Presentation on DSA

MDS 1st Semester

Topic: (unit-8)

AVL Trees

Prabin Adhikari

Roll No : 23

Definition

An AVL tree is a binary search tree with a ***balance* condition**.

AVL is named for its inventors: **A**del'son-**V**el'skii and **L**andis

AVL tree *approximates* the ideal tree **(completely balanced tree)**.

AVL tree maintains a height close to the minimum.

In AVL tree, **balance factor** of every node is -1,0 or +1.

Where, Balance factor = height of left sub tree – height of right sub tree.

Every AVL tree is binary search tree, but every binary search tree need not to be AVL tree.

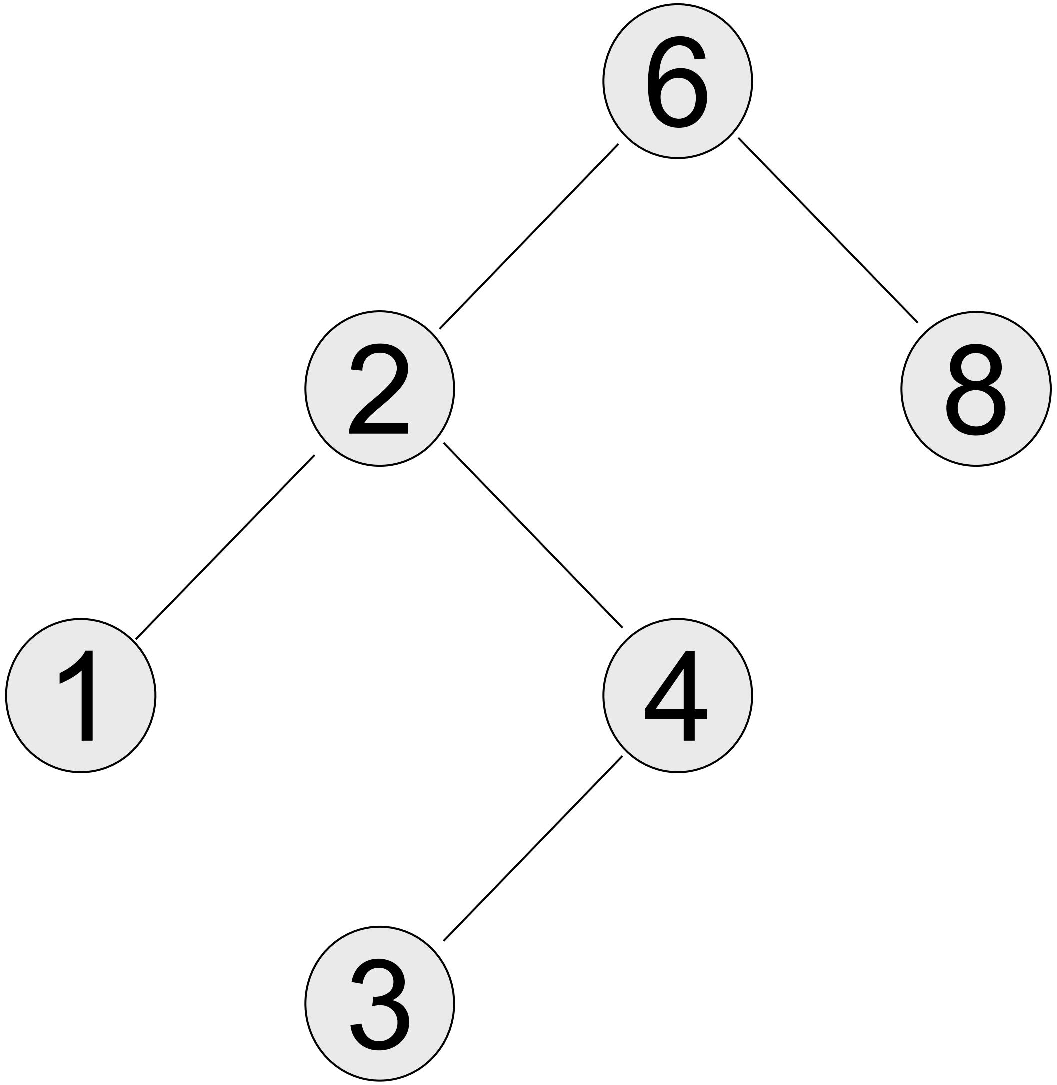
Operation perform as search, insertion, deletion with **O(log n) time complexity.**

.

Balance condition ?
Height ?

An AVL tree is a binary search tree such that for any node in the tree, the height of the left and right subtrees can differ by at most 1.

- Height is the length of the longest path from root to a leaf
- Here in the figure height of the tree is 3, subtree rooted by node 2 is 2, by 8 is 0. Tree is unbalanced



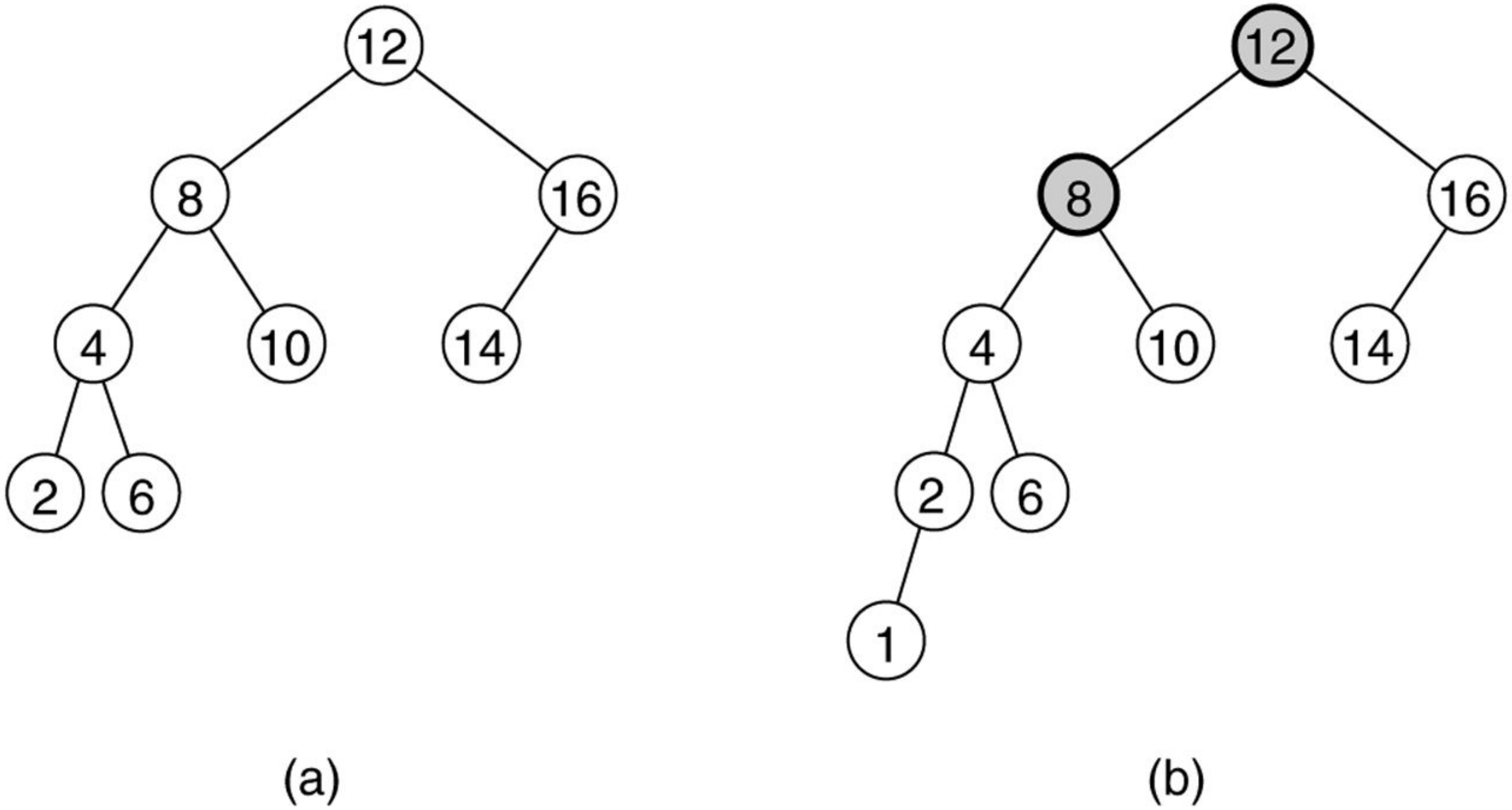
Examples

Two binary search trees:

(a) an AVL tree

(b) not an AVL tree

(unbalanced nodes are darkened)



(a)

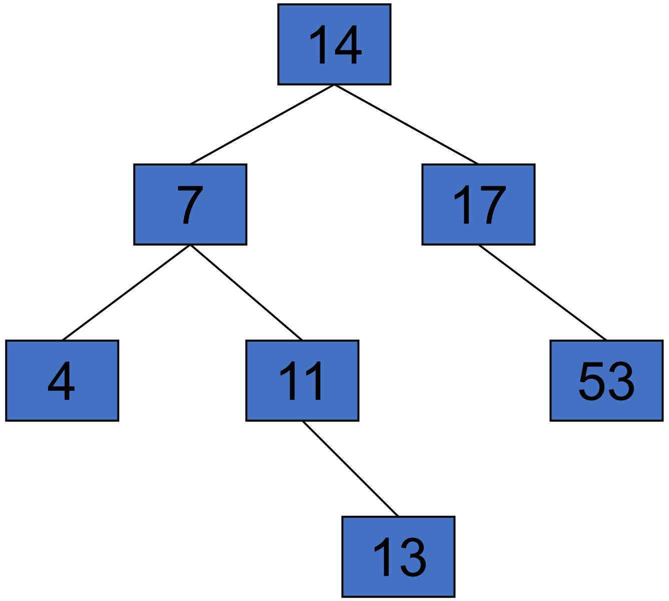
(b)

Properties of AVL Tree

- The depth of a typical node in an AVL tree is very close to the optimal *log N*.
- Consequently, all searching operations in an AVL tree have logarithmic worst-case bounds.
- An update (insert or remove) in an AVL tree could destroy the balance. It must then be rebalanced before the operation can be considered complete.
- After an insertion, only nodes that are on the path from the insertion point to the root can have their balances altered.

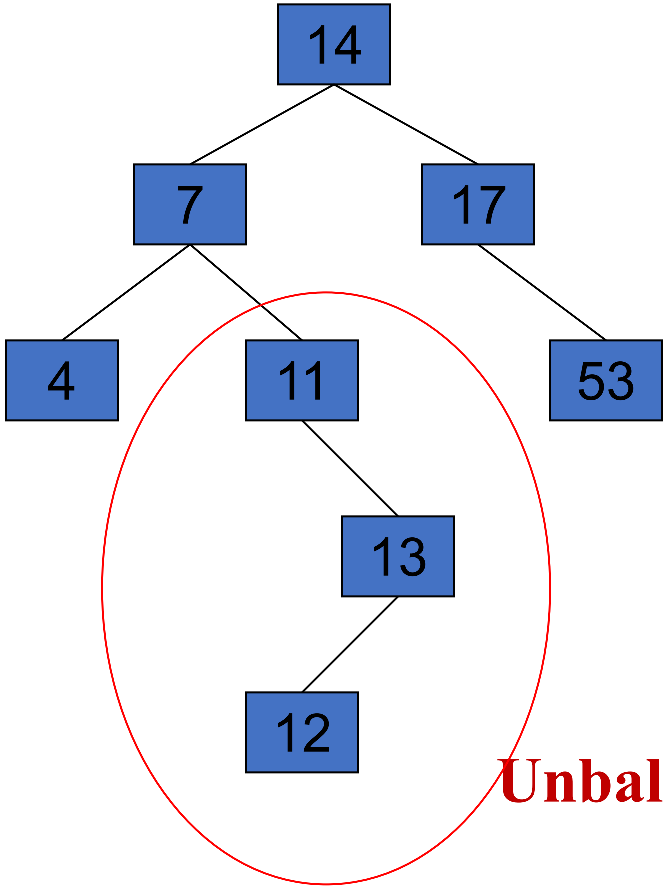
Simple Insertion operation on AVL Tree (Example)

Insert 14, 17, 7, 53, 4,11,13 into an empty AVL tree



Balanced AVL Tree

Now insert 12

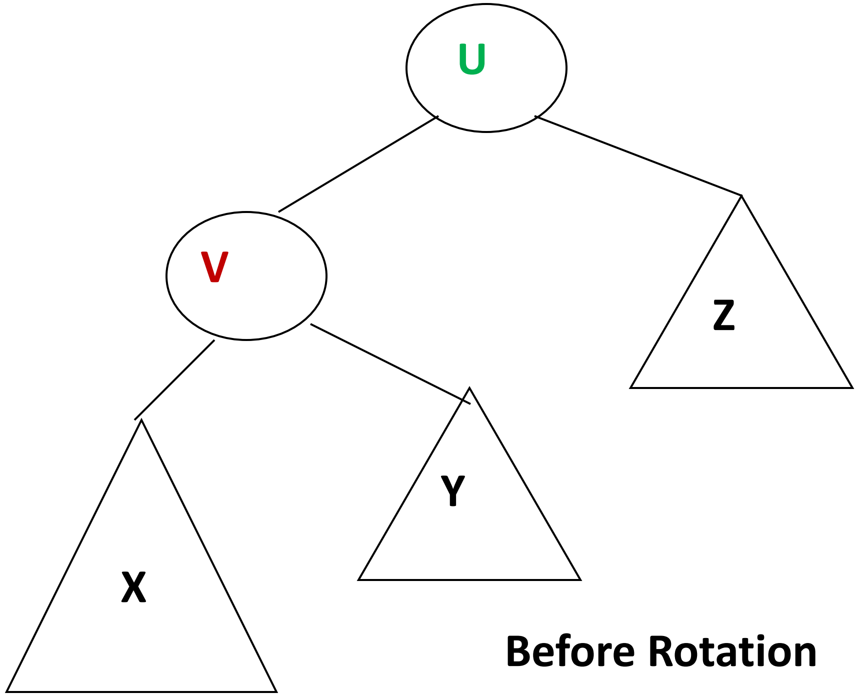


Unbalanced (let's balance)

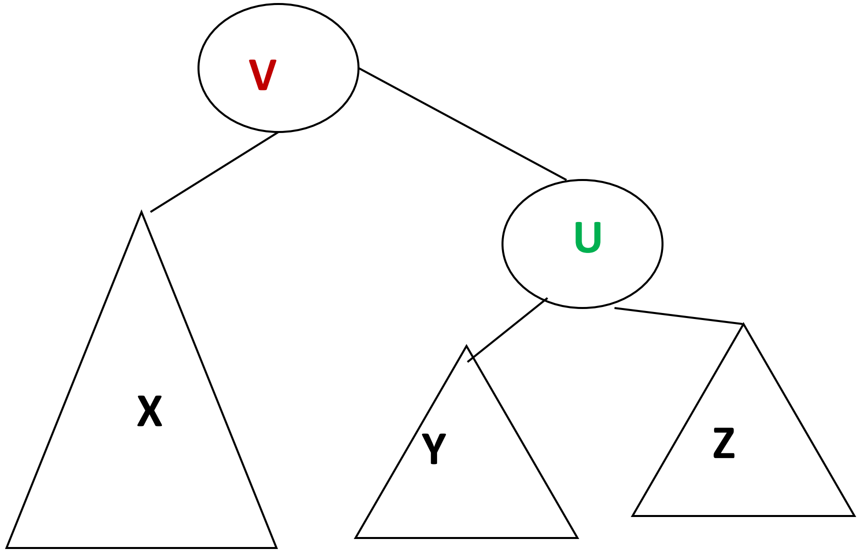
For Balancing the Tree, Single and Double rotation is used

Insertion in left child of left subtree

Single Rotation



Before Rotation

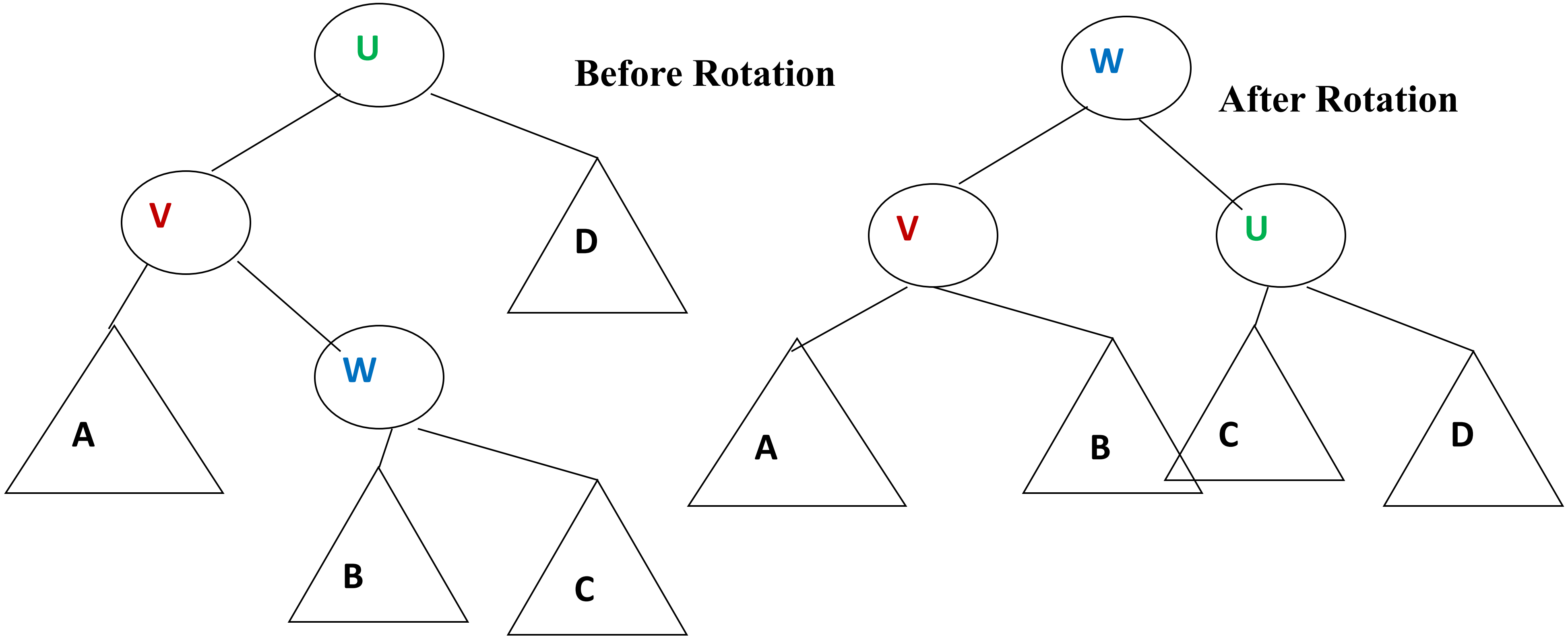


After Rotation

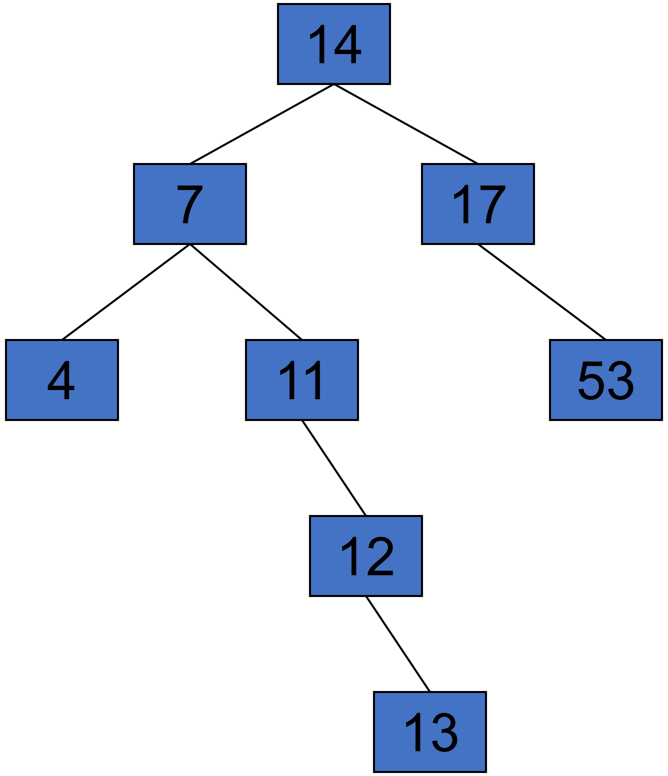
Double Rotation

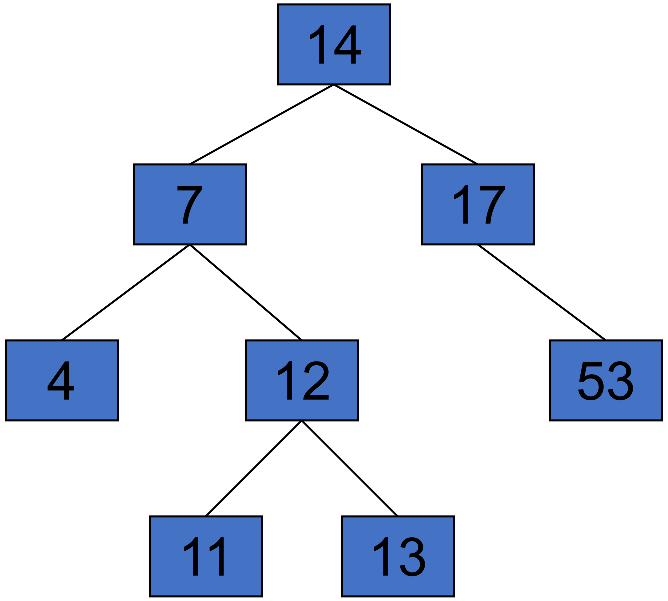
Suppose, imbalance is due to an insertion in the left subtree of right child

Single Rotation does not work!



Above Example





Finally Balanced

Insertion operation Example (step wise)

Insert 3, 2, 1, 4, 5, 6, 7, 16, 15, 14

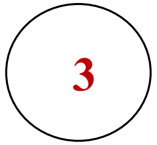


Fig 1

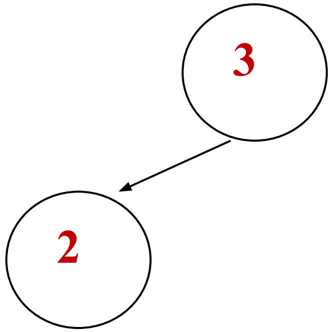


Fig 2

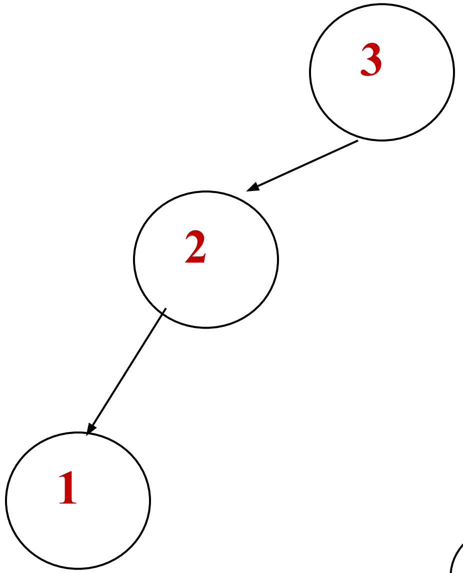


Fig 3

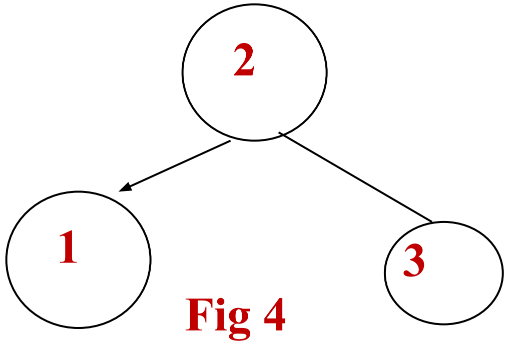


Fig 4

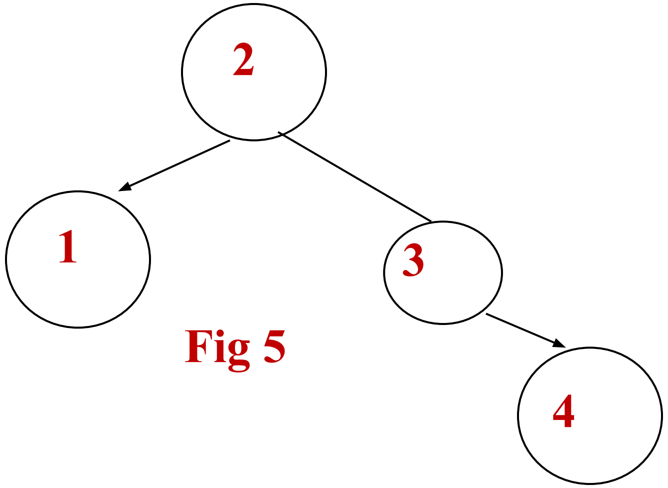


Fig 5

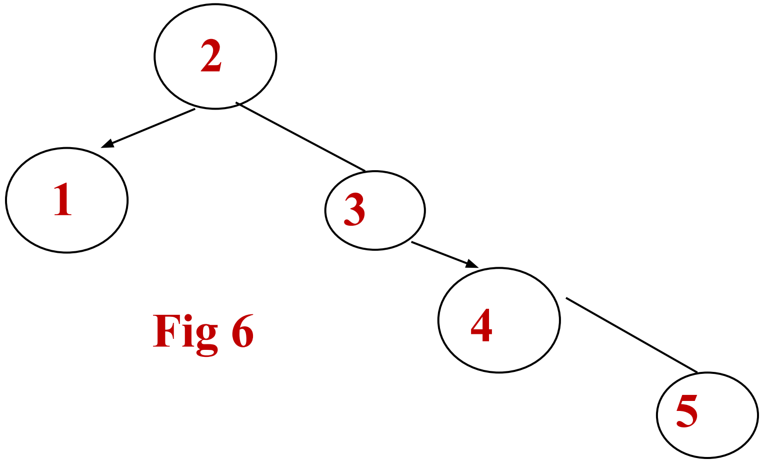


Fig 6

Insert 3, 2, 1, 4, 5, 6, 7, 16, 15, 14

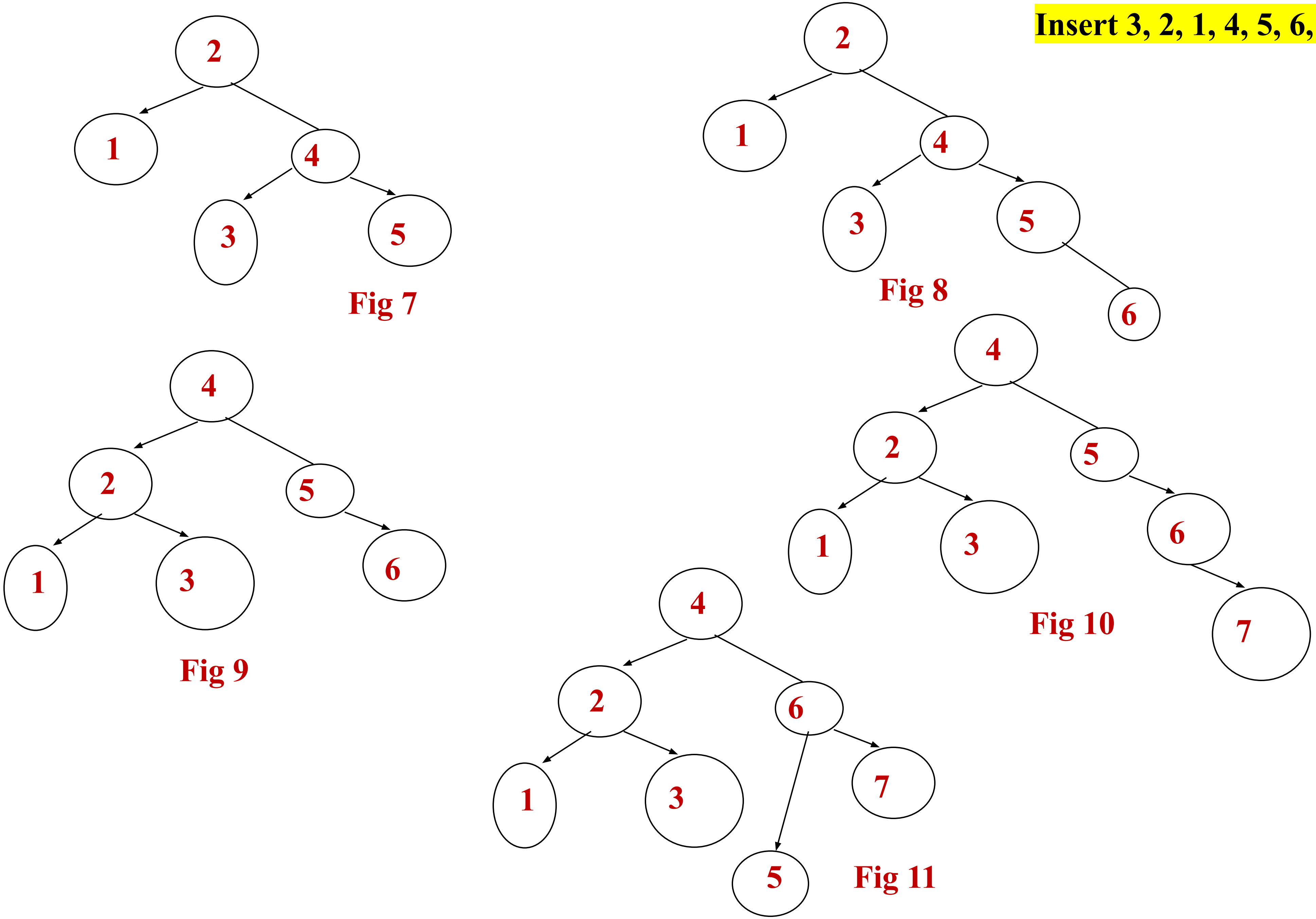


Fig 7

Fig 8

Fig 9

Fig 10

Fig 11

Insert 3, 2, 1, 4, 5, 6, 7, 16, 15, 14

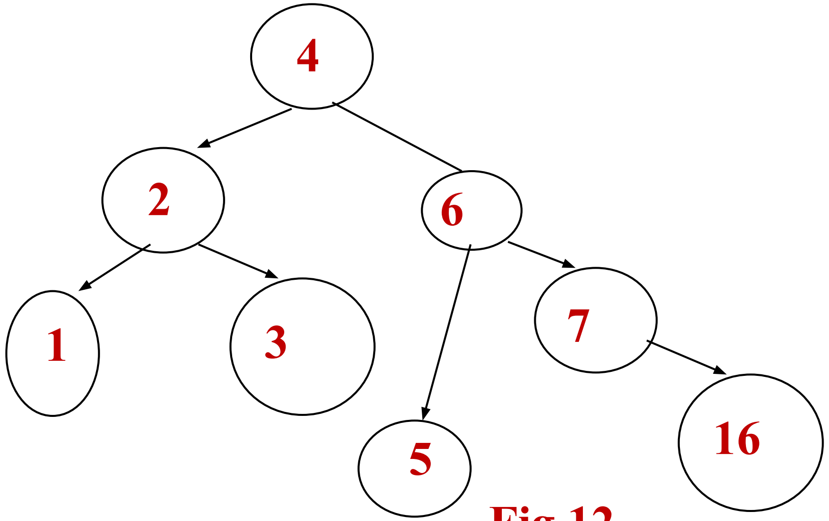


Fig 12

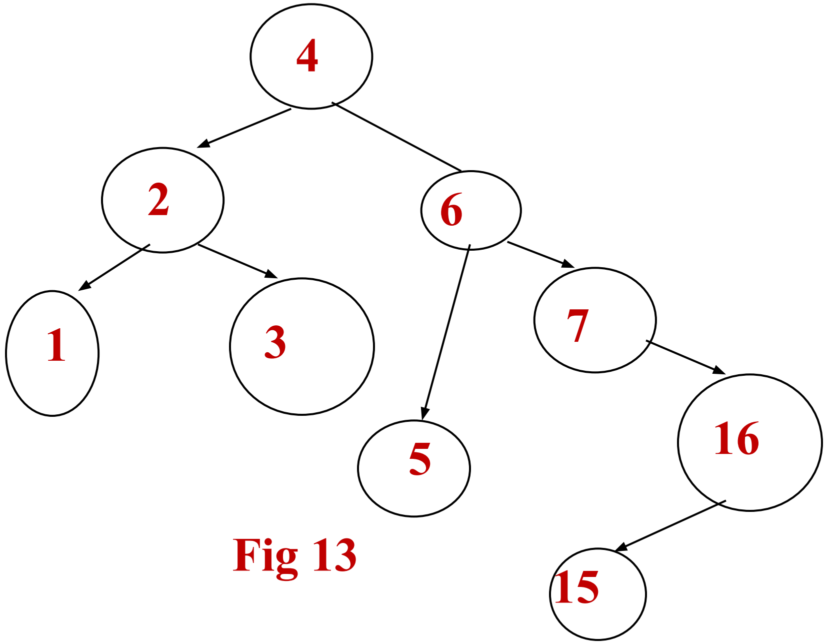


Fig 13

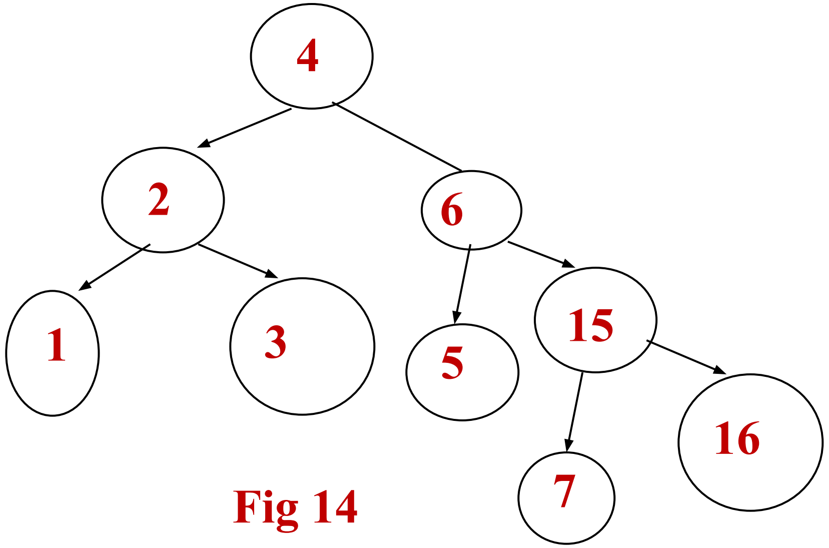


Fig 14

Insert 3, 2, 1, 4, 5, 6, 7, 16, 15, 14

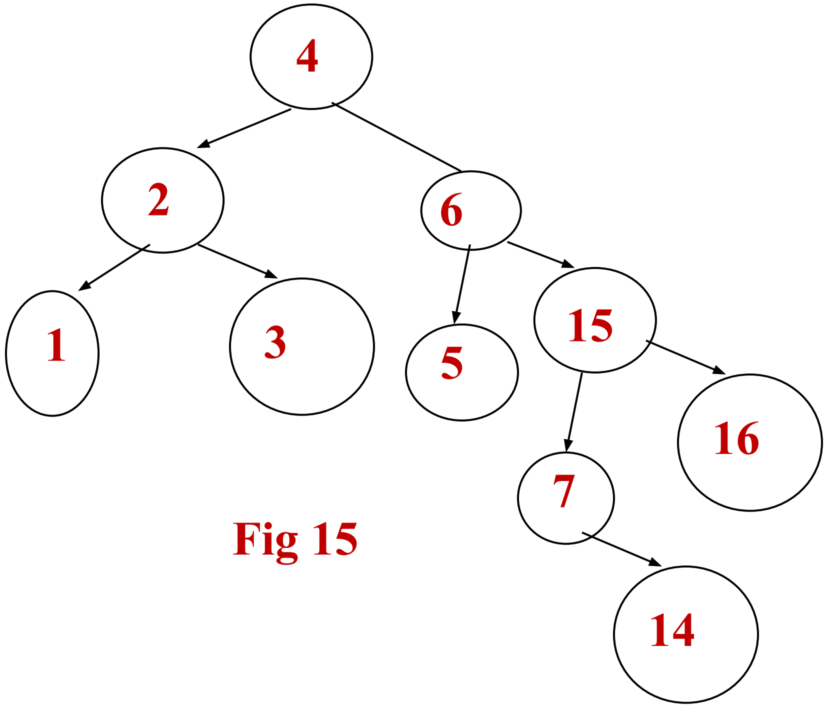


Fig 15

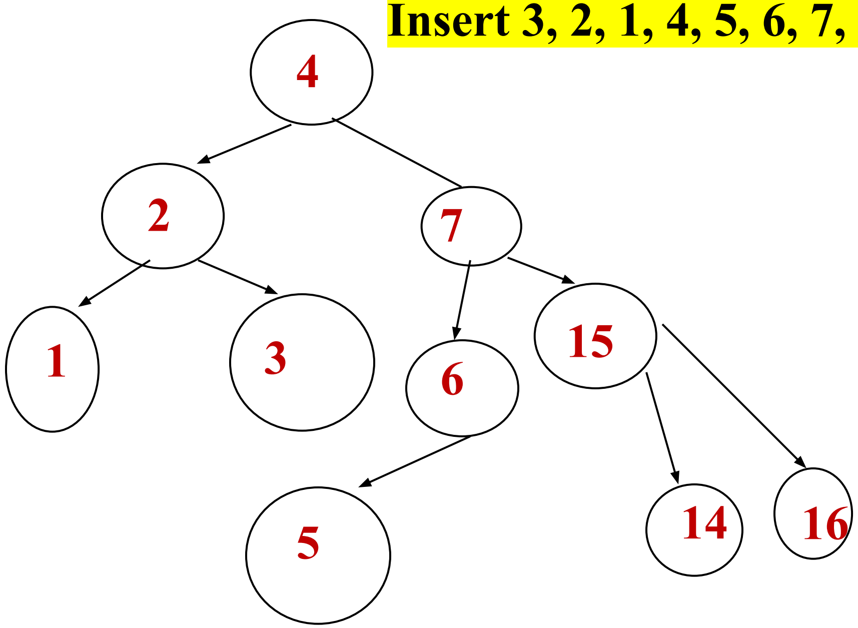


Fig 16

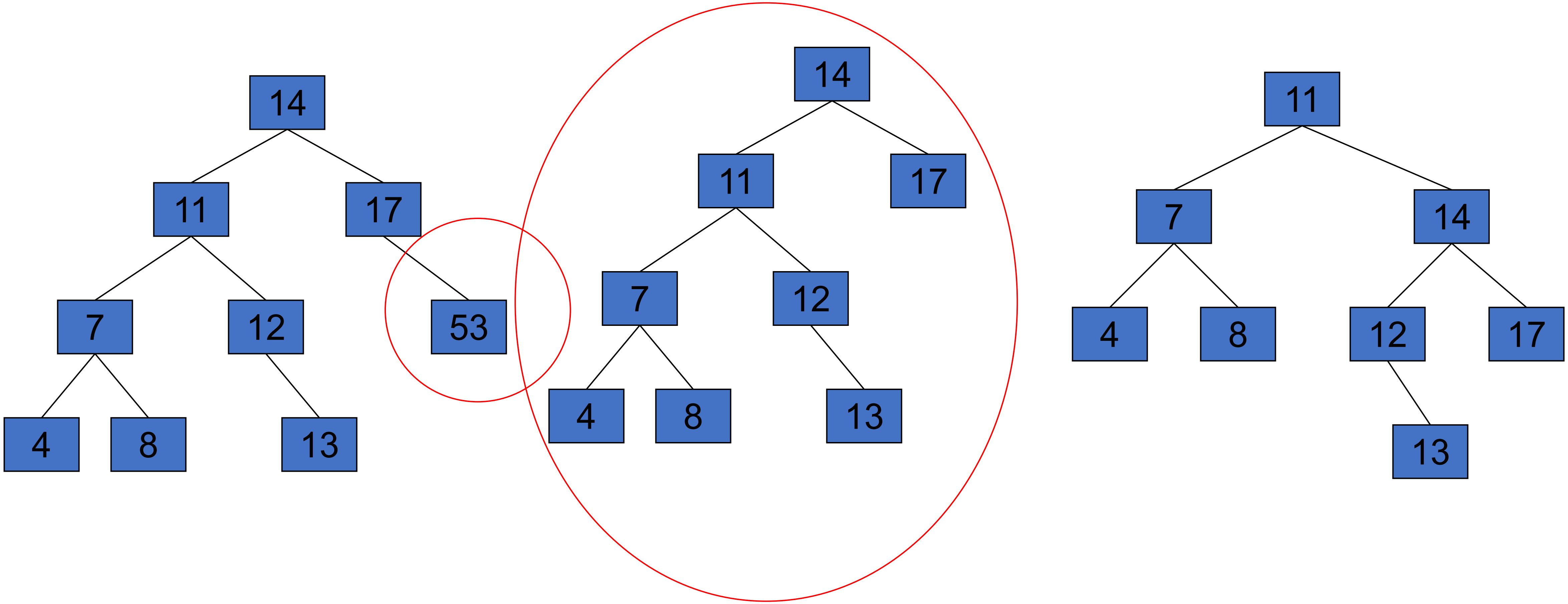
- **Deletions also can be done with similar rotations**

Example of Deletion Operation from below AVL Tree

Now remove 53

Now remove 53, unbalanced

Balanced!



Applications

Databases:

- Indexing
- Query optimization

Memory Management:

- Dynamic memory allocation

File Systems:

- Directory management
- Metadata management

Networking:

- Routing tables
- Prefix matching

Compilers:

- Syntax trees
- Symbol tables

Gaming:

- Collision detection
- Leaderboard management

Geographic Information Systems (GIS):

- Spatial data indexing
- Map data management

Cryptography:

- Digital certificates

Artificial Intelligence:

- Decision trees
- Search algorithms

Financial Systems:

- Transaction management
- Order matching

Thank You !!