like searching, insertions, and deletions in logarithmic amortized time.
- B-tree is an m-way search tree in which each node, with the possible exception of the root, is at-least half full.

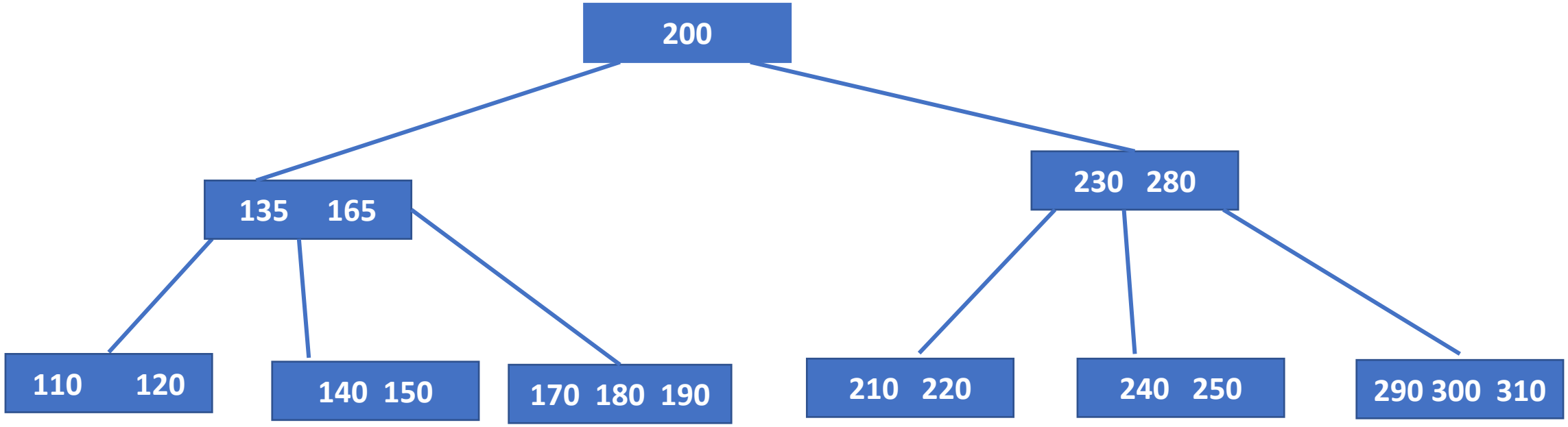
Properties of B-tree

- All the leaves created are at same level.
- B-Tree is determined by a number of degree m. The value of m depends upon the block size on the disk .
- The left subtree of the node has lesser values than the right side of the subtree.
- The maximum number of child nodes(a root node as well as its child nodes can contain) is calculated by :m – 1
- Every node except the root node must contain minimum keys of[m/2]-1
- The maximum number of child nodes a node can have is equal to its degree that is m.
- The minimum children a node can have is half of the order that is m/2

DATA STRUCTURES AND ITS APPLICATIONS

Case study: B-tree

Following is an example of B-Tree of minimum order 5



Case study: B-tree

Why use B-Tree

- B-tree reduces the number of reads made on the disk
- B Trees can be easily optimized to adjust its size according to the disk size
- It is a specially designed technique for handling a bulky amount of data.
- It is a useful algorithm for databases and file systems.
- B-tree is efficient for reading and writing from large blocks of data

Search

Let Key be the element to be searched .
Start from the root and recursively traverse down.
If the value of key is lesser than the root value,
then search left subtree,
if the value of key is greater than the root value,
then search the right subtree.
If the node has the found key,
then return the node.
If the key is not found in the node,
then traverse down to the child with a greater key.
If key is not found in the tree,
then return NULL.

Insertion

- B-tree insertion takes place at leaf node.
- Locate the leaf node for the data being inserted.
- If the node is not full that is fewer than m-1 entries, the new data is simply inserted in the sequence of node.
- When the leaf node is full, we say overflow condition.
- Overflow requires that the leaf node be split into 2 nodes, each containing half of the data.
- To split the node, create a new node and copy the data from the end of the full node to the new node.
- After the data has been split, the new entry is inserted into either the original or the new node depending on its key value.
- Then the median data entry is inserted into parent node.

Insertion

Consider a sequence of integers and minimum degree ‘t’ of the tree is 3. The maximum number of keys the node can hold is 2t-1=5

Initially root=NULL

Insert 1:

Insert 2:

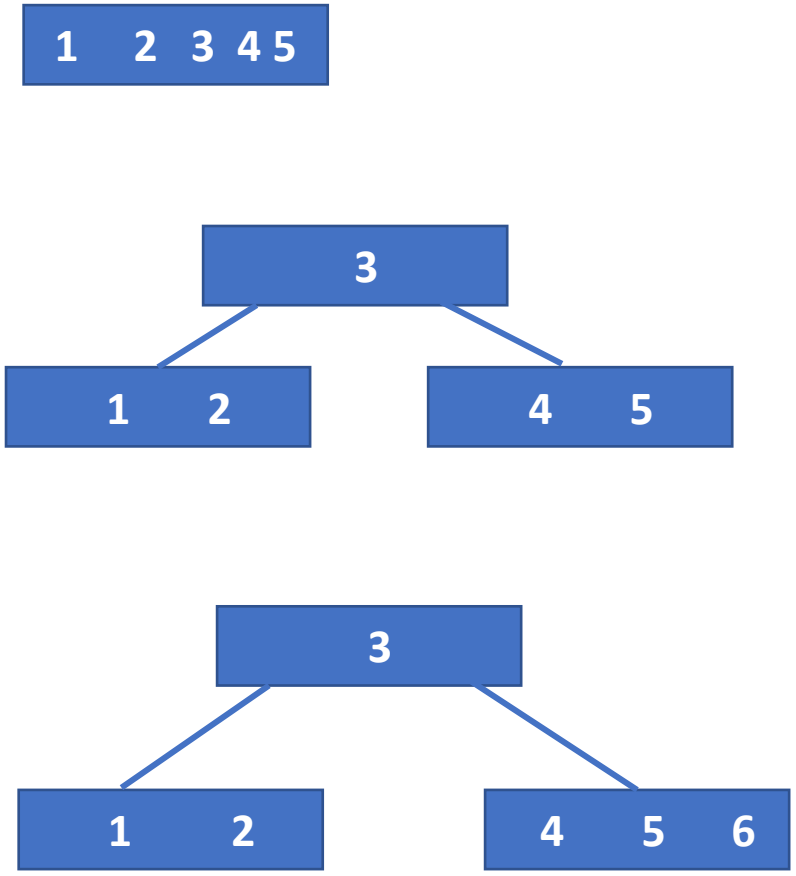
Insert 3:

Insert 4:

Insert 5:

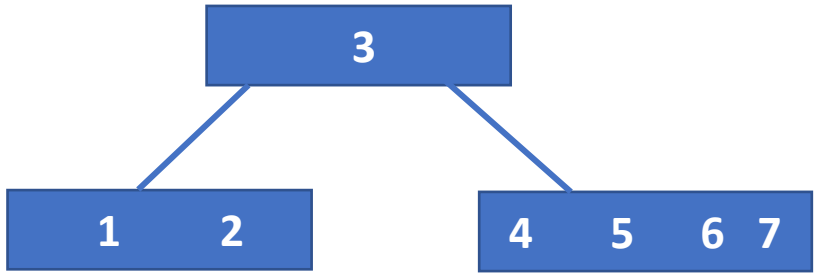
Properties of B-tree

Insert 6:

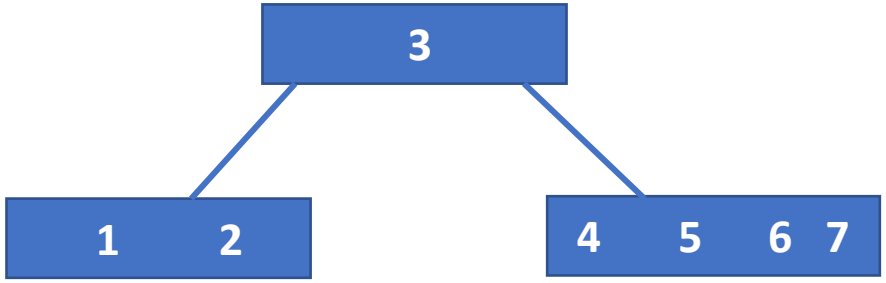


Properties of B-tree

Insert 7:

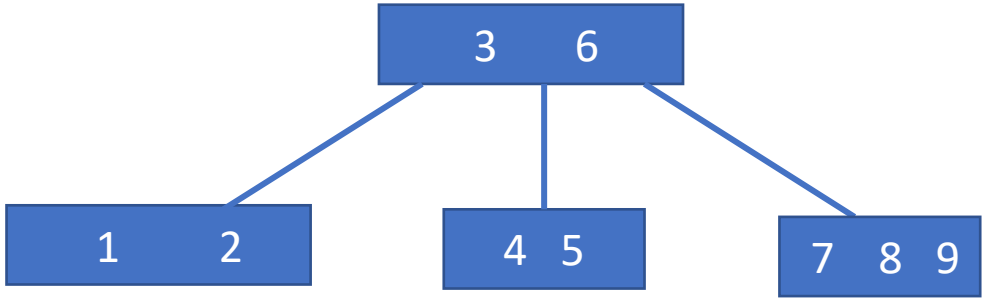


Insert 8:



Properties of B-tree

Insert 9:



Insertion Algorithm

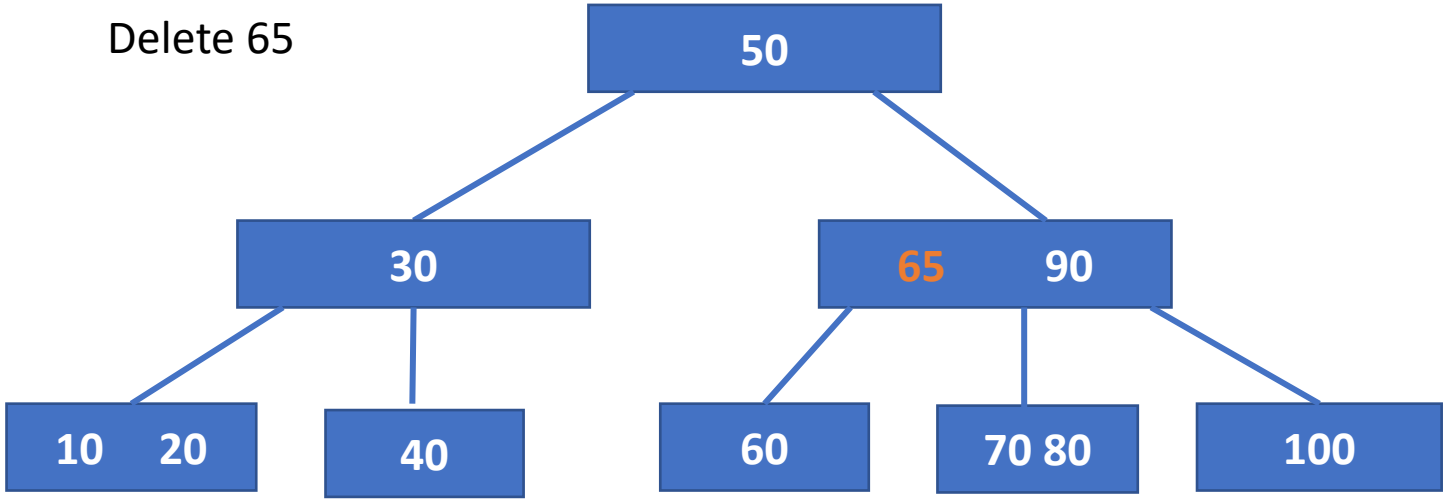
1.Traverse the B Tree in order to find the appropriate leaf node at which the node can be inserted.
2.If the leaf node contain less than m-1 keys then insert the element in the increasing order.
3.Else, if the leaf node contains m-1 keys, then follow the following steps.
    a)Insert the new element in the increasing order of elements.
    b)Split the node into the two nodes at the median.
    c)Push the median element upto its parent node.
    d)If the parent node also contain m-1 number of keys, then split it too by following the same steps.

B-tree Deletion

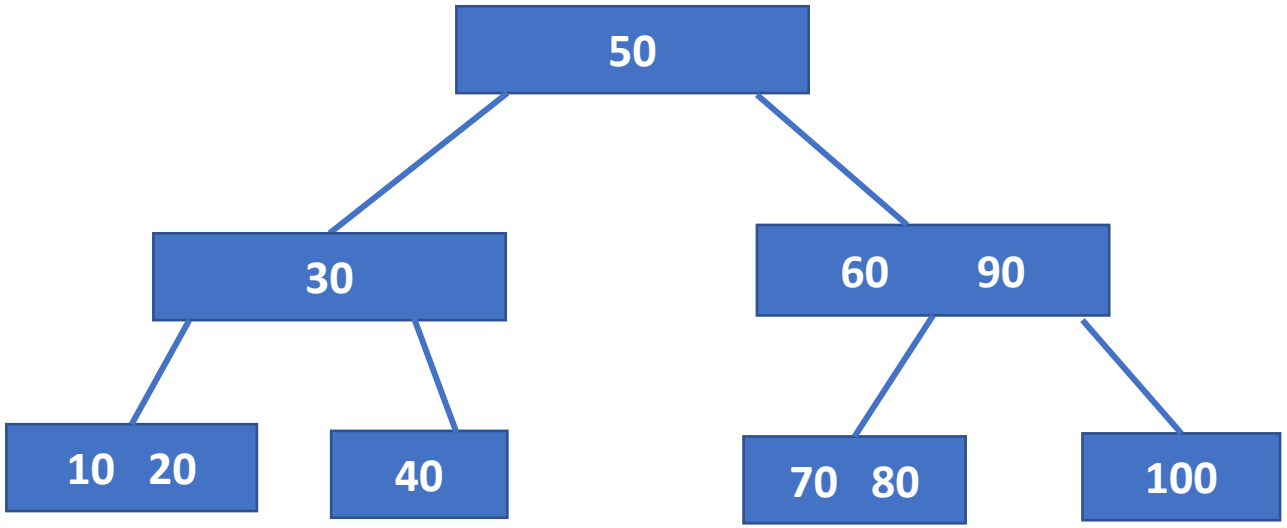
- When deleting a node from B-tree , there are three considerations
- 1. data to be deleted are actually present in the tree.
- 2. if the node does not have enough entries after the deletion, then correct the structural deficiency.
- A deletion that results in a node with fewer than minimum number of entries is an underflow
- 3. Deletion should takes place only from leaf node.
- If the data to be deleted are in internal node, find a data entry to take their place.

Deletion

Delete 65

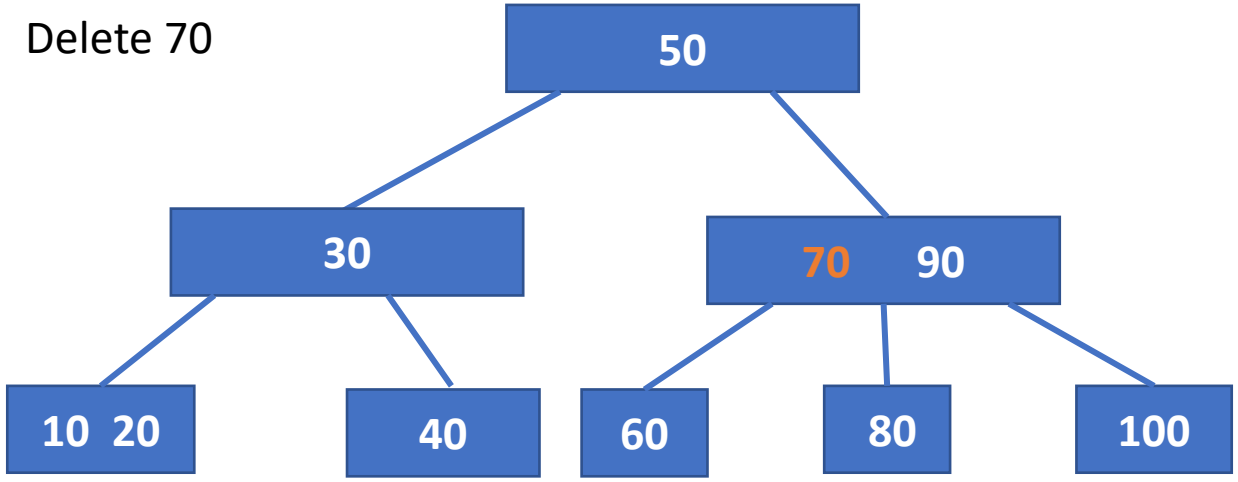


Deletion Algorithm

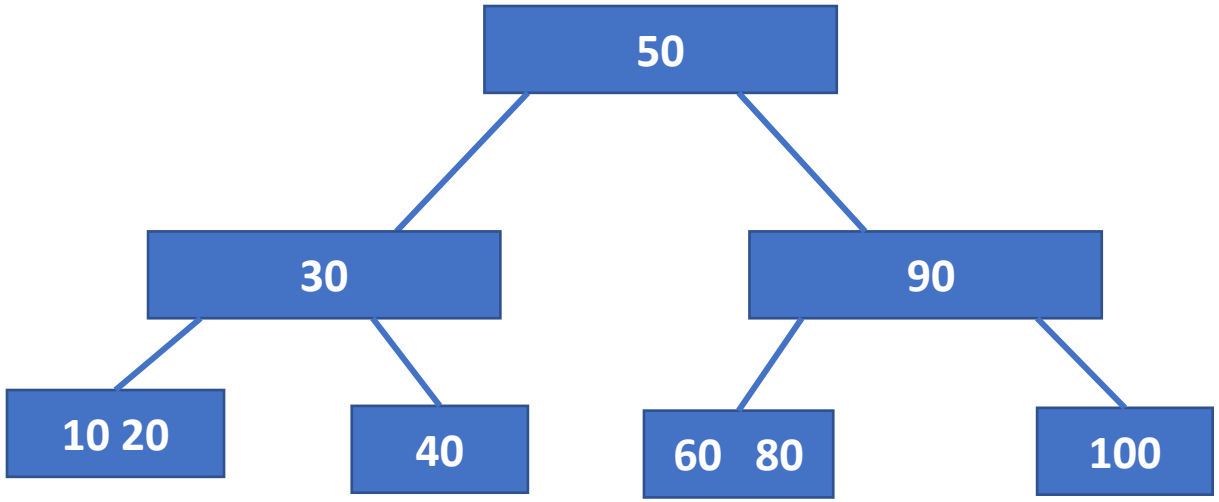


Deletion Algorithm

Delete 70

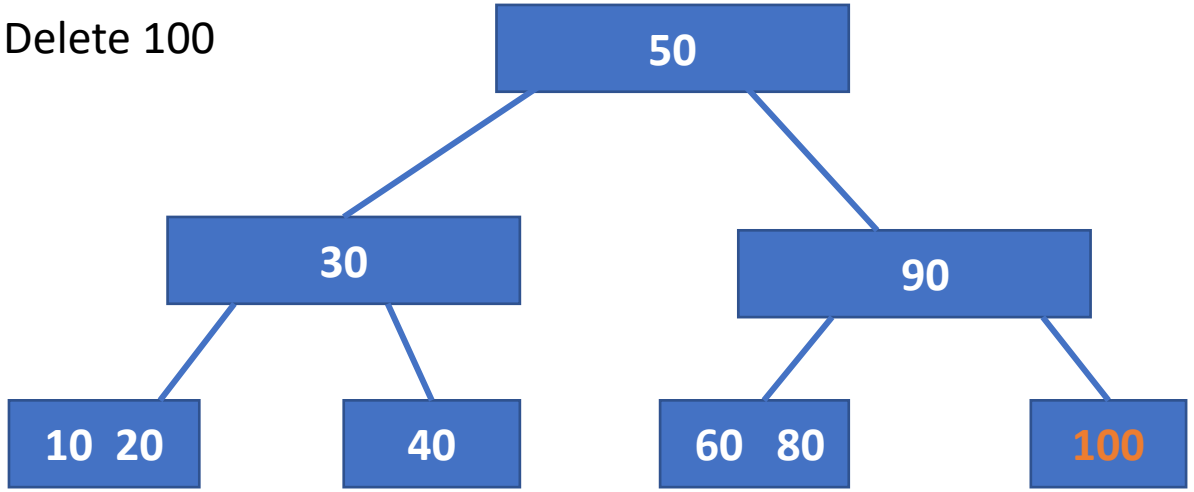


Deletion Algorithm

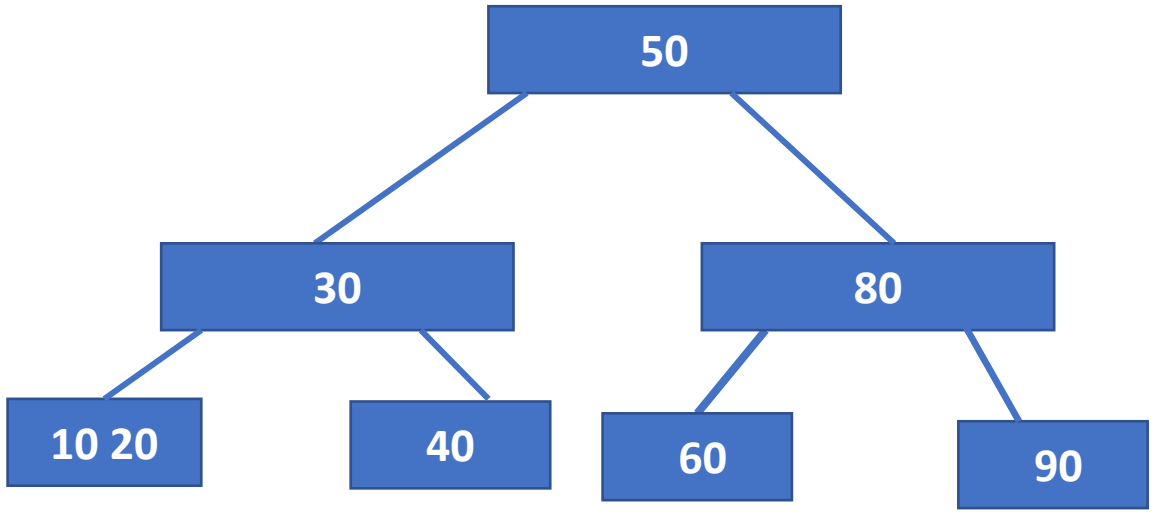


Deletion Algorithm

Delete 100

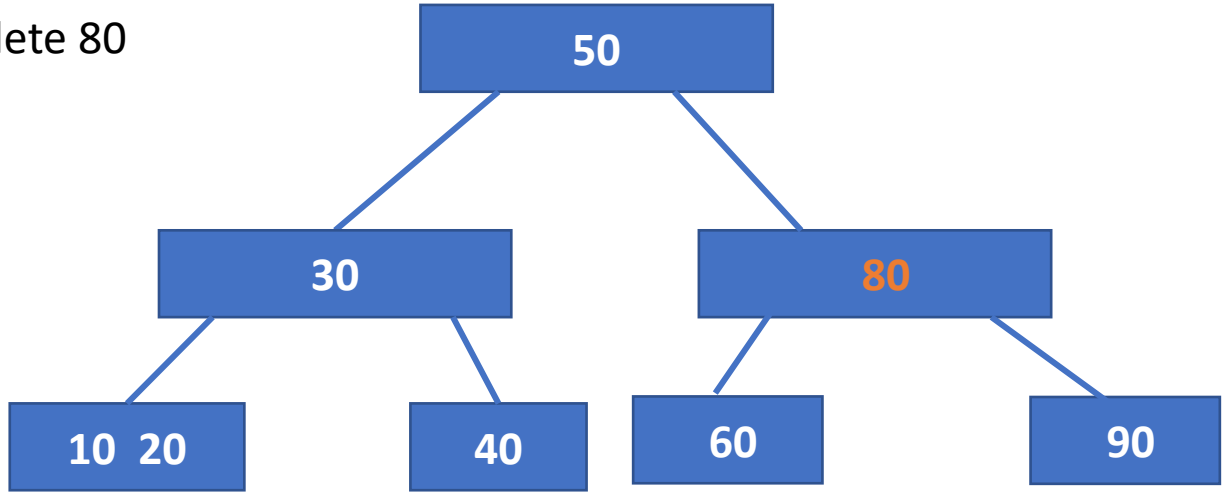


Deletion Algorithm



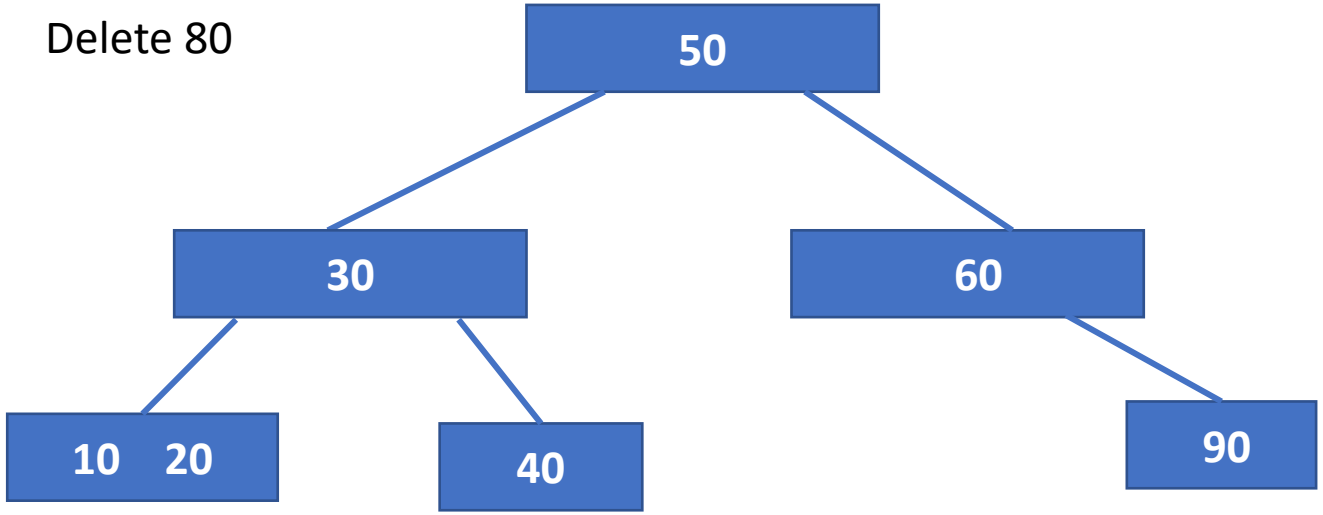
Deletion Algorithm

Delete 80



Deletion Algorithm

Delete 80



Deletion Algorithm

1.Locate the leaf node.
2.If there are more than m/2 keys in the leaf node then delete key
3.If the leaf node doesn't contain m/2 keys then
    a)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.
    b)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.
4.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.
5.If parent is left with less than m/2 nodes then, apply the above process on the parent too.

THANK YOU

Saritha

Department of Computer Science & Engineering

Saritha.k@pes.edu

9844668963