ADVANCED DATA STRUCTURES & ALGORITHMS

Splay Trees

Dr. G.Kalyani
Department of Information Technology
VR Siddhartha Engineering College

Introduction to splay trees

- **Definition**: Splay trees are **self-adjusting binary search trees that move recently accessed elements to the root**.

- **Key Properties**:
  - **Self-Adjusting**: restructure themselves based on access patterns.
  - **Amortized Efficiency**: Despite potentially expensive individual operations, the average time per operation is logarithmic over a sequence of operations.

- **Applications**: Useful in scenarios where certain elements are accessed more frequently (e.g., caches, search engines).

What is a Bottom-Up Splay Tree?

- **Definition**:
  - A variant of splay trees that **performs splaying in a bottom-up manner**, as opposed to top-down.

- **Characteristics**:
  - Operates on a subtree formed after the operations (insert/delete/search).
  - All rotations occur within the tree to propagate the recent element accessed to the root.

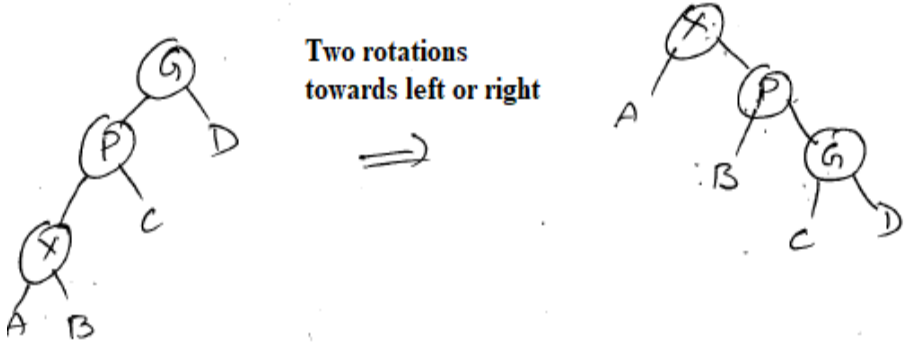
Basic Operations Overview

- **Splay Operation**:
  - Brief introduction to how nodes are moved to the root after an access.
- **Search**:
  - Finding a node and moving it to the root through splaying.
- **Insert**:
  - Adding a new node while maintaining binary search tree properties.
- **Delete**:
  - Removing a node and adjusting the tree structure.

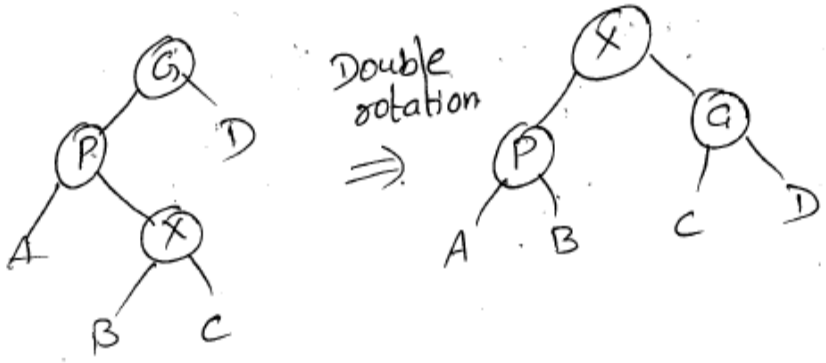
Splay Operation

- **Process:** When accessing a node, perform a series of rotations to bring it to the root.
  - **Zig:** Let X be node on the access path we are rotating. If the parent (P) of X is the root, we rotate X and root.
  - **If X has both parent (P) and grand parent (G).** There are two cases

Zig-Zig: Double rotation if the node and its parent are both left or right children.



Zig-Zag: Double rotation if the node is a right child and its parent is a left child or vice versa.

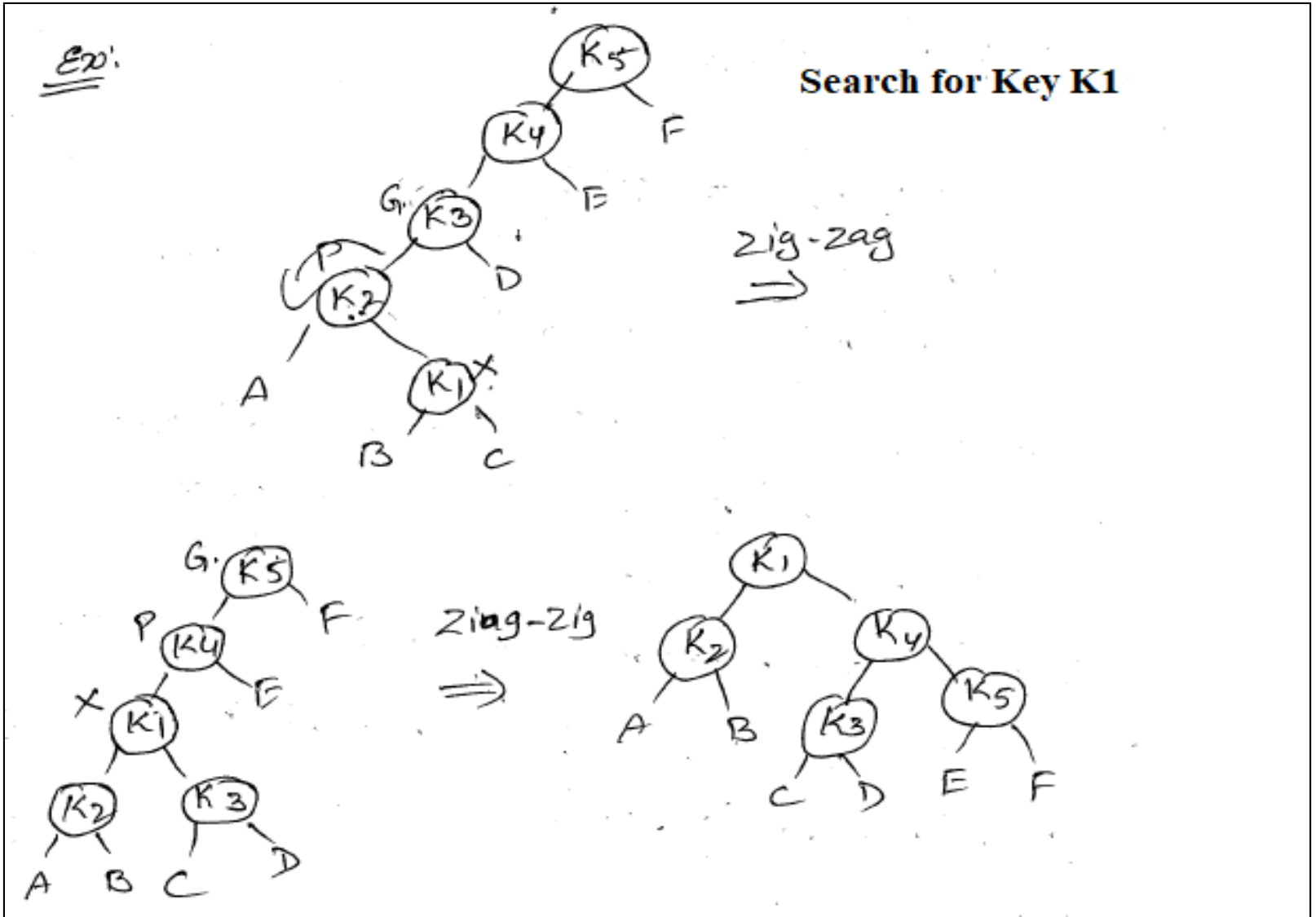


Searching in Bottom-Up Splay Trees

- **Process**:
  - Search for the target element X.
  - **If found** → Splay the X to the root.
  - **If not-found** → splay the last element accessed in the search process to the root.

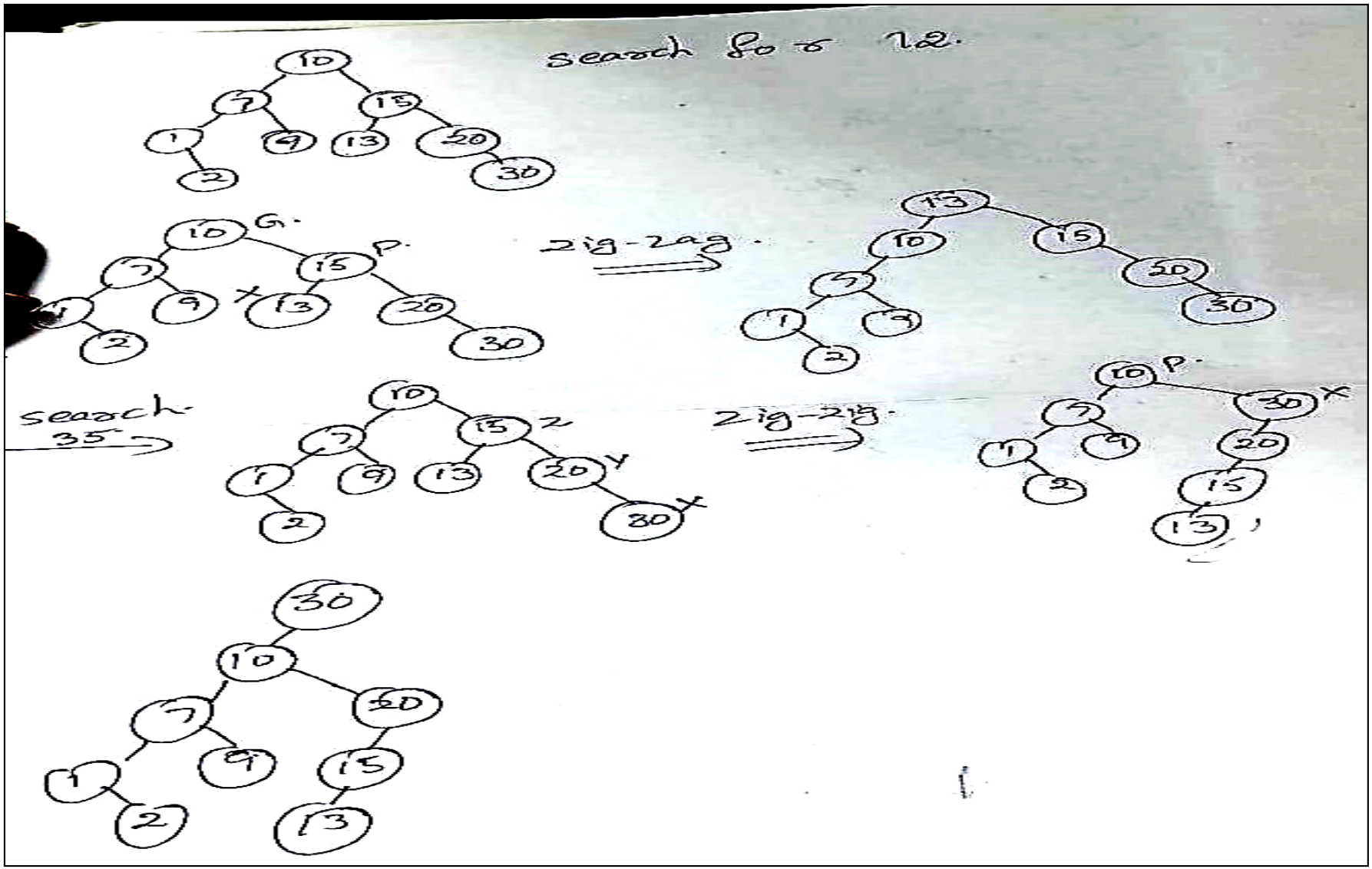
Example for Search in Bottom-Up Splay Trees

Found Case



Example for Search in Bottom-Up Splay Trees

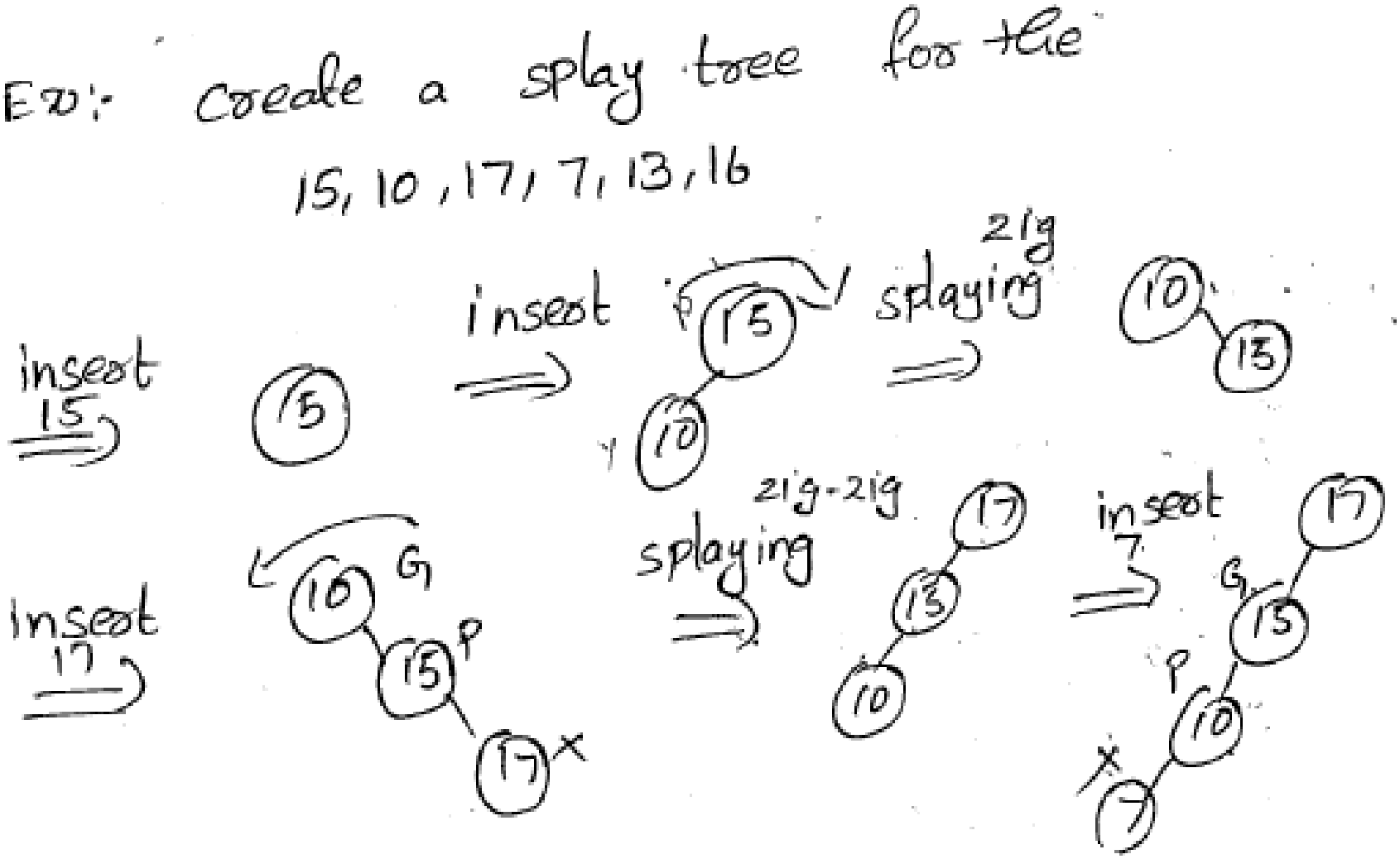
Not-Found Case



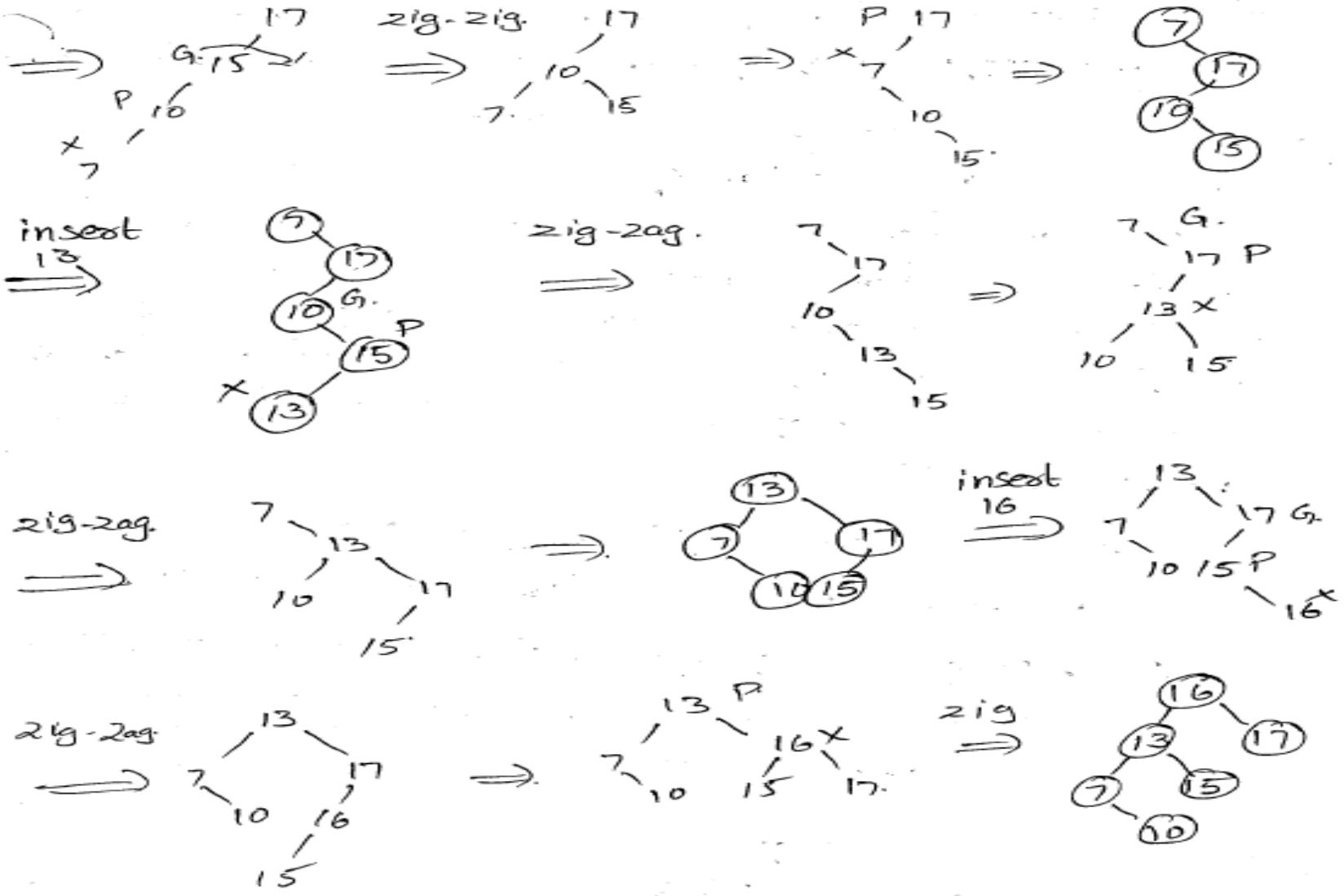
Insertion in Bottom-Up Splay Trees

- **Process**:
  - Insert the node like in a regular binary search tree.
  - Splay the newly inserted node to the root.

Example to Create a Bottom-Up Splay Tree



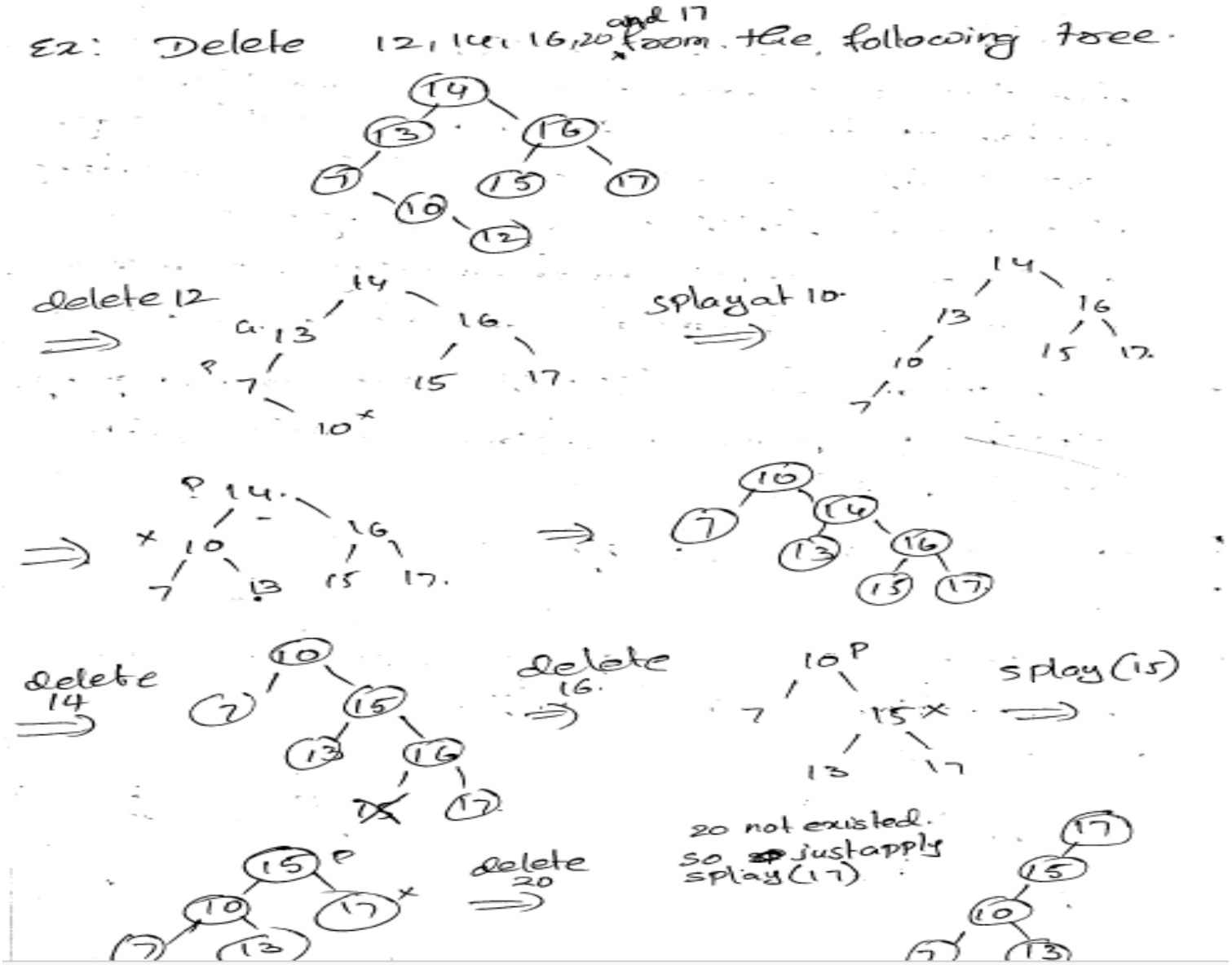
Example to Create a Bottom-Up Splay Tree



Example for Deletion in Bottom-Up Splay Trees

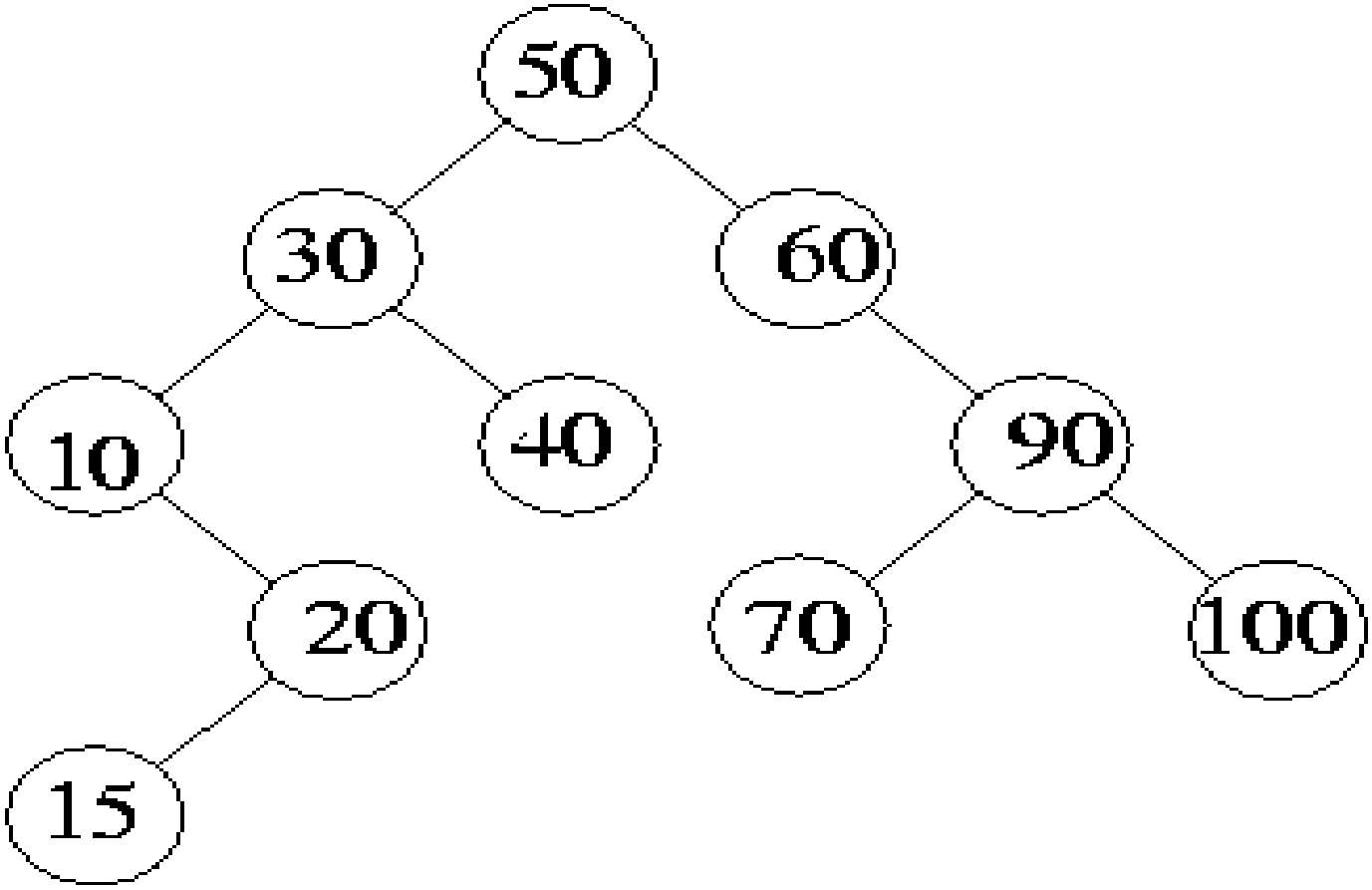
- **Process**:
  - Search for the node to be deleted
  - Delete the node
  - Apply splaying with respect to the parent of the deleted node.

Deletion in Bottom-Up Splay Trees



Deletion in Bottom-Up Splay Trees

Delete 15, 40, 70, 10, 30 and 90



Drawback in Bottom-Up Splay Trees

- Bottom-up splaying strategy requires a traverse from the root down the tree and then a bottom-up traversal to implement the splaying step.
- This can be done either by maintain the parent links or by storing the access path on a stack unfortunately both methods require a substantial amount of overhead.
- A solution for this is Top-down splaying, which performs rotations on the initial access paths in top-down.

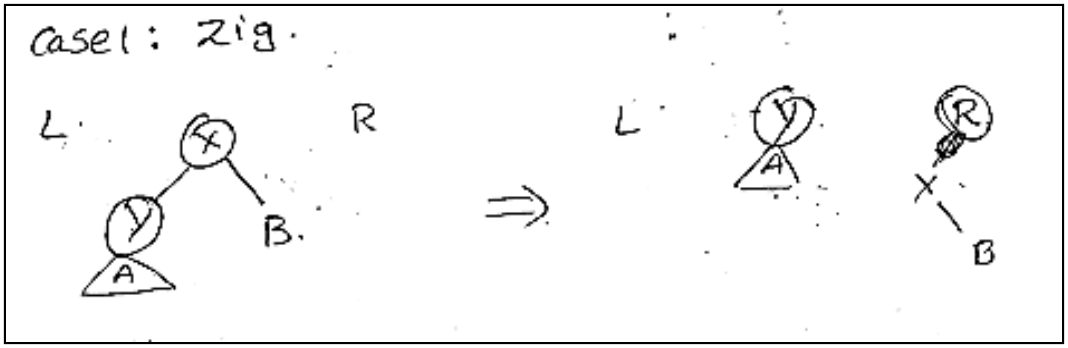
Top-Down Splay Trees

- In top-down splay trees, the **splay operation is performed while descending the tree from the root to the accessed node**, which differs from the more traditional bottom-up approach where restructuring occurs on the way back up.

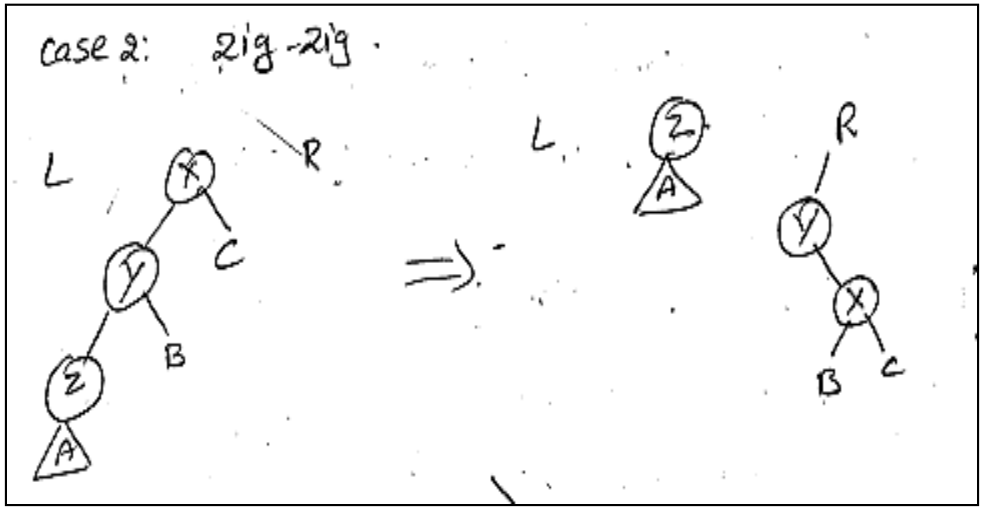
- **Splay Operation during Descent:**
  - The **tree is divided into three parts during the splay operation: a left tree, a right tree, and a middle tree** that contains the current node.
  - The left and right trees are constructed dynamically as the tree is traversed, based on comparisons made at each step.

Splay Operation

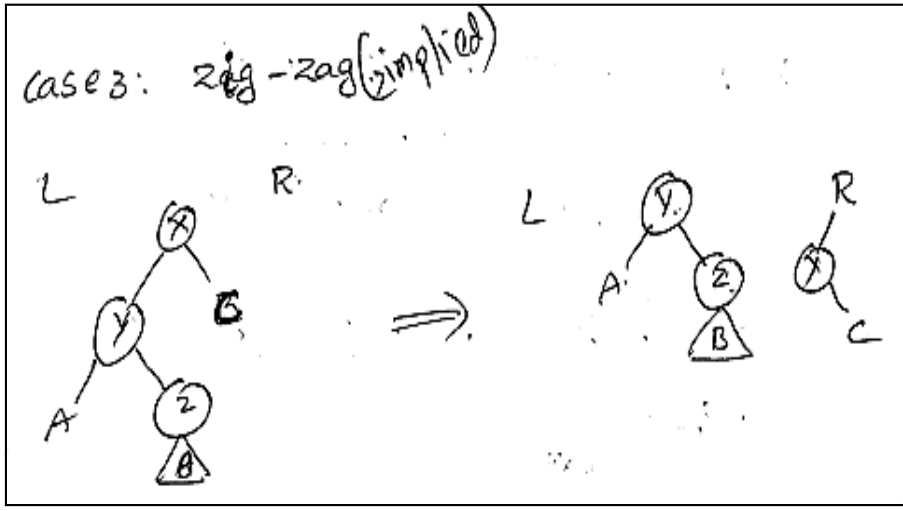
- In top-down splaying, three cases of rotations are applied
- **Zig:**



Zig-Zig:

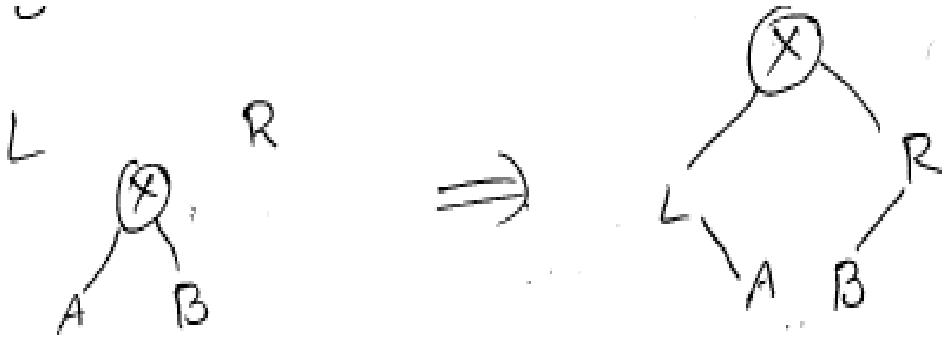


Zig-Zag:



Splay Operation

- once we performed the final splaying step also(till the required element is the root), a final arrangement should be done to give the resultant tree.
- **Merge:**

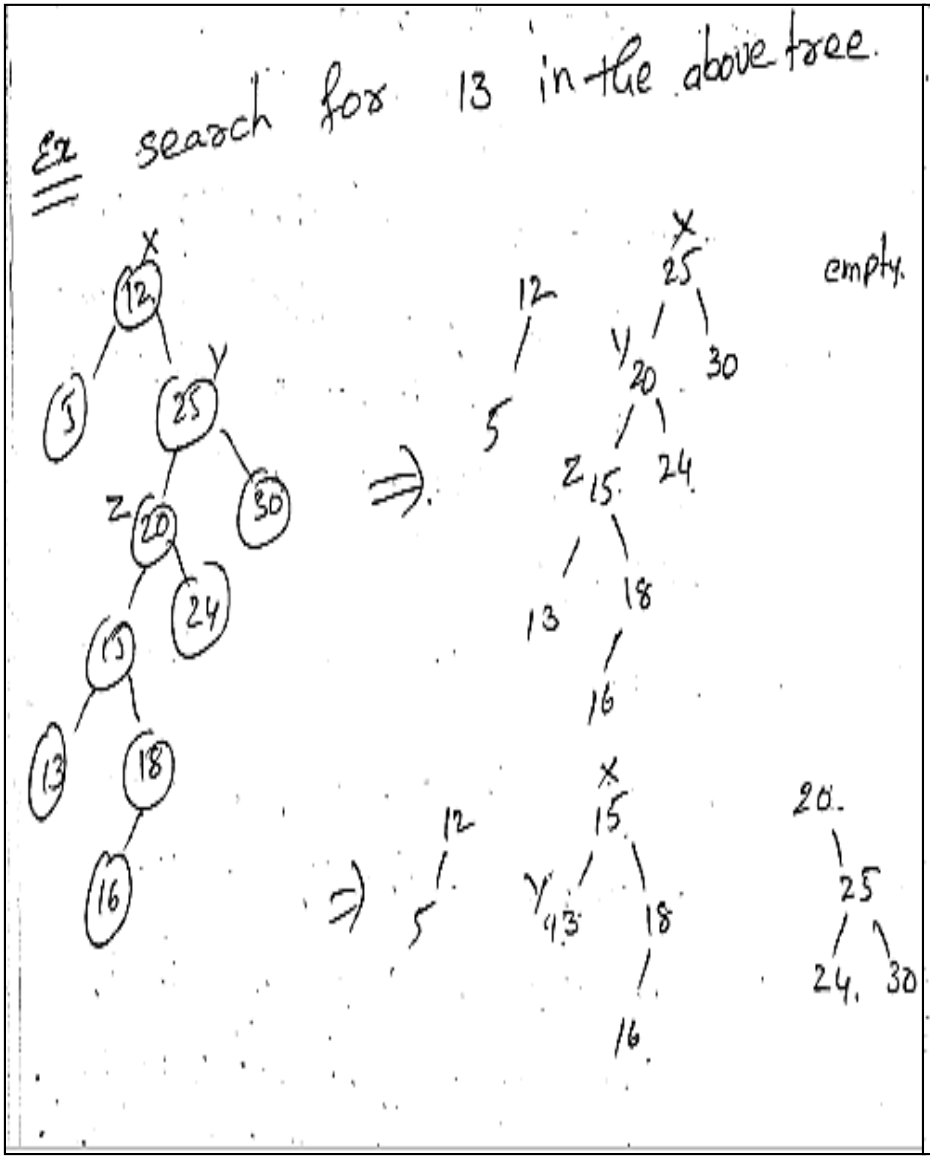


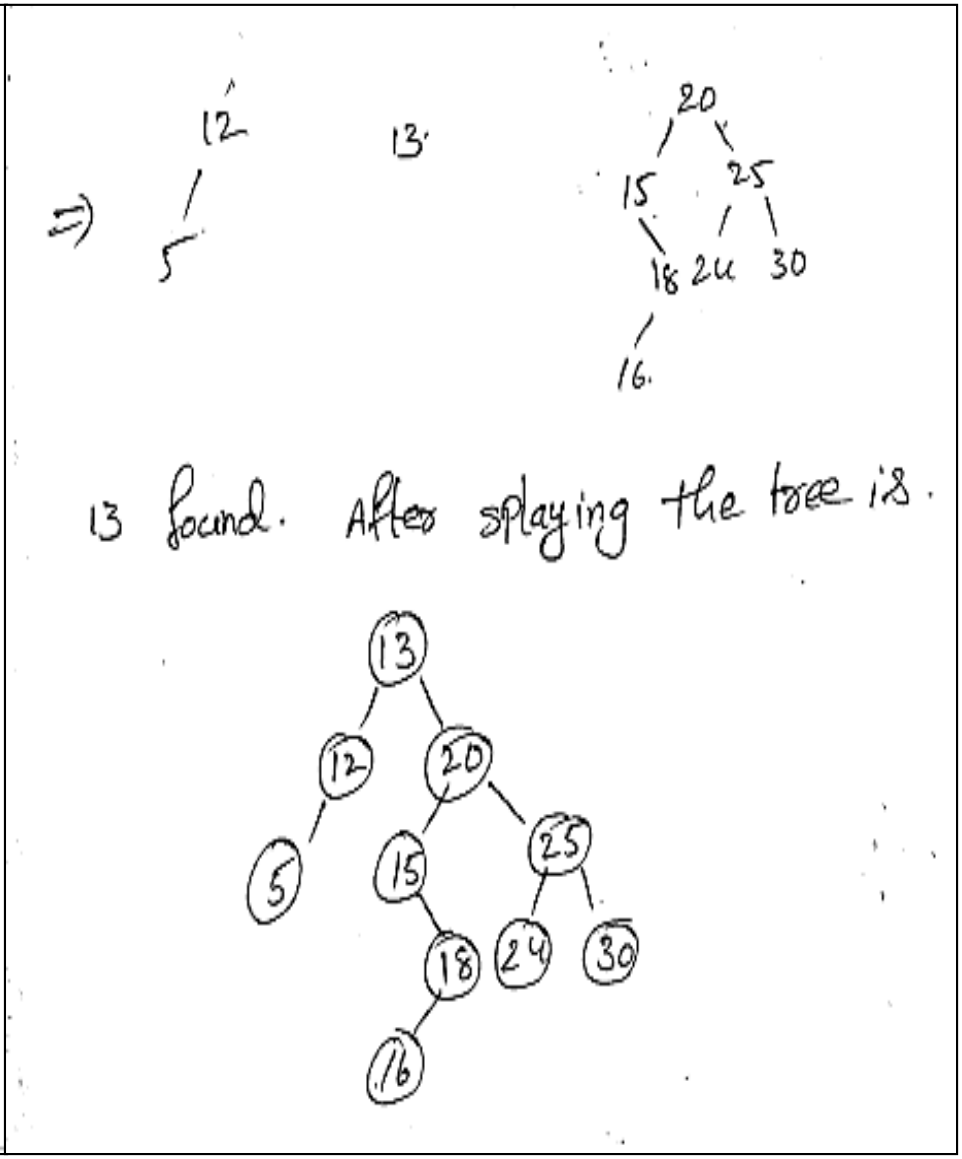
Searching in Top-Down Splay Trees

- **For successful Search**, the search element becomes the root of the tree.
- **For unsuccessful search**, the node where the search ends will become the root of the tree.

Example for Search in Top-Down Splay Trees

Found Case

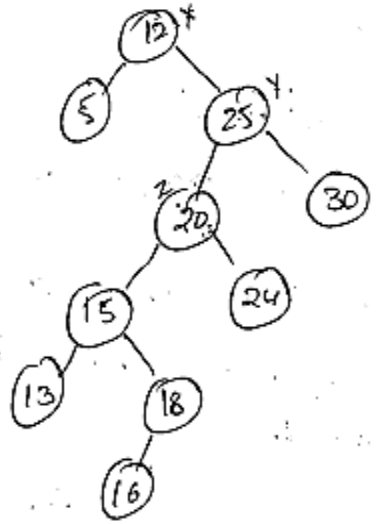




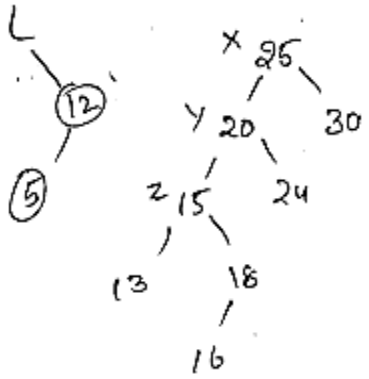
Example for Search in Top-Down Splay Trees

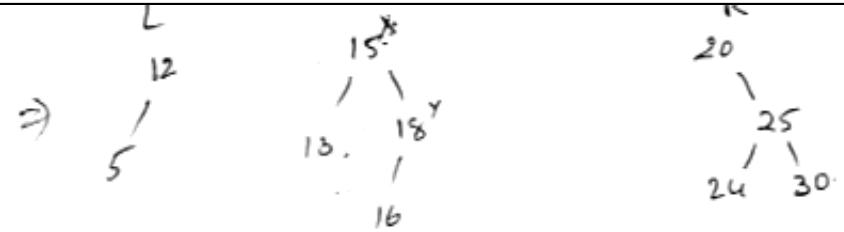
Not- Found Case

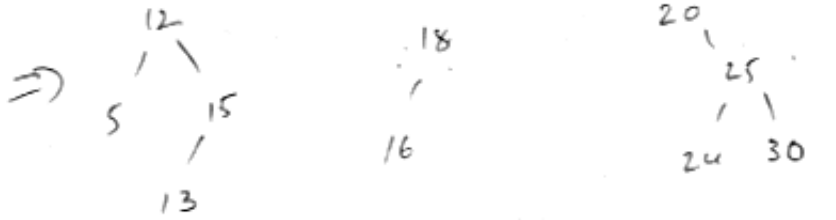
Ex: search for 19 in the following splay tree.



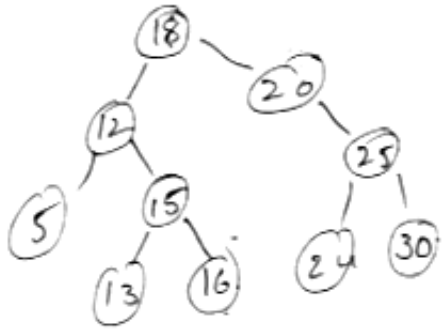
Soln:







Finally



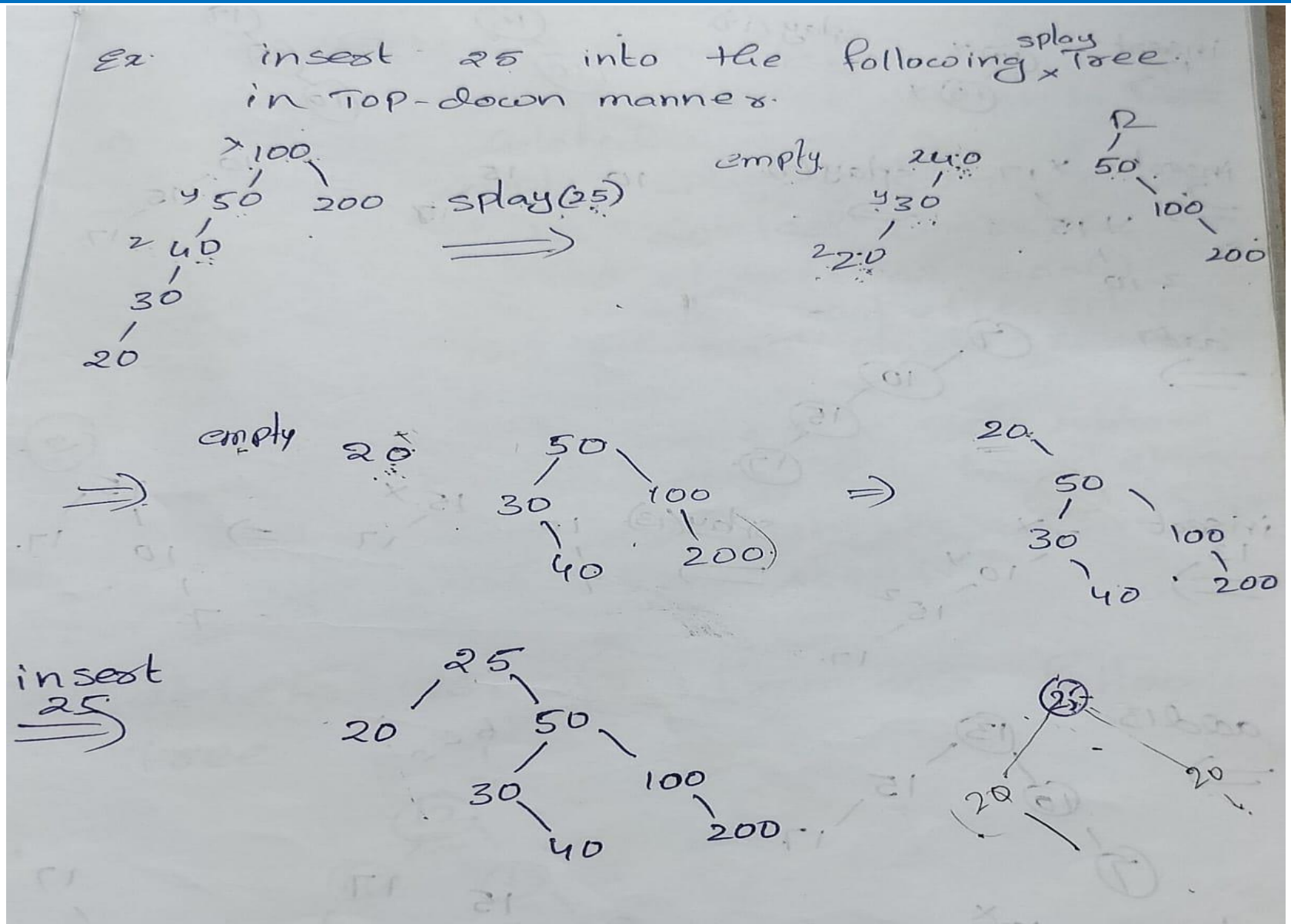
because of root is 18, element 19 not found

Insertion in Top-Down Splay Trees

Process: Let K be the key to be inserted

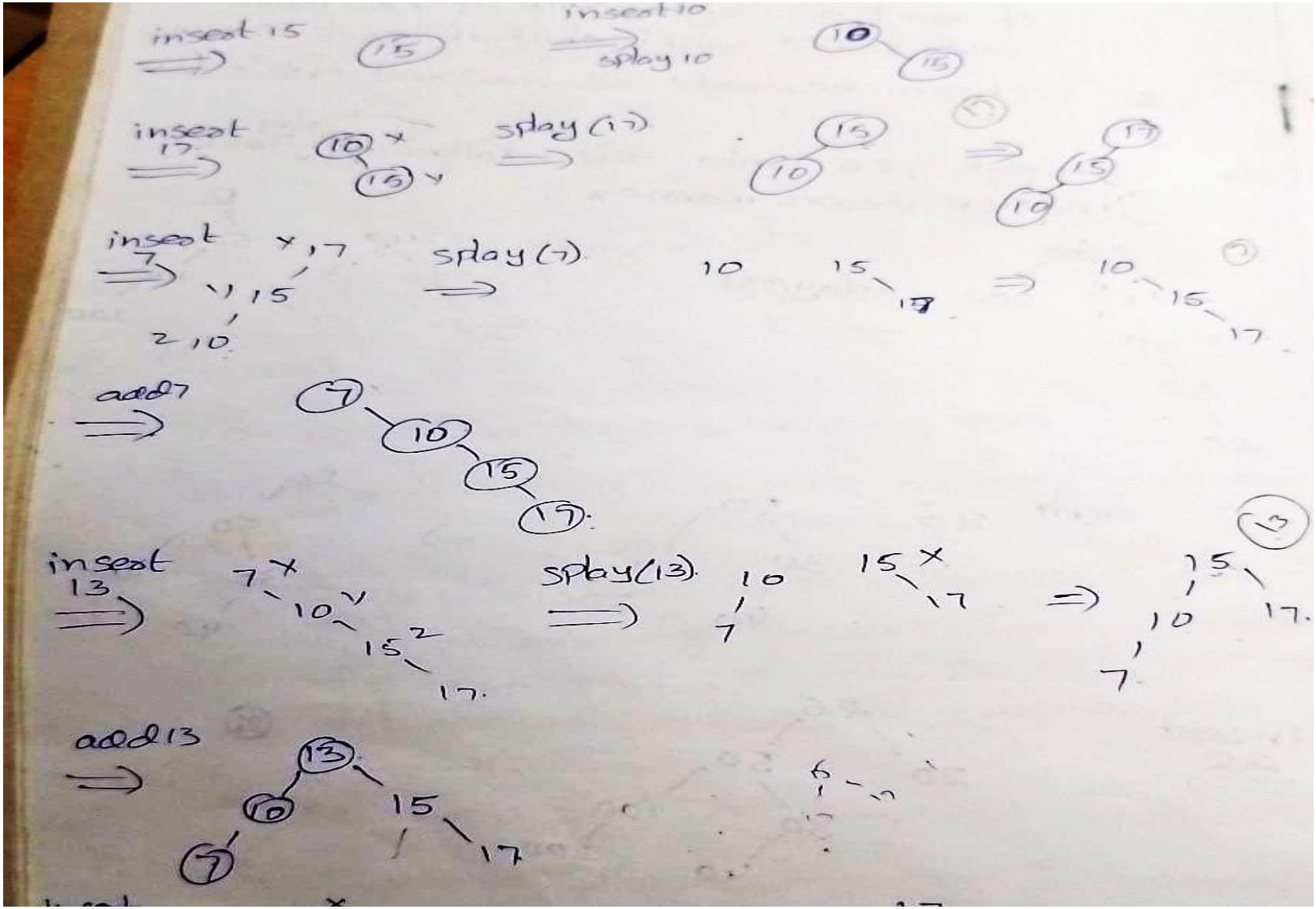
- **Case-1: If Root is null:** Create a new node and insert K.
- **Case-2: if root is not null:**
  - **Splay the tree with the given key access path.**
    - If K is already present in the tree, then k becomes the root of the tree after all splaying. If K is not present then the last accessed leaf node becomes the new root after splaying.
  - **If new root is same as K, the exit.**
  - **Otherwise create a new node with K and compare the root of the tree(after splaying) with K.**
  - **If K is smaller than root:** make root as the right child of K, and make the left child of root as the left child of K.
  - **If K is greater than root:** make root as the left child of K, and make the right child of root as the right child of K.
- **Return Node with K as the new root of the tree (after insertion).**

Example for insert in Top-Down Splay Tree



Example to Create a Top-Down Splay Tree

Create a top down splay tree with keys 15,10,17,13,14,16,2,12



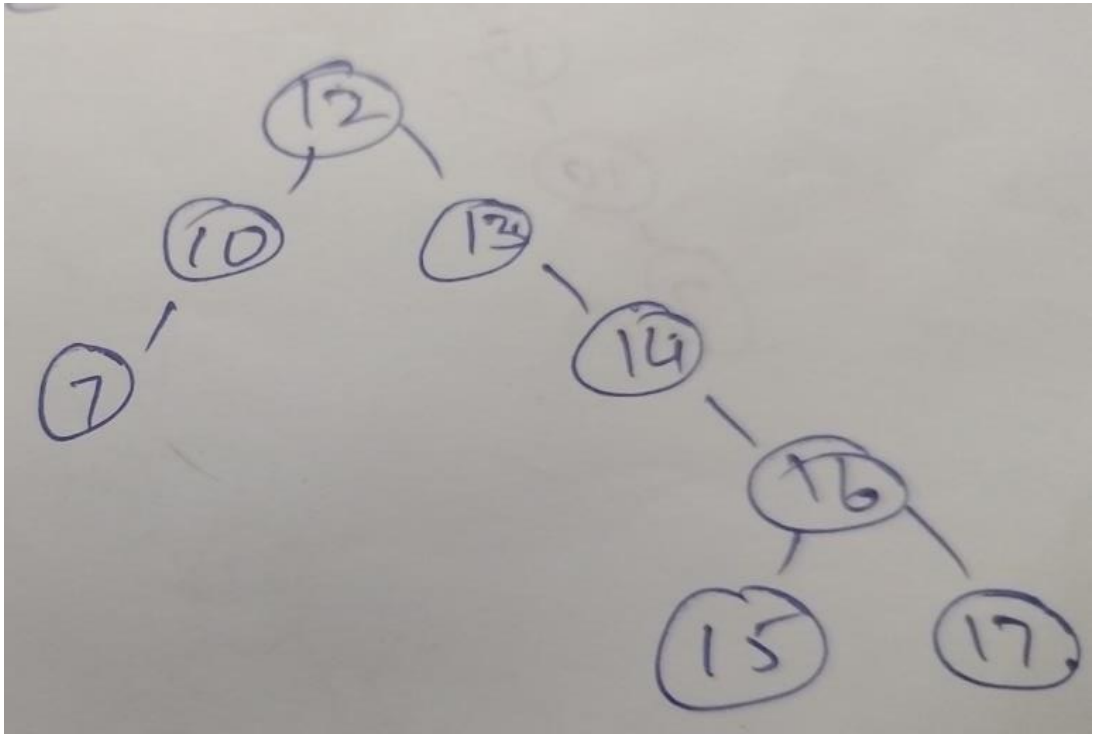
Example for Deletion in Top-Down Splay Trees

Process:

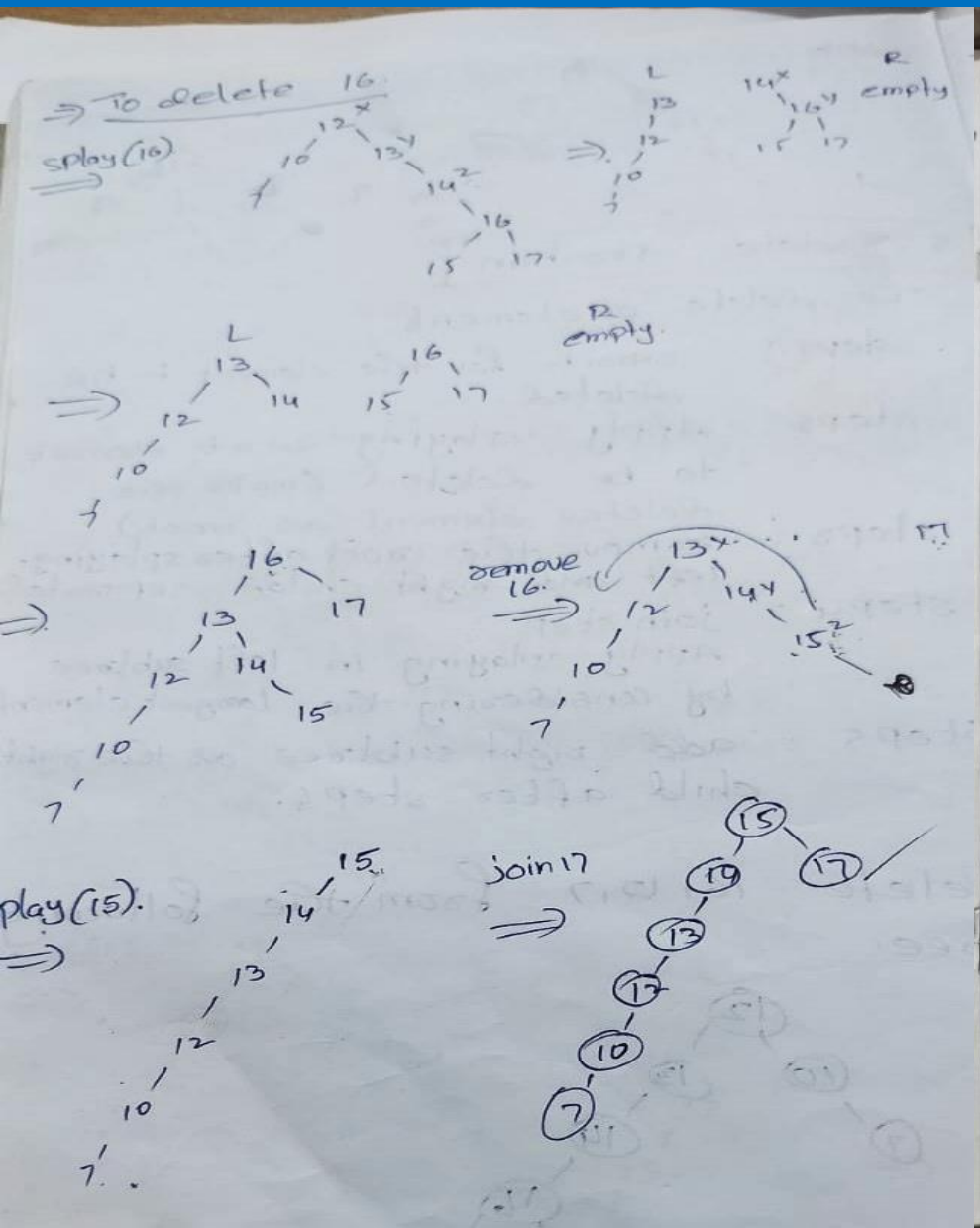
- 1. Apply the splaying with the key to be deleted.
  - If element is there , it becomes root of the tree.
  - Otherwise element not found.
- Remove the root after splaying. (left and right subtrees gets separated)
- Join step: Apply splaying in the left subtree by considering the largest element. Attach the right subtree as the right child to the root after splaying.

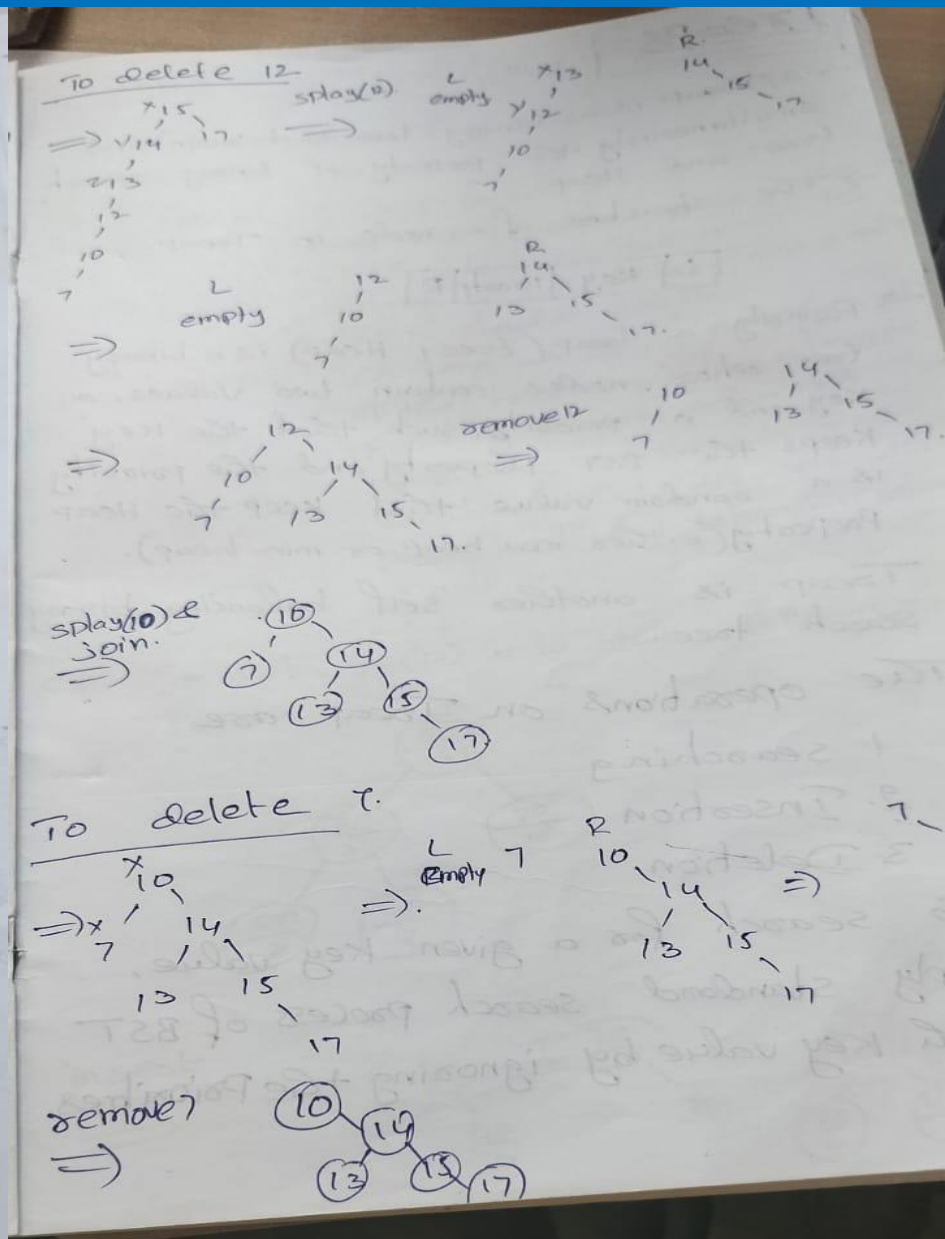
Deletion in Top-Down Splay Trees

Delete 16,12, and 7 from the following tree



Deletion in Top-Down Splay Trees





Summary

- Definition of Splay Tree
- Bottom-up Vs Top-Down Splay Trees
- Bottom-Up Splay Trees
  - Splaying Rotations (Zig, Zig-Zig, Zig-Zag)
  - Operations
    - Search
    - Insertion
    - Deletion
- Top-Down Splay Trees
  - Splaying Rotations (Zig, Zig-Zig, simplified Zig-Zag)
  - Operations
    - Search
    - Insertion
    - Deletion

"నీ విజయానికి అడ్డుకునేది...
నీలోని ప్రతికూల ఆలోచనలే..
క్రింద పడ్డామని ప్రయత్నం ఆపితే
చేసే పనిలో ఎన్నటికీ
విజయం సాధించలేము.."

అబ్దుల్ కలాం