

# ECAP770

ADVANCE DATA STRUCTURES

---

Ashwani Kumar  
Assistant Professor



# Learning Outcomes



After this lecture, you will be able to

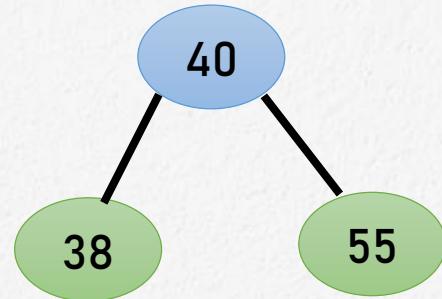
- Understand 2-3 trees properties and operations

# 2-3 Tree

- A 2-3 tree data structure is a specific form of a B tree where every node with children has either two children and one data element or three children and two data elements.
- It is a self-balancing tree. It is balanced with every leaf node at equal distance from the root node.

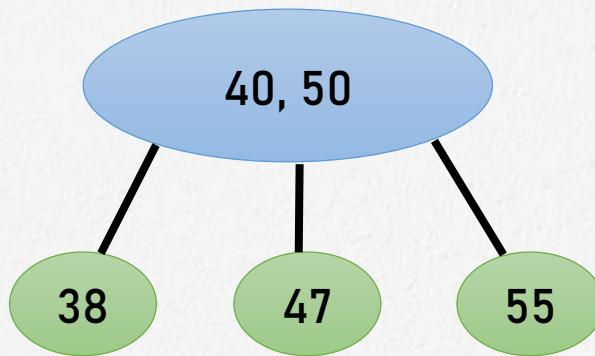
# 2 node

- 2-Node: A node with a single data element that has two child nodes.



# 3 node

- 3-Node: A node with two data elements that has three child nodes



# Properties of 2-3 Trees

- Every internal node in the tree is a 2-node or a 3-node i.e it has either one value or two values.
- A node with one value is either a leaf node or has exactly two children. Values in left sub tree < value in node < values in right sub tree.
- Data stored in sorted manner.
- Insertion operation performed in leaf node.

# Properties of 2-3 Trees

- A node with two values is either a leaf node or has exactly 3 children. It cannot have 2 children.
- Values in left sub tree < first value in node < values in middle sub tree < second value in node < value in right sub tree.
- All leaf nodes are at the same level.

# Example: 2-3 Tree

Data: 50 30 20 70 60 40 65



Step 1



Step 2

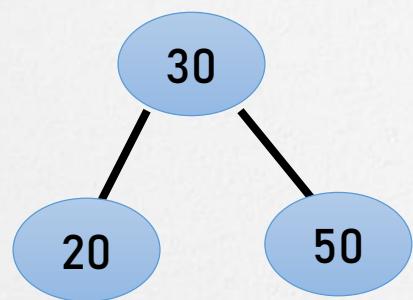


Splitting

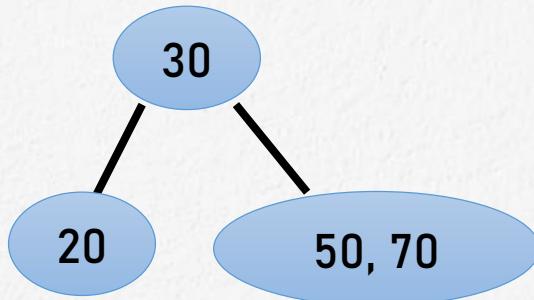
Step 3

# Example: 2-3 Tree

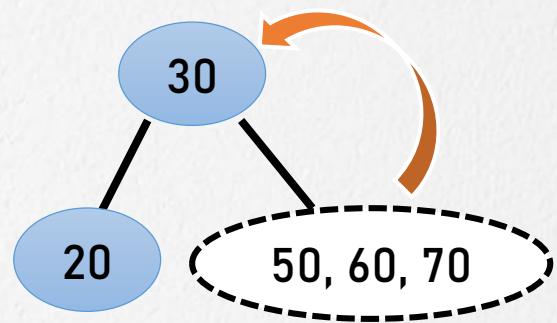
Data: 50 30 20 70 60 40 65



Step 4



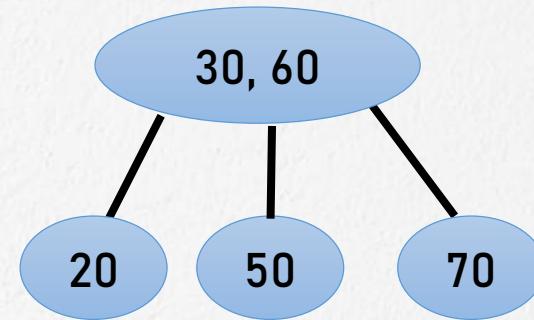
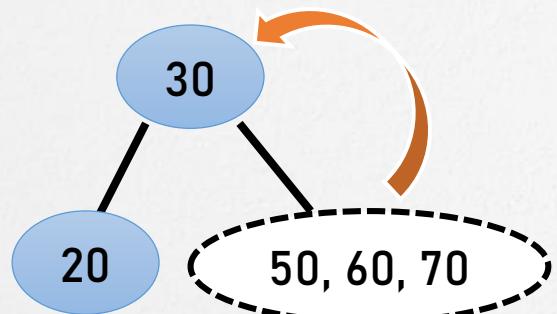
Step 5



Step 6

# Example: 2-3 Tree

Data: 50 30 20 70 60 40 65

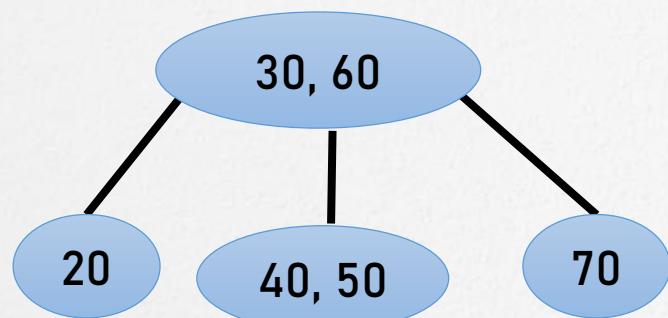


Step 6

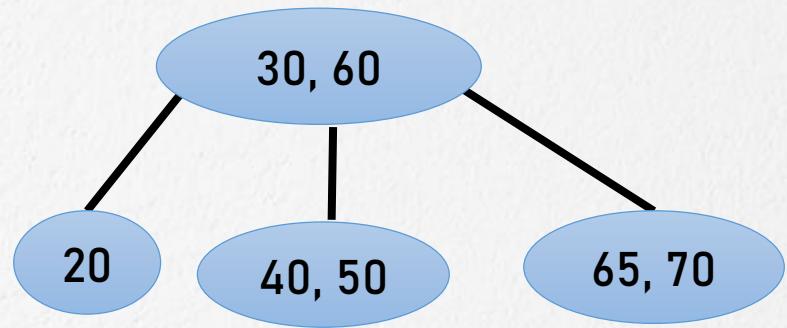
Step 7

# Example: 2-3 Tree

Data: 50 30 20 70 60 40 65



Step 8



Step 9

# 2-3 Tree operations

Insertion

Deletion

Search

# Insertion operation

- If the tree is empty, create a node and put value into the node
- Otherwise find the leaf node where the value belongs.
- If the leaf node has only one value, put the new value into the node

# Insertion operation

- If the leaf node has more than two values, split the node and promote the median of the three values to parent.
- If the parent then has three values, continue to split and promote, forming a new root node if necessary

# Search operation

- If Tree is empty, return False (data item cannot be found in the tree).
- If current node contains data value which is equal to data, return True.
- If we reach the leaf-node and it doesn't contain the required key value, return False.

# Search operation

## Recursive Calls

- If  $\text{data} < \text{currentNode.leftVal}$ , we explore the left sub tree of the current node.
- Else if  $\text{currentNode.leftVal} < \text{data} < \text{currentNode.rightVal}$ , we explore the middle sub tree of the current node.
- Else if  $\text{data} > \text{currentNode.rightVal}$ , we explore the right sub tree of the current node.

# Deletion process

There are three cases in deletion process

1. When the record is to be removed from a leaf node containing two records.

In this case, the record is simply removed, and no other nodes are affected.

# Deletion process

2. when the only record in a leaf node is to be removed.
3. when a record is to be removed from an internal node.

In both the second and the third cases, the deleted record is replaced with another that can take its place while maintaining the correct order of 2-3 tree.

That's all for now...