# CAP770

## B-Tree

**B-Tree is a self-balanced search tree in which every node contains multiple keys and has more than two children.**

A B-Tree is a special kind of tree in a data structure which is also called m-way tree

## M-way tree

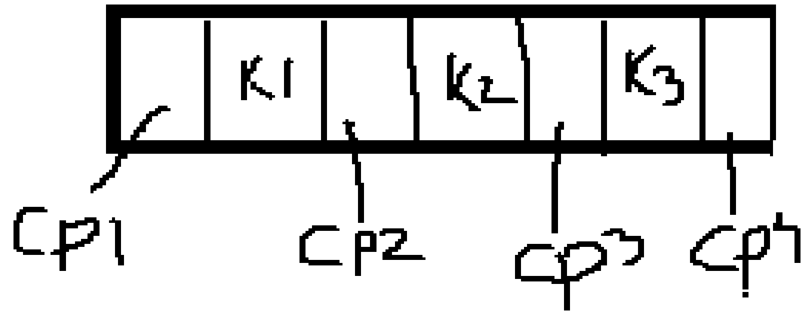
A **multiway tree** is a tree that can have more than two children.
A **multiway tree of order m** (or an **m-way tree**) is one in which a tree can have m children.

Each node can have m-children

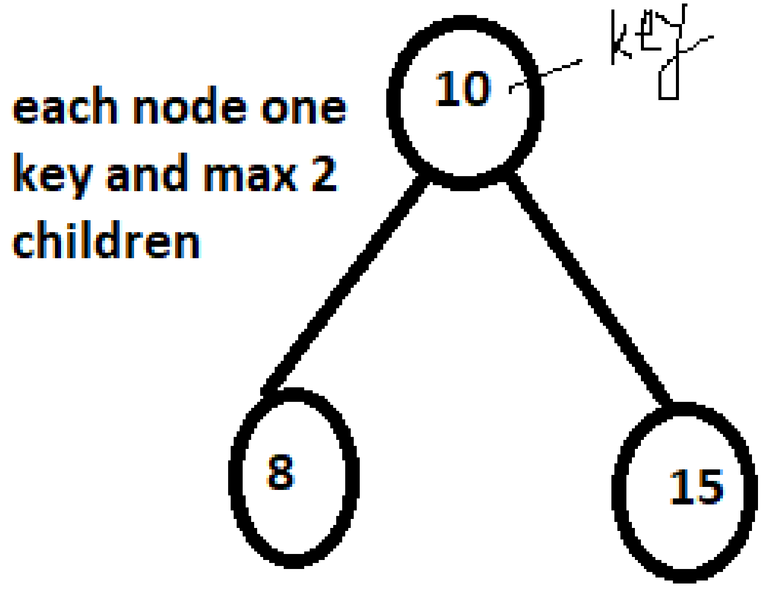
And m-1 children

Structure in node representatio

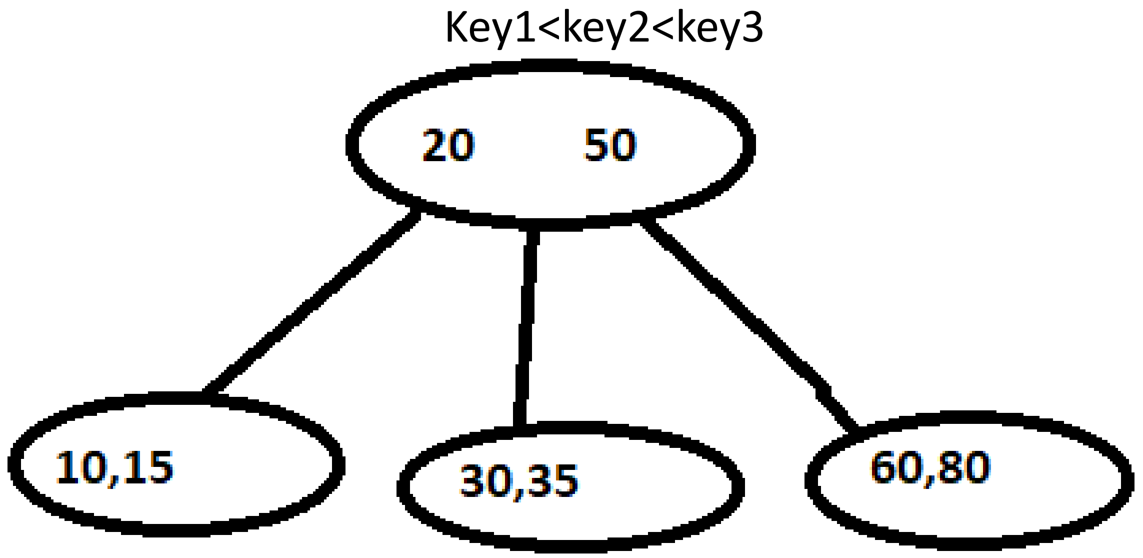
4-way Tree



BST



M-way search tree



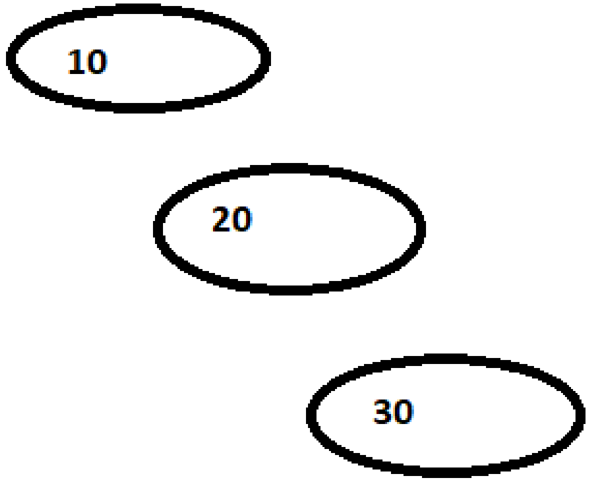
Key:2

Max children=3 so this is called

3- way search tree

Suppose we want to create 10-way tree

And we have key: 10,20,30



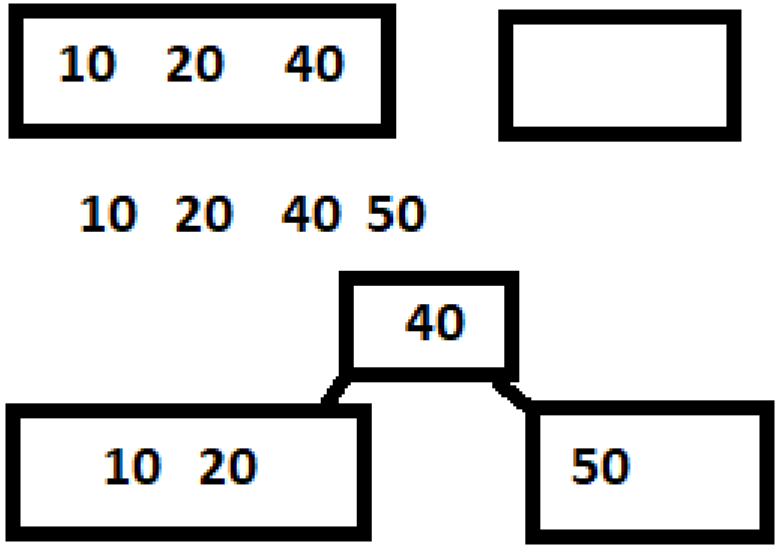
To overcome this problem go for B-Tree

B-tree is a M-way tree with some rule.

1. Every node should have M/2 children
2. Root can have minimum 2 children
3. All leaf at same level
4. All the key values in a node must be in Ascending Order.

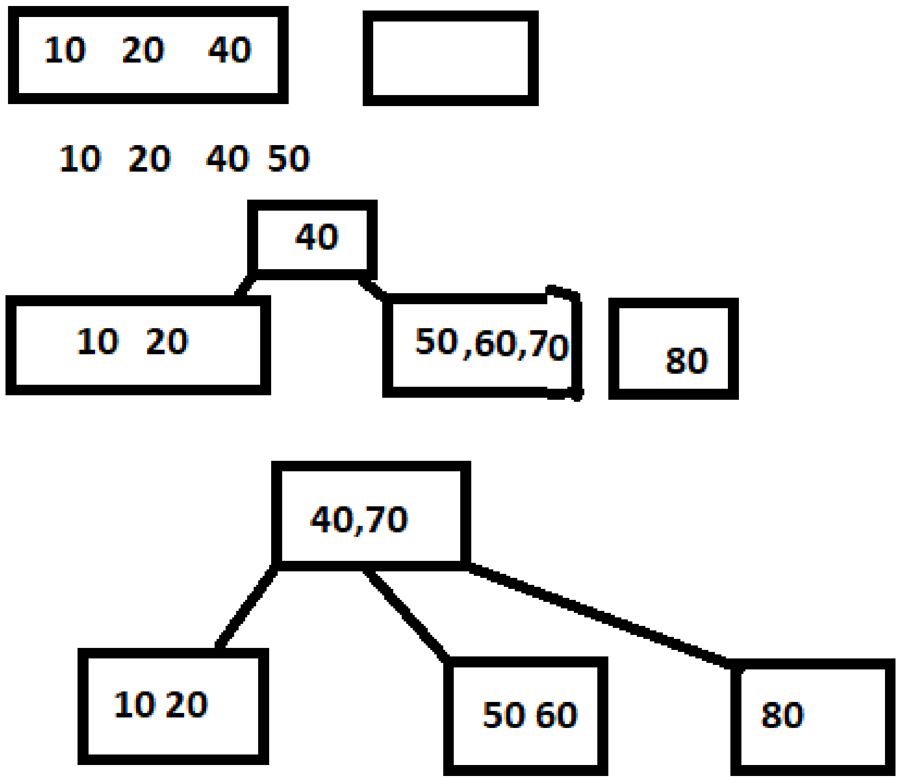
M=4-way tree

Keys: 10,20,40,50



M=4-way tree

Keys: 10,20,40,50,60,70,80



## Why use B-Tree

Here, are reasons of using B-Tree

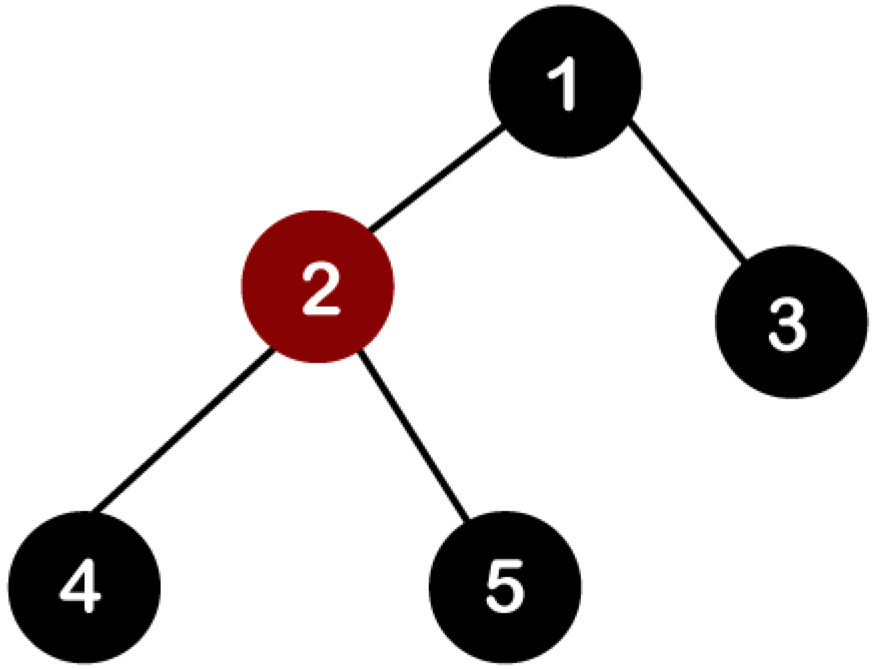
- It is a specially designed technique for handling a bulky amount of data.
- It is a useful algorithm for databases and file systems.

## *red-black tree*

A *red-black tree* is a binary search tree with one extra attribute for each node: the *colour*, which is either red or black.

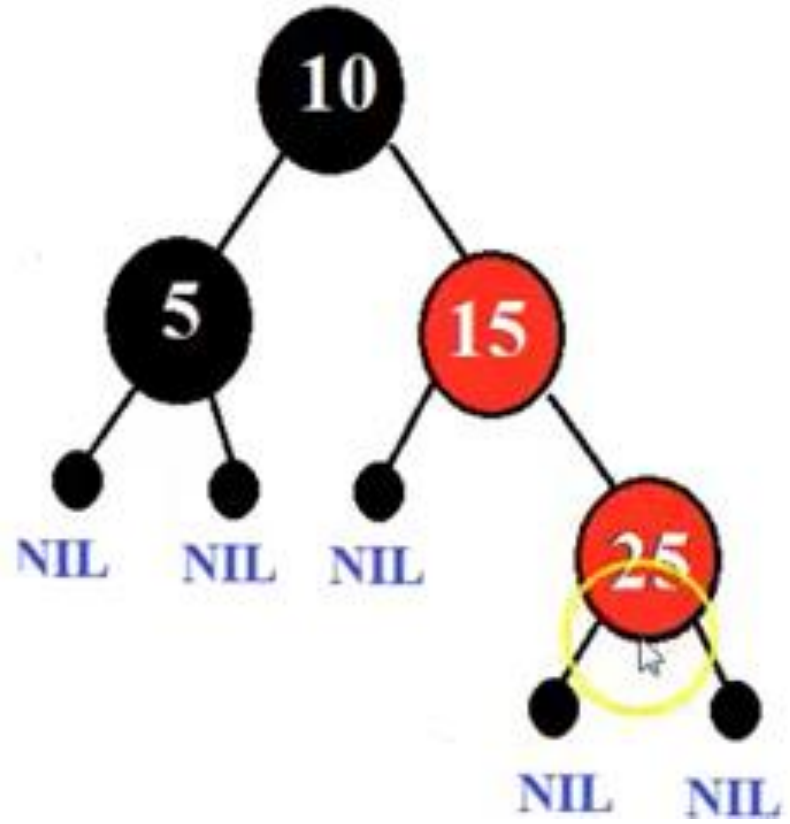
A red-black tree satisfies the following properties:

1. **Red/Black Property:** Every node is colored, either red or black.
2. **Root Property:** The root is black.
3. **Leaf Property:** Every leaf (NIL) is black.
4. **Red Property:** If a red node has children then, the children are always black.


1
2
3
4
5

# How to Identify Red and Black Tree?

**Example#1 to identify Red and Black Tree**



0) BST?
1) Every node is Red or Black?
2) Black Root?
3) Does two adjacent Red nodes?

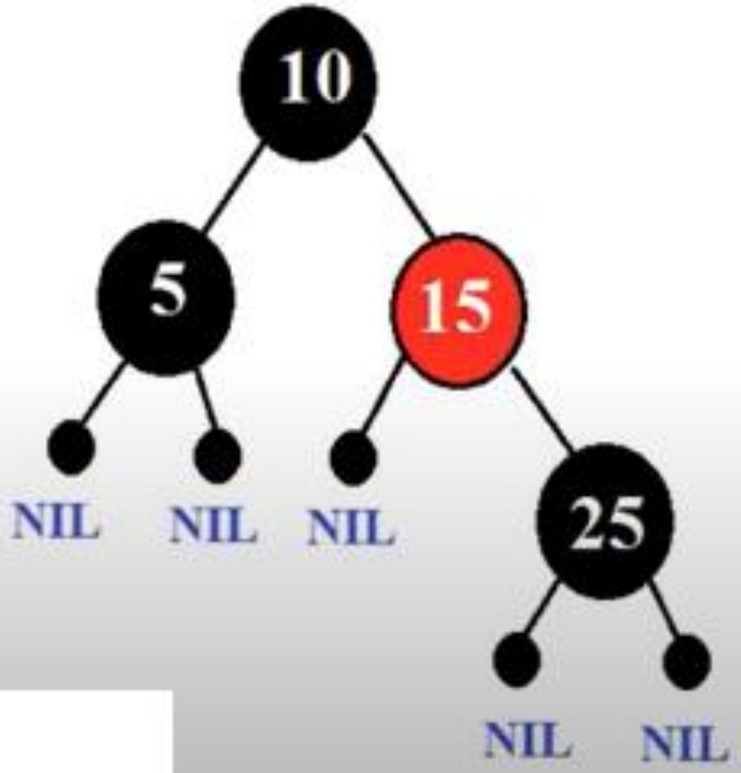
All conditions are not True.

Condition 3 is False.

Given tree is not RB Tree.

## How to Identify Red and Black Tree?

**Example#2 to identify Red and Black Tree**

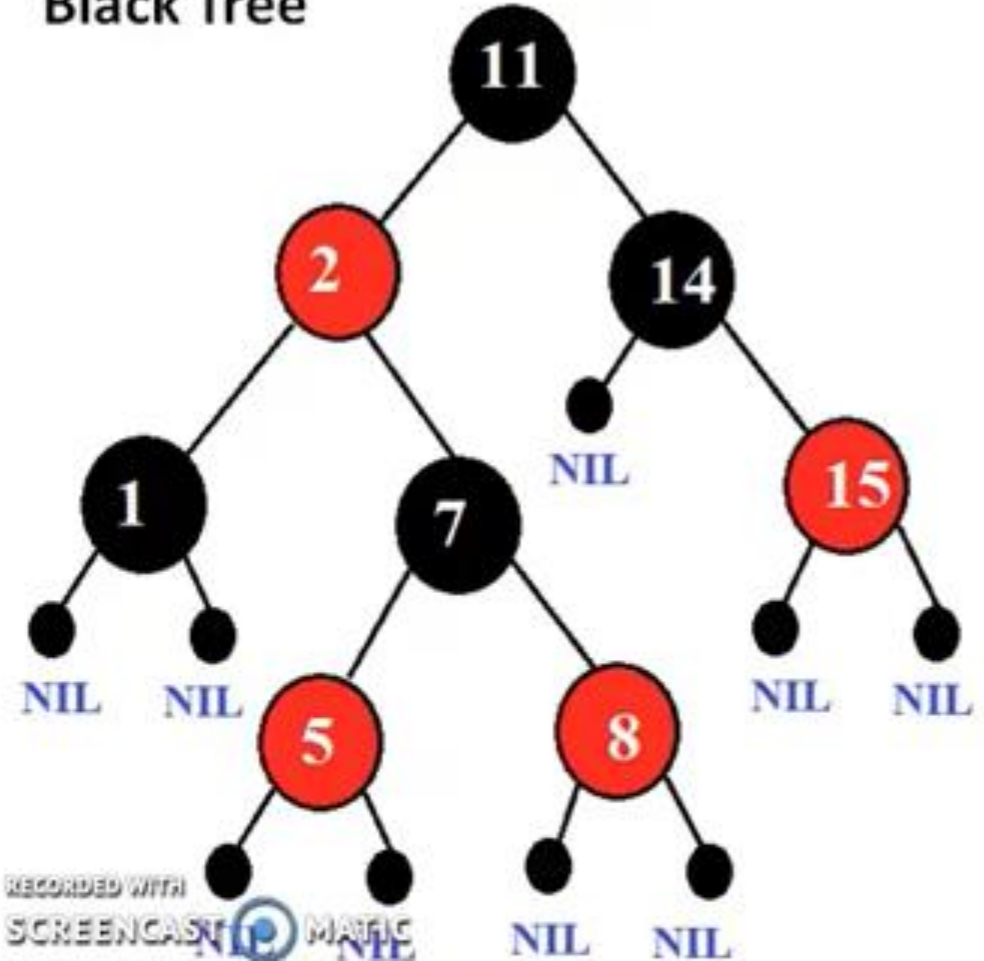


0) BST?
1) Every node is Red or Black?
2) Black Root?
3) Does two adjacent Red nodes?
4) Does Count of Black nodes is equal in every path?

All conditions are not True.
Condition 4 is False.
Given tree is not RB Tree.

# How to Identify Red and Black Tree?

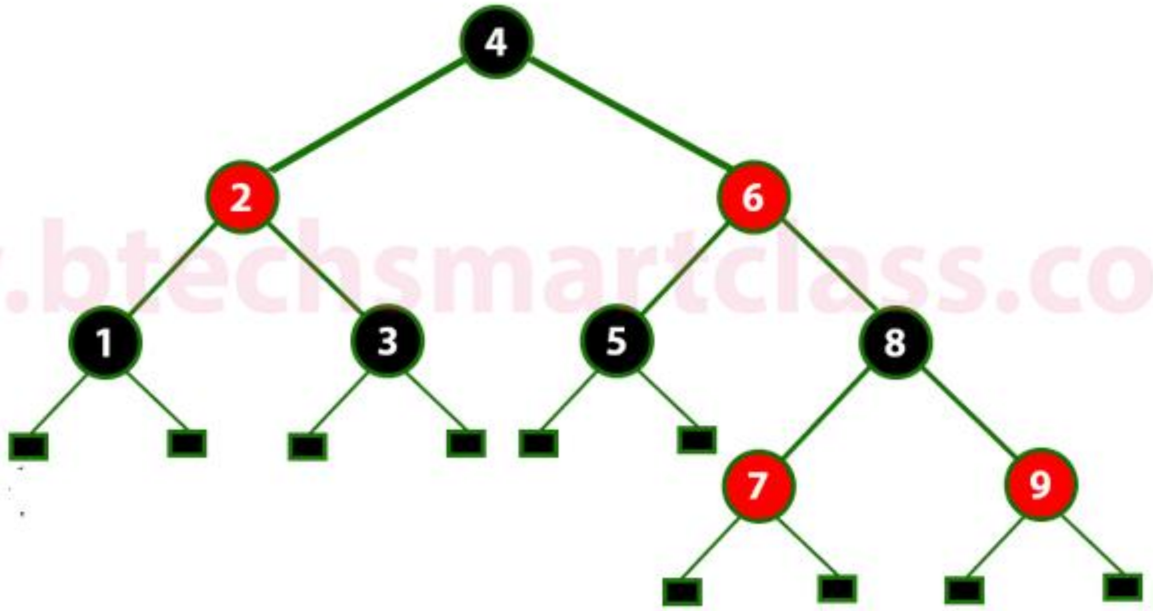
**Example#3 to identify Red and Black Tree**



0) BST?
1) Every node is Red or Black?
2) Black Root?
3) Does two adjacent Red nodes?
4) Does Count of Black nodes is equal in every path?
5) Black leaves?

All conditions are true.

Given tree is RB Tree.


4
2
6
1
3
5
8
7
9

## splay trees: properties and operations

- A splay tree is a self-balancing tree
- A splay tree contains the same operations as a Binary search tree, i.e., Insertion, deletion and searching, but it also contains one more operation, i.e., splaying. So. all the operations in the splay tree are followed by splaying.
- Here splaying means that the operation that we are performing on any element should become the root node after performing some rearrangements. The rearrangement of the tree will be done through the rotations.

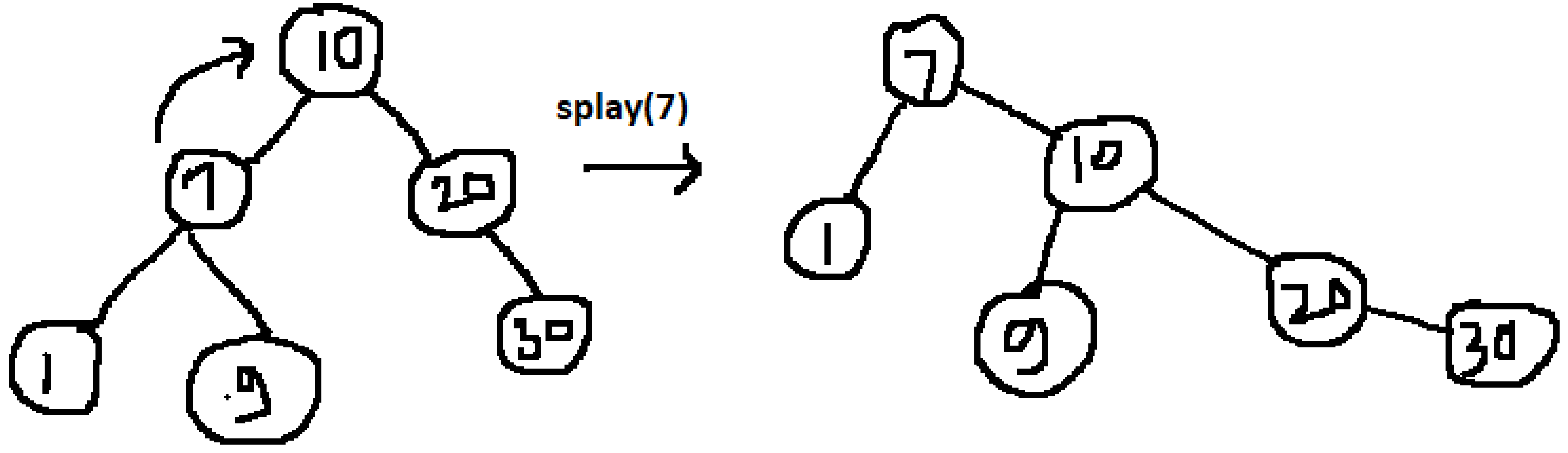
Rotations in Splay Tree

1. Zig Rotation
2. Zag Rotation
3. Zig - Zig Rotation
4. Zag - Zag Rotation
5. Zig - Zag Rotation
6. Zag - Zig Rotation

(zig-mean right, zag-means left)

## Zig Rotation

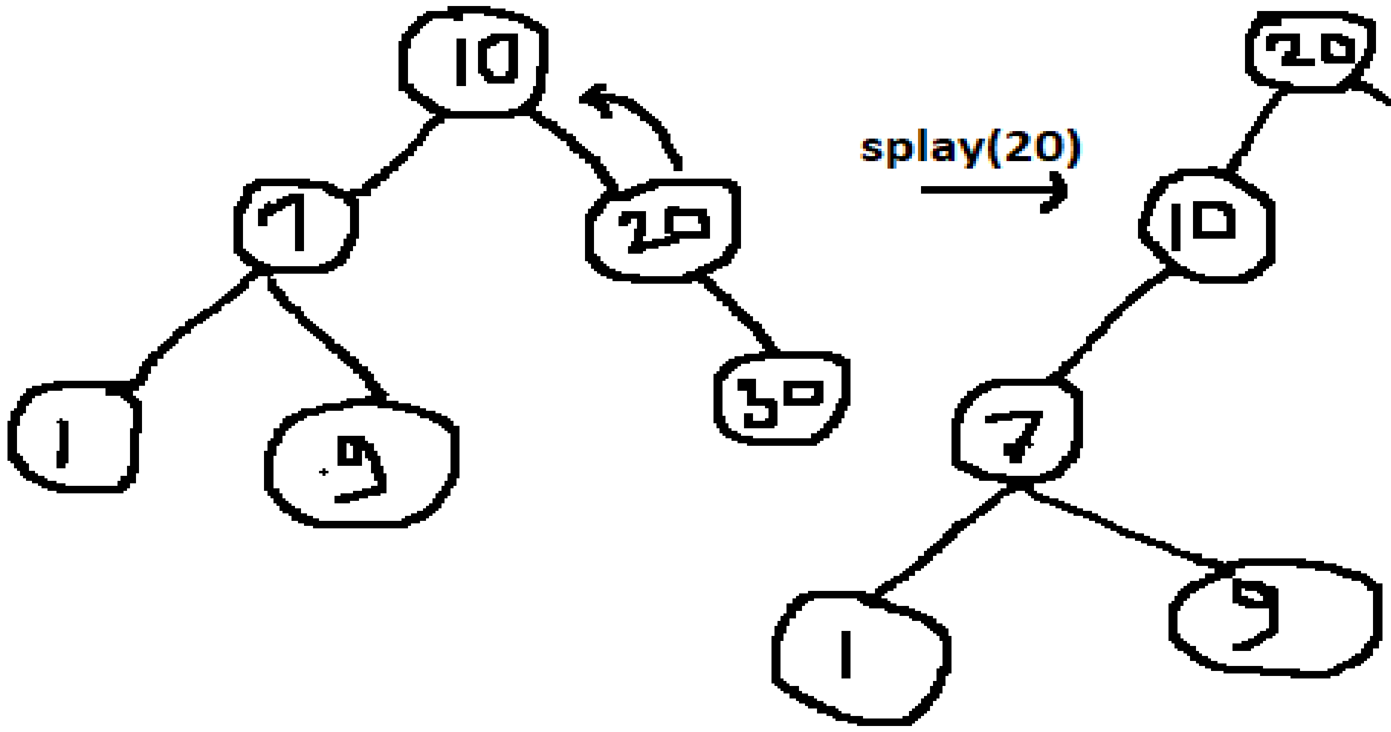
The **Zig Rotation** in splay tree is similar to the single right rotation in AVL Tree rotations. In zig rotation, every node moves one position to the right from its current position.



9 is right of 7 and left of 10

## Zag Rotation

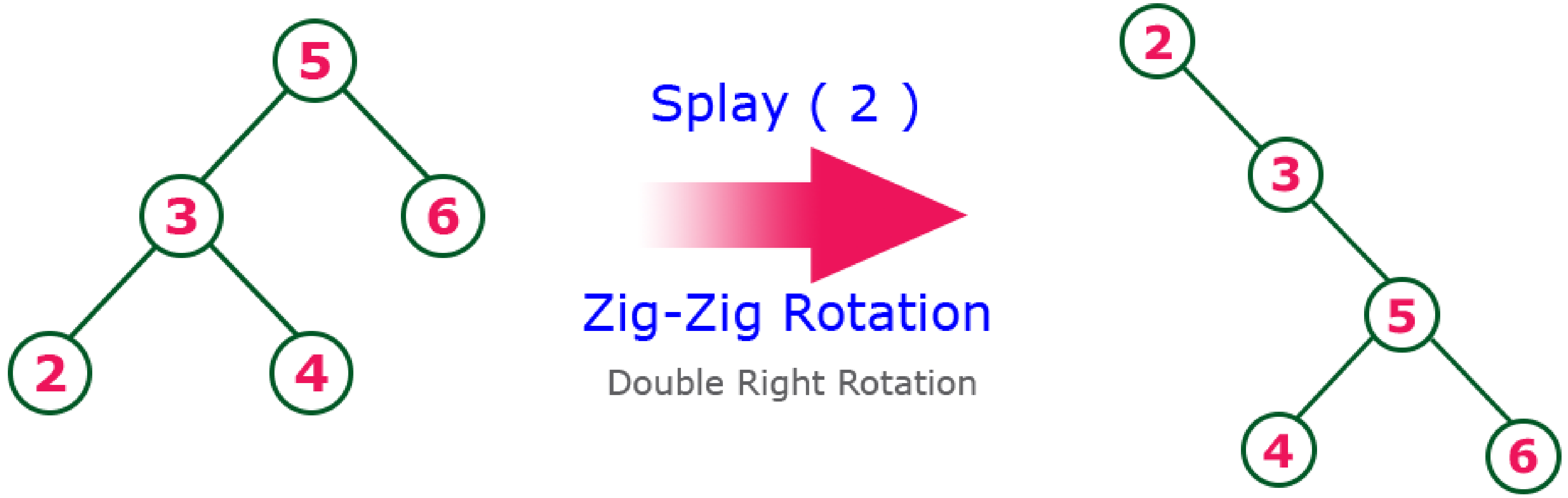
The **Zag Rotation** in splay tree is similar to the single left rotation in AVL Tree rotations. In zag rotation, every node moves one position to the left from its current position.



## Zig-Zig Rotation

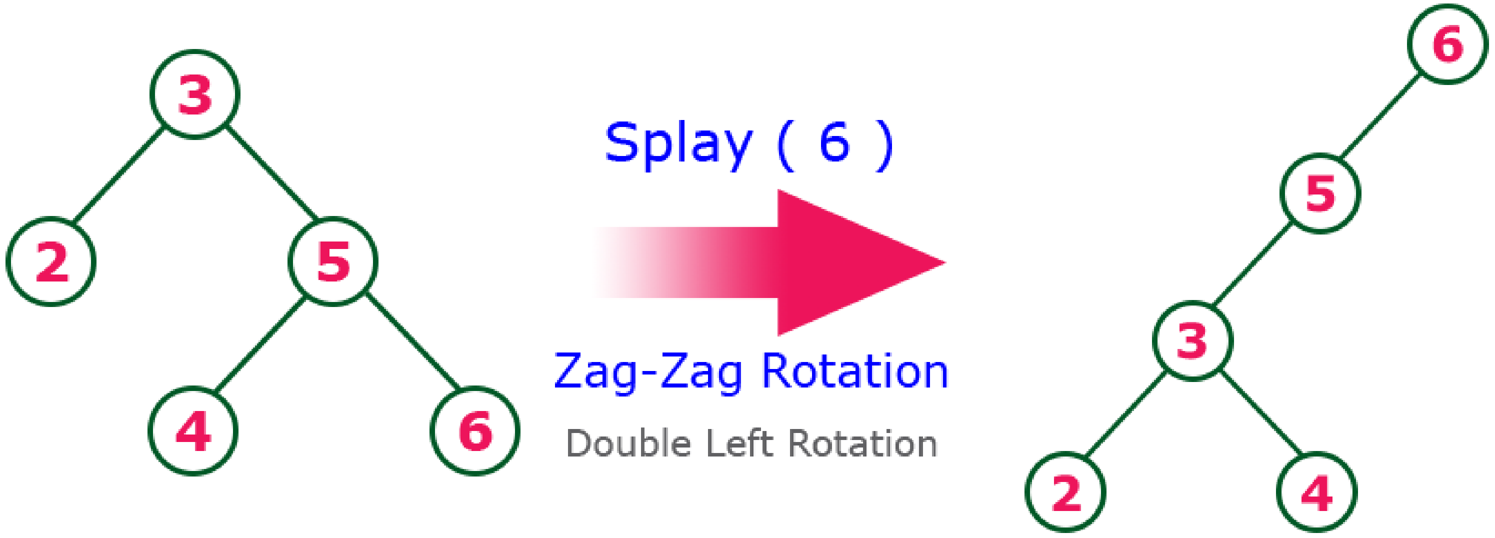
The **Zig-Zig Rotation** in splay tree is a double zig rotation. In zig-zig rotation, every node moves two positions to the right from its current position.

Element should have parent and grand parent to perform Zig-Zig rotation



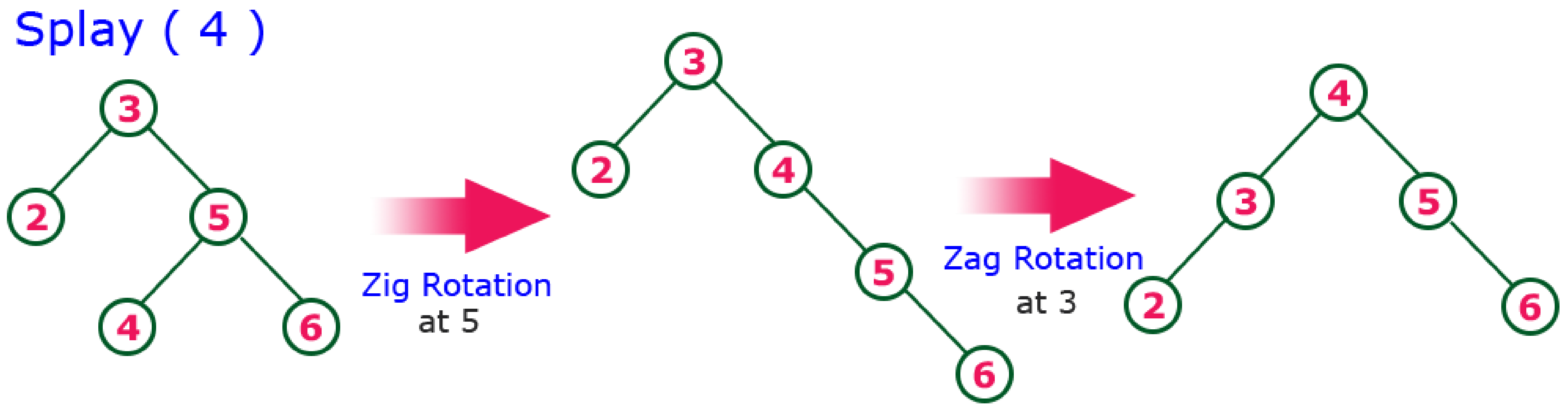
## Zag-Zag Rotation

The **Zag-Zag Rotation** in splay tree is a double zag rotation. In zag-zag rotation, every node moves two positions to the left from its



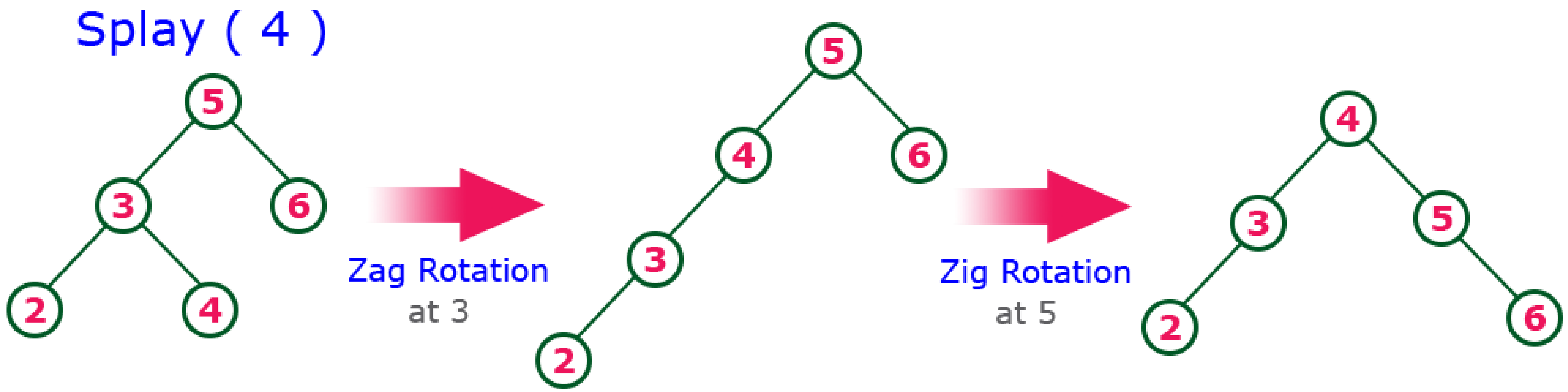
## Zig-Zag Rotation

The **Zig-Zag Rotation** in splay tree is a sequence of zig rotation followed by zag rotation. In zig-zag rotation, every node moves one position to the right followed by one position to the left from its current position.



## Zag-Zig Rotation

- The **Zag-Zig Rotation** in splay tree is a sequence of zag rotation followed by zig rotation. In zag-zig rotation, every node moves one position to the left followed by one position to the right from its current position.

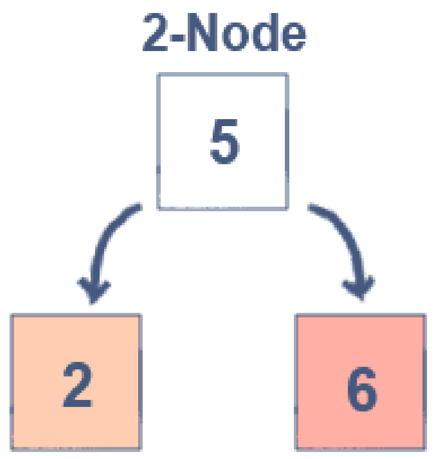


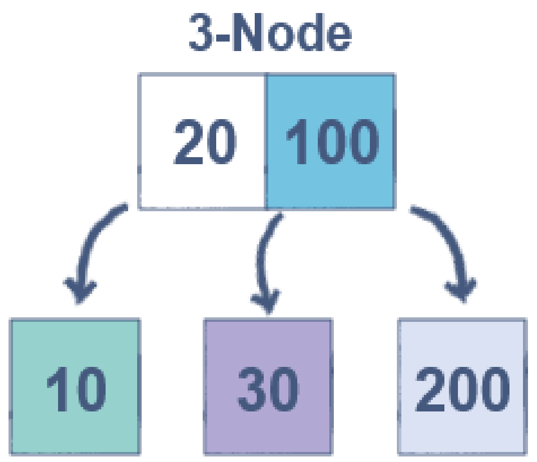
## 2-3 Tree

A 2-3 Tree is a Binary Search Tree where every node can be of two types:

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

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





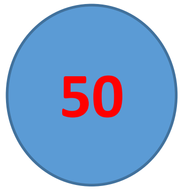
## Properties:

- It is balanced in nature
- Every Internal node is 2-Node or 3-Node
- All leaf nodes are in same level
- Data is store in sorted manner

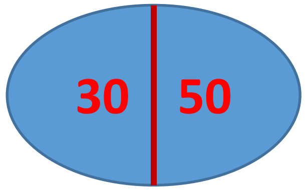
# Example:

Consider Elements: 50, 30, 10, 70, 60

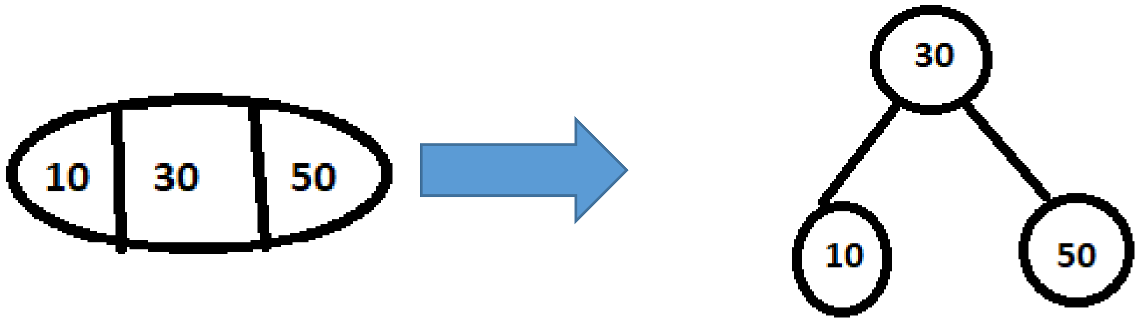
Step1:



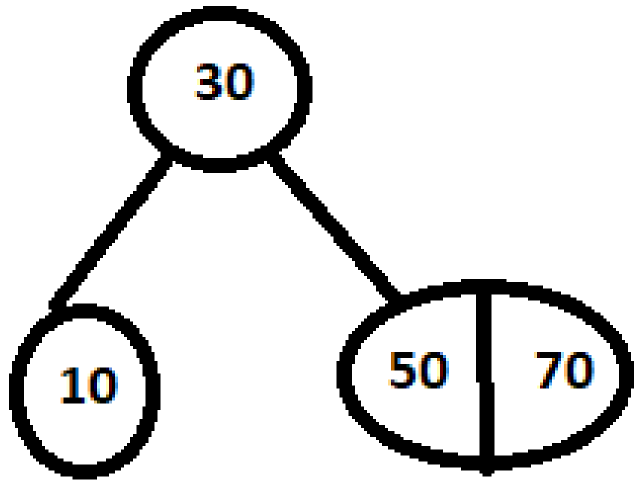
Step2:



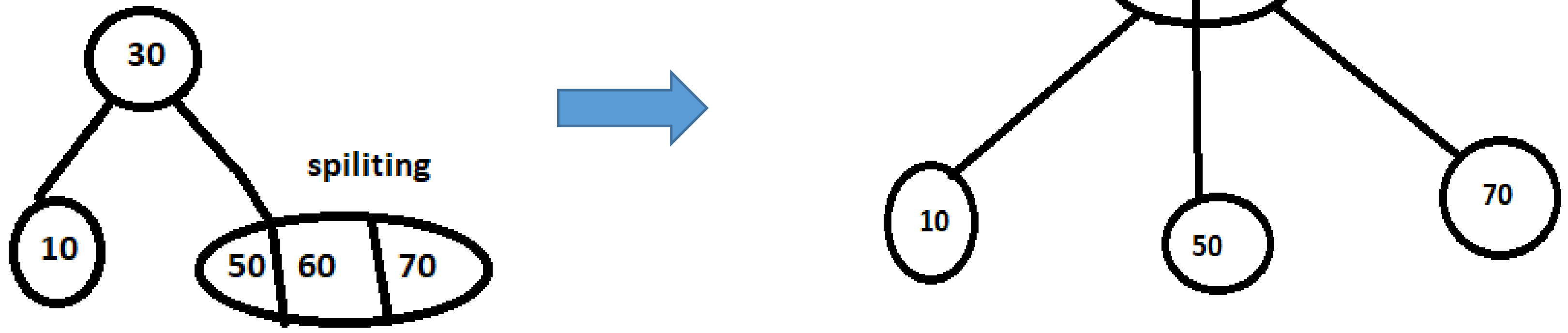
Step3:



Step4:



Step5:


30
10
spiliting
50
60
70
30
60
10
50
70