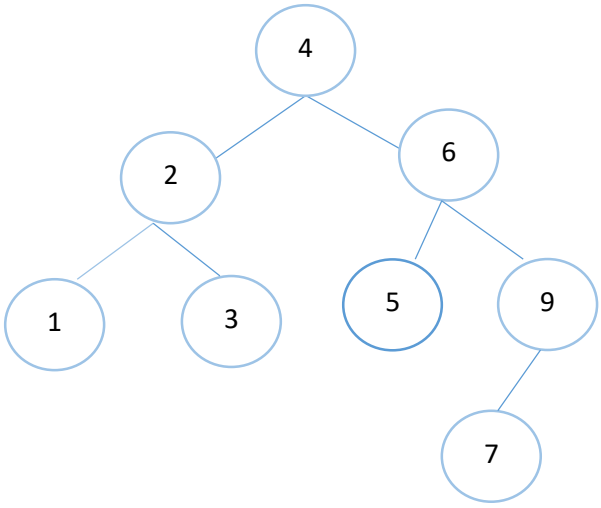
Splay Tree

Reg.No:
UWU/CST/16/048

Splay Tree



- Splay Tree is a self-adjusted Binary Search Tree in which every operation on element rearranges the tree so that the element is placed at the root position of the tree.

- When an element is accessed in a splay tree, tree rotations are used to move it to the top of the tree.

- In splay tree, every operation is performed at root of the tree. All the operations in splay tree are involved with a common operation called **"Splaying"**.

- Splaying an element is the process of bringing it to the root position by performing suitable rotation operations.

- By splaying elements we bring more frequently used elements closer to the root of the tree so that any operation on those elements is performed quickly. That means the splaying operation automatically brings more frequently used elements closer to the root of the tree.

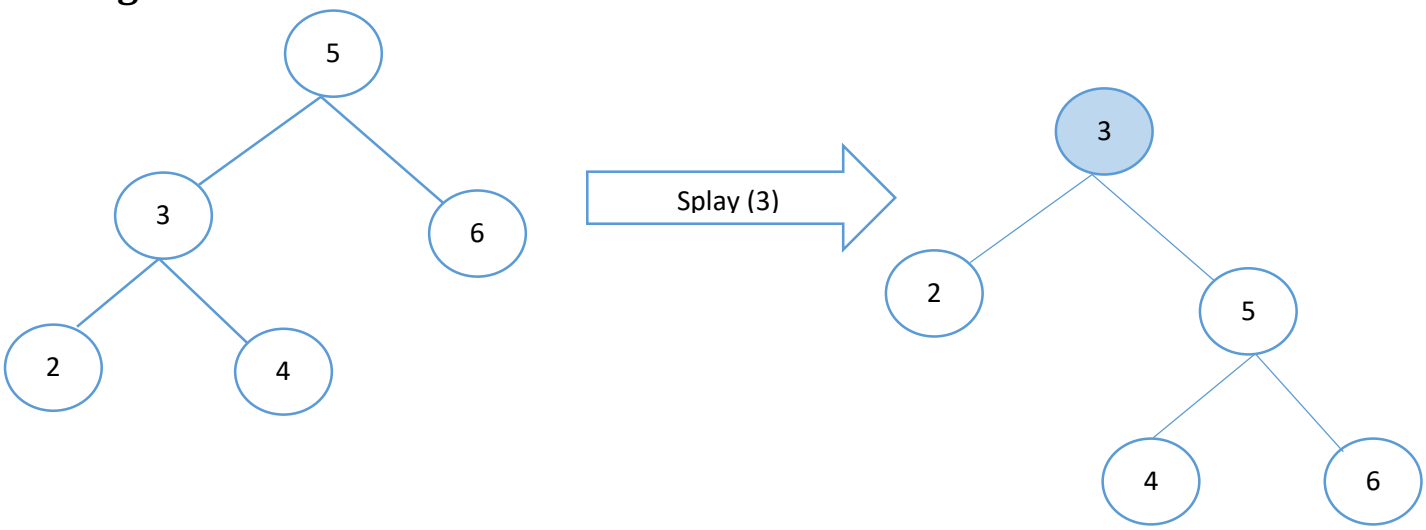
Every operation on splay tree performs the splaying operation. For example, the insertion operation first inserts the new element using the binary search tree insertion process, then the newly inserted element is splayed so that it is placed at root of the tree. The search operation in splay tree is nothing but searching the element using binary search process and then splaying that searched element so that it is placed at the root of the tree.

Algorithm	**Average**	**Worst case**
Space	O(n)	O(n)
Search	O(log n)	amortized O(log n)
Insert	O(log n)	amortized O(log n)
Delete	O(log n)	amortized O(log n)

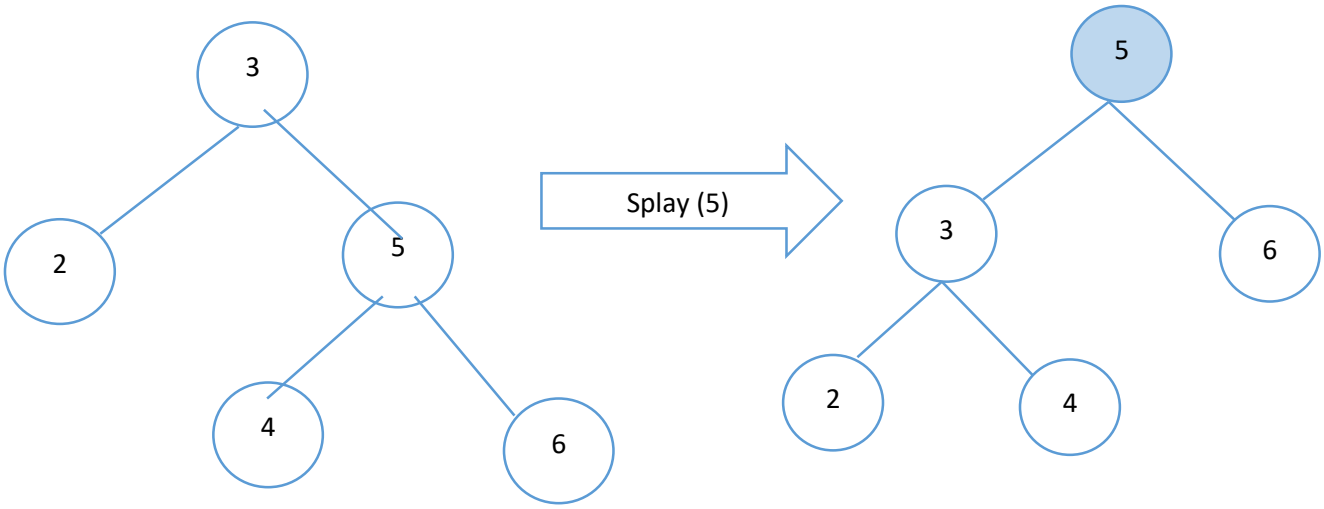
Splay Tree Rotations

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

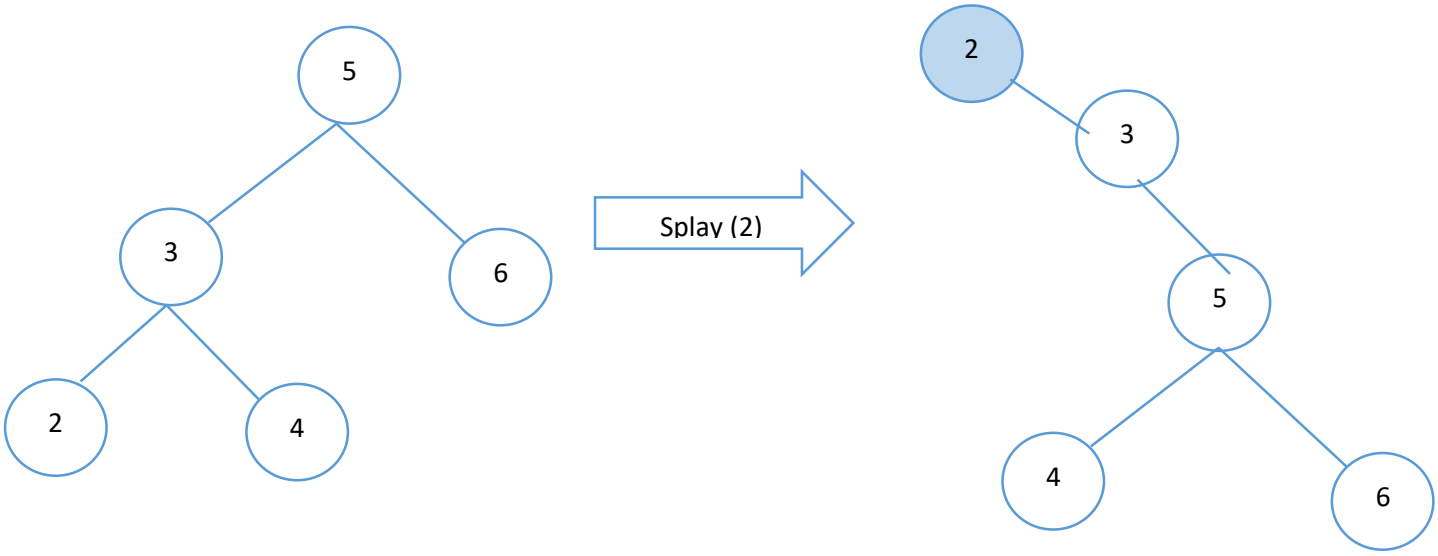
Zig Rotation



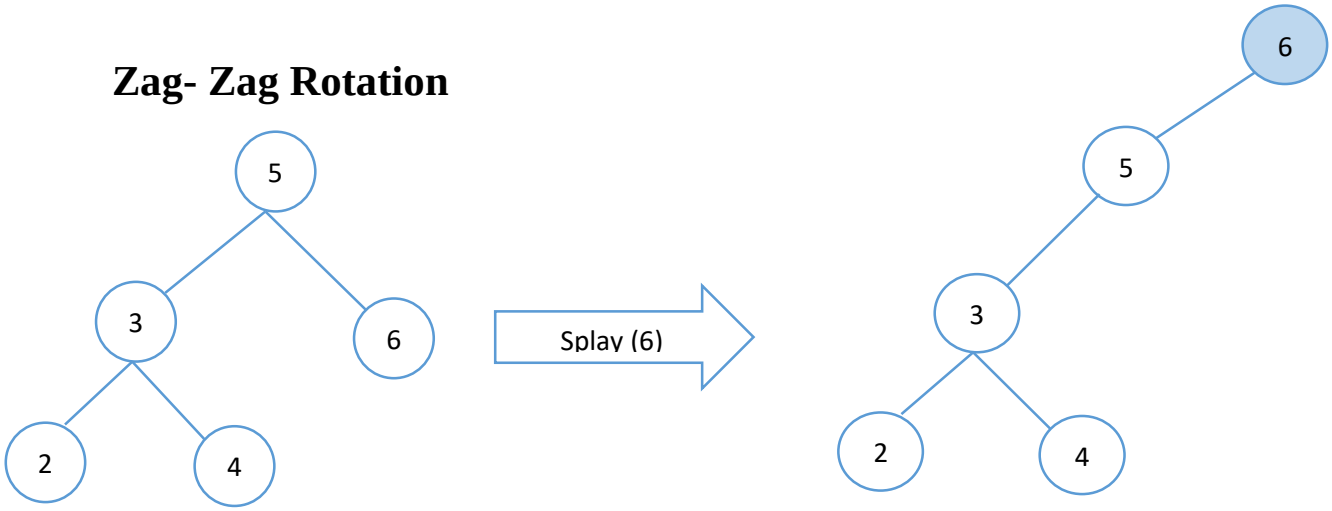
Zag Rotation



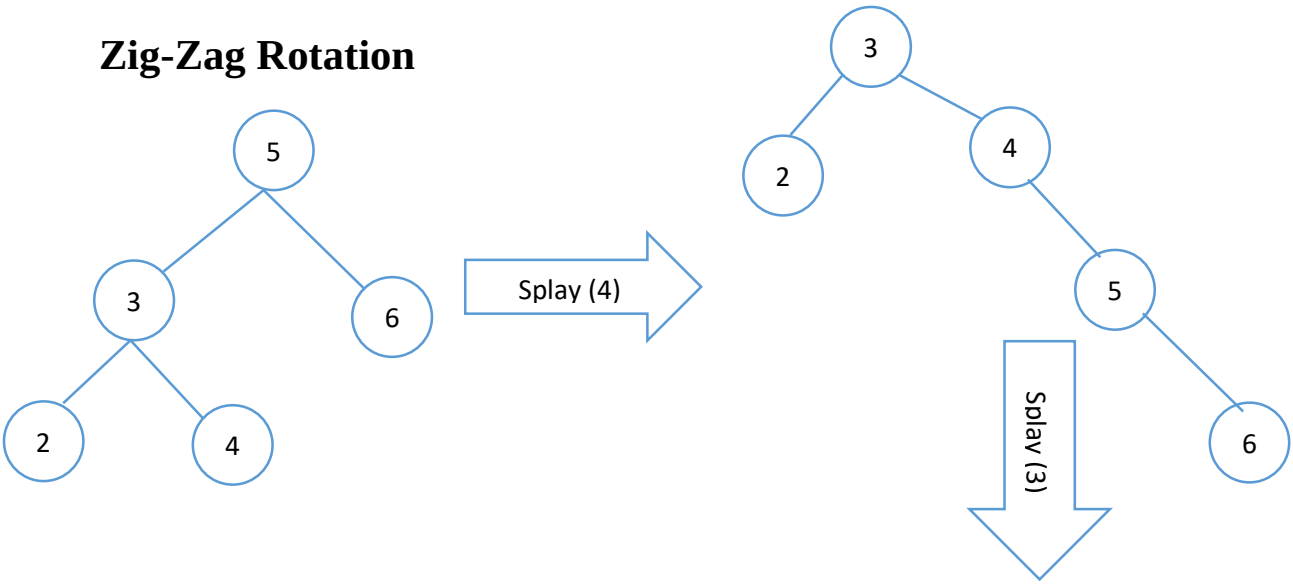
Zig-Zig Rotation



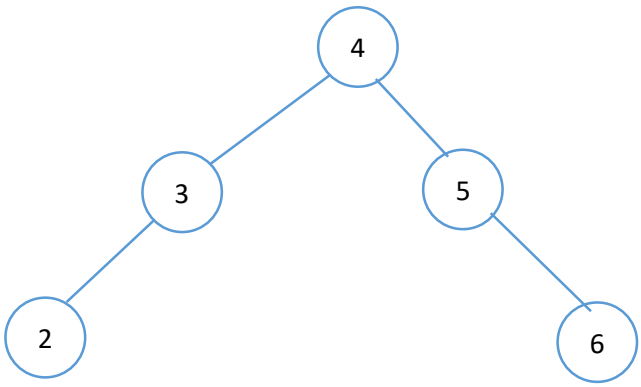
Zag- Zag Rotation



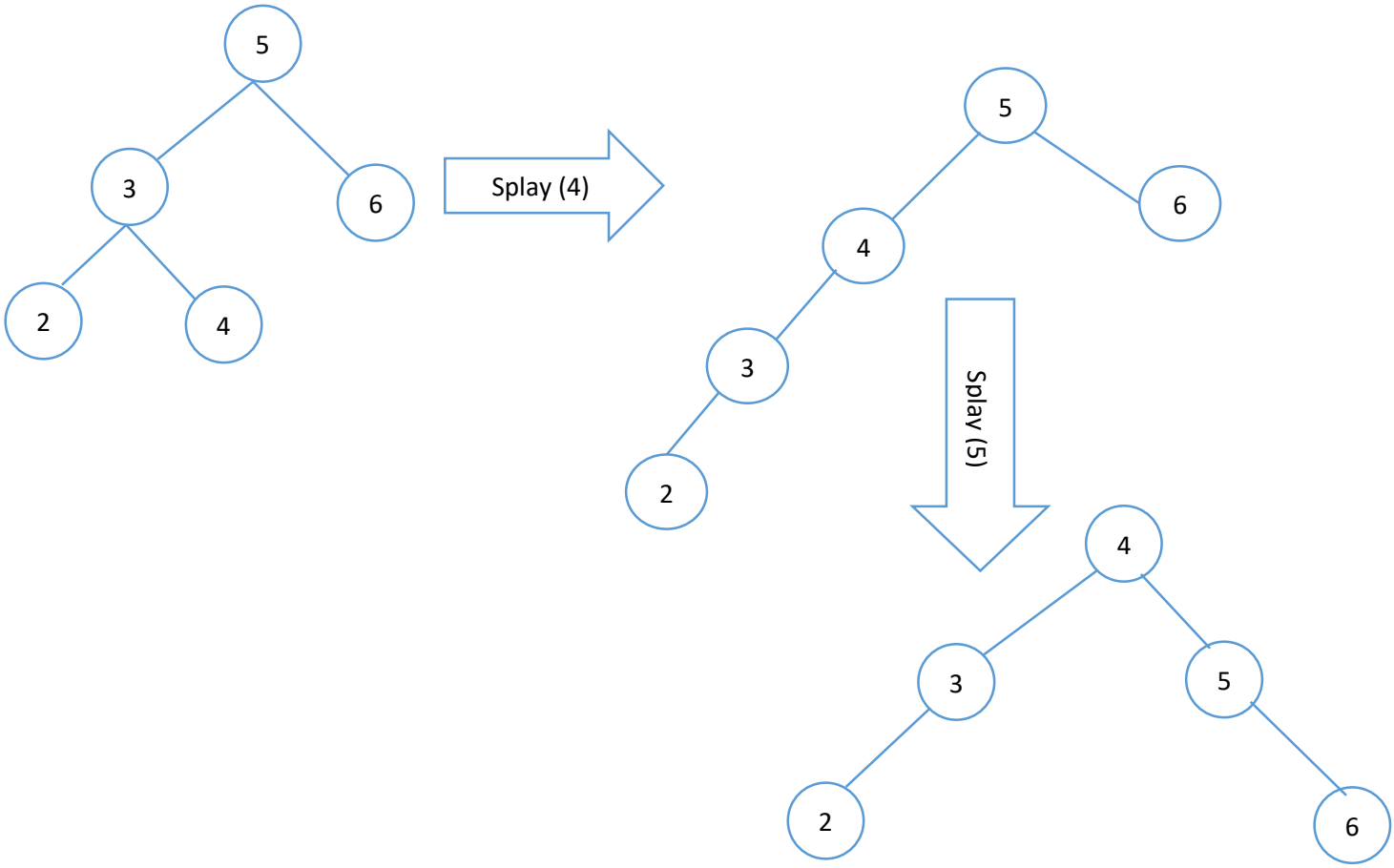
Zig-Zag Rotation



.



Zag –Zig Rotation

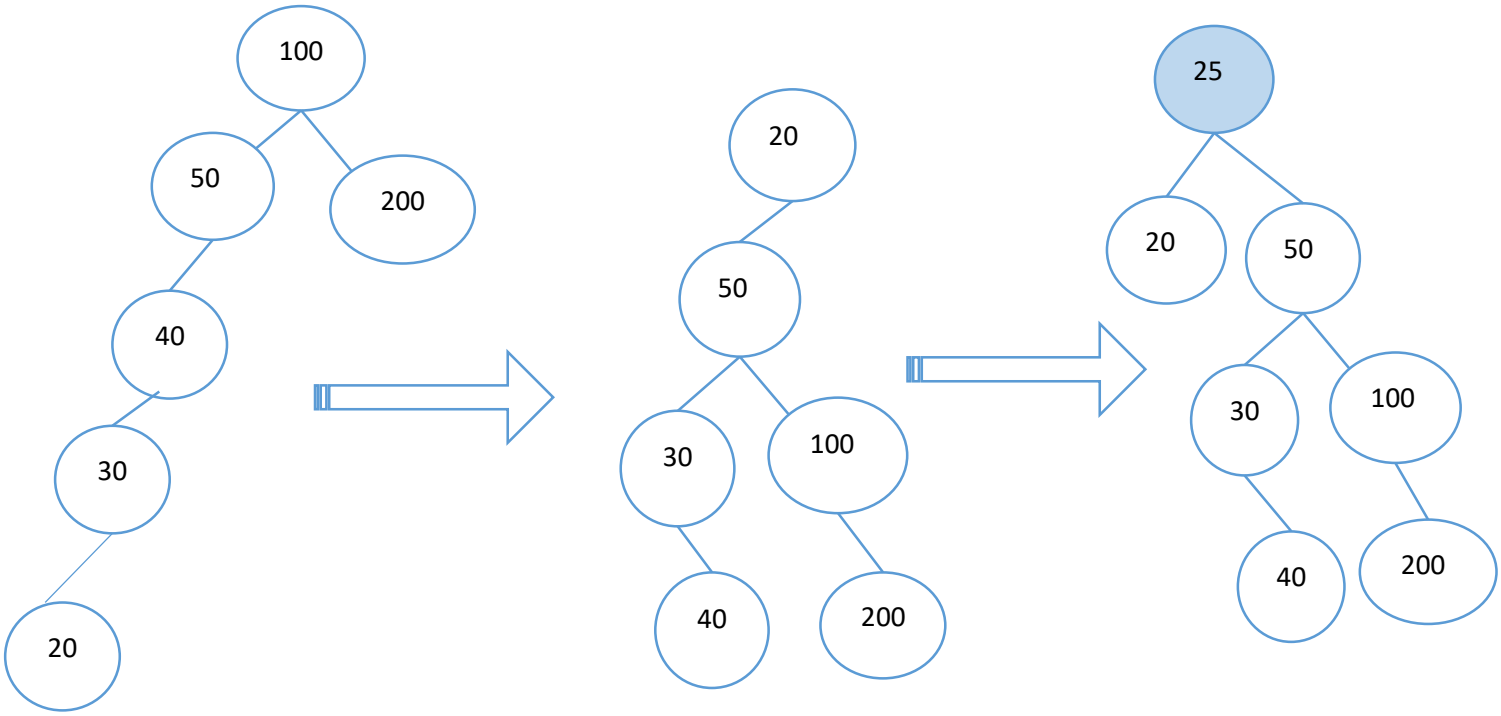


Splay Tree Insert

Following are different cases to insert a key k in splay tree

- Root is NULL: We simply allocate a new node and return it as root.
- Splay the given key k. If k is already present, then it becomes the new root. If not present, then last accessed leaf node becomes the new root.
- If new root's key is same as k, don't do anything as k is already present.
- Else allocate memory for new node and compare root's key with k.
- If k is smaller than root's key, make root as right child of new node, copy left child of root as left child of new node and make left child of root as NULL.
- If k is greater than root's key, make root as left child of new node, copy right child of root as right child of new node and make right child of root as NULL.
- Return new node as new root of tree.

Example: insert (25)

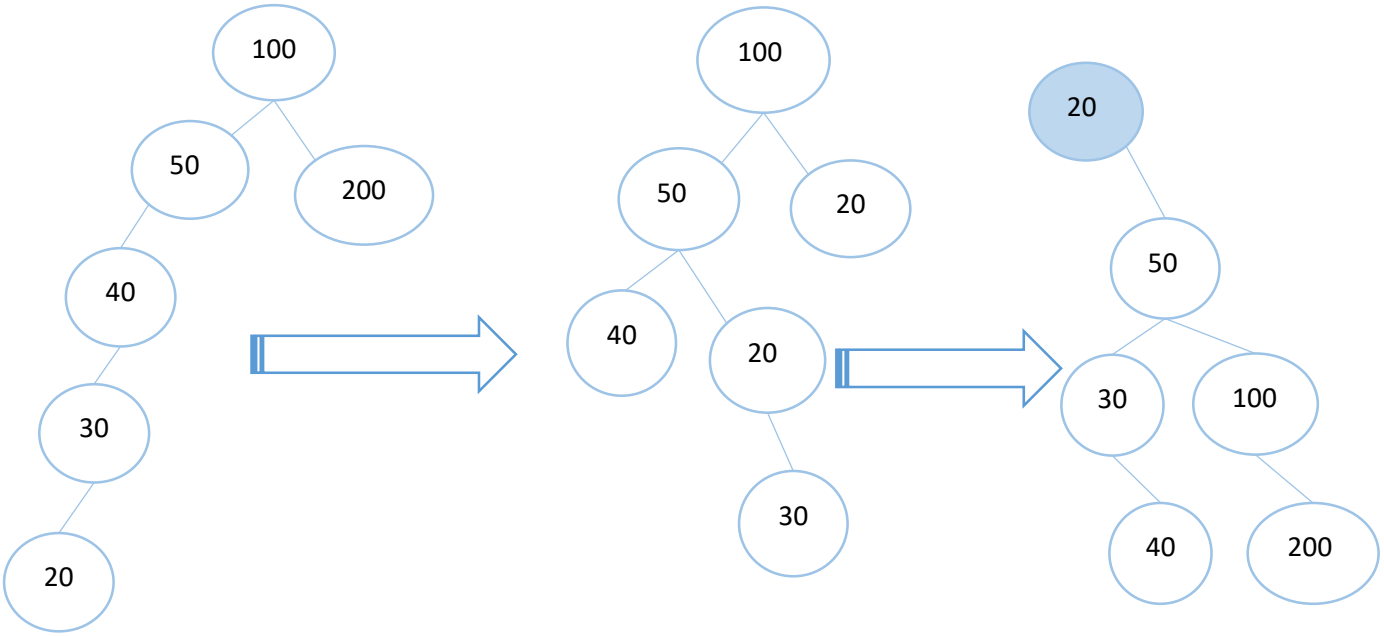


Splay Tree Search

The search operation in Splay tree does the standard BST search, in addition to search, it also splays

- If the search is successful, then the node that is found is splayed and becomes the new root.
- Else the last node accessed prior to reaching the NULL is splayed and becomes the new root.

Example: Search (20)



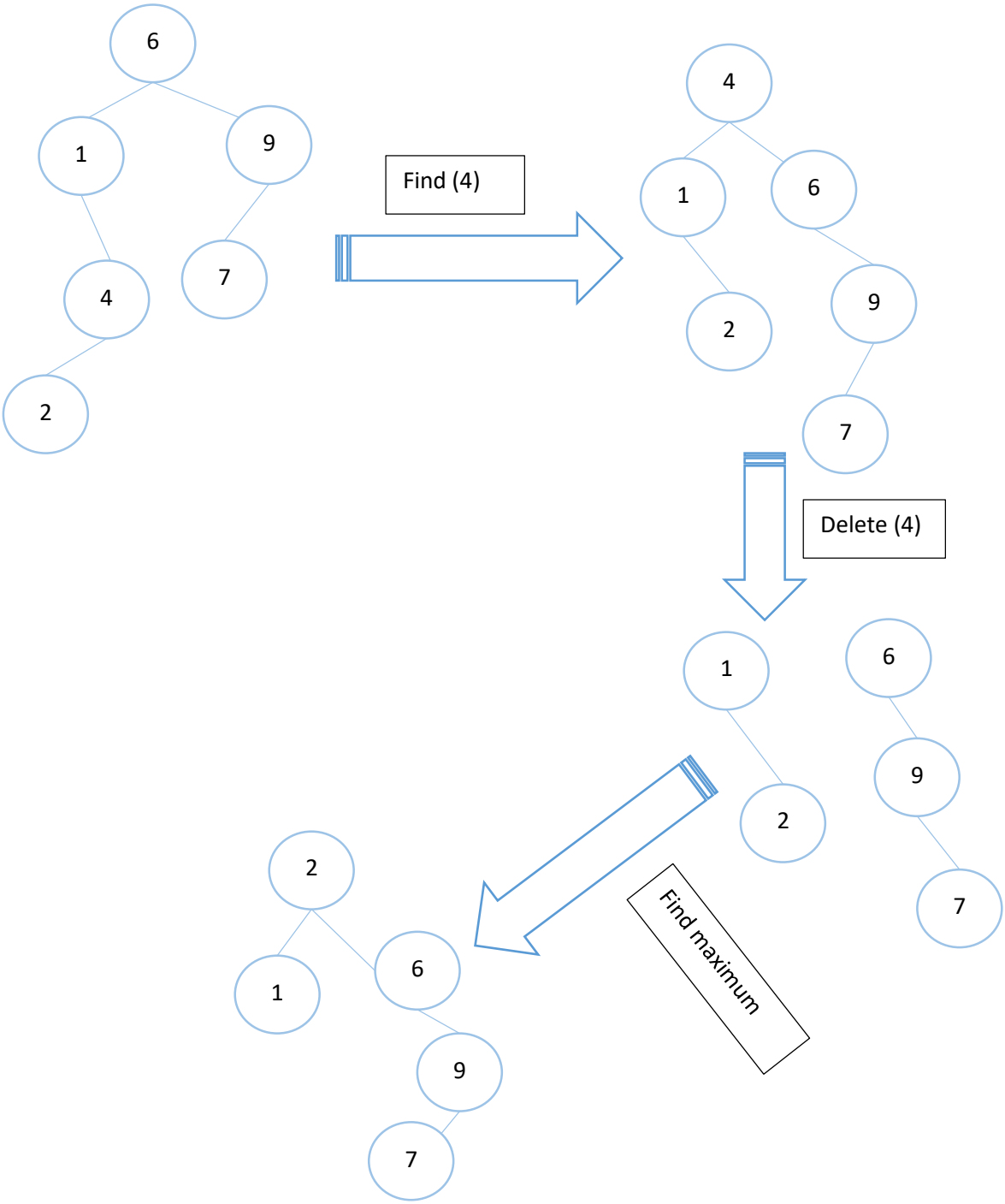
Splay Tree Delete

Following are the different cases to delete a key **k** from splay tree.

- ✓ If Root is NULL: We simply return the root.
- ✓ Else splay the given key k. If k is present, then it becomes the new root. If not present, then last accessed leaf node becomes the new root.
- ✓ If new root's key is not same as **k**, then return the root as, **k** is not present.
- ✓ Else the key **k** is present.

- Split the tree into two trees Tree1 = root's left sub tree and Tree2 = root's right sub tree and delete the root node.
- Let the roots of Tree1 and Tree2 be Root1 and Root2 respectively.
- If Root1 is NULL: Return Root2.
- Else, Splay the maximum node (node having the maximum value) of Tree1.
- After the Splay procedure, make Root2 as the right child of Root1 and return Root1.

Example: delete (4)



Advantages

- Average-case performance is an efficient as other trees.
- Splay trees do not need to store any bookkeeping data.
- Simple implementation
- Allows access to both the previous and new versions after an update (persistent data structure).
- Stable sorting

  .

Disadvantages

- The most significant disadvantage of splay trees is that the height of a splay tree can be linear.
- The representation of splay trees can change even when they are accessed in a 'read-only' manner.

Application of Splay Tree

- Network router

  A network router receives network packets at a high rate from incoming connections and must quickly decide on which outgoing wire to send each packet, based on the IP address in the packet. The router needs a big table (a map) that can be used to look up an IP address and find out which outgoing connection to use. Splay trees can provide good performance in this situation