

2

```
def insert(root, data):
    if root is None:
        return Node(data)
    if data < root.key:
        root.left = insert(root.left, data)
    else:
        root.right = insert(root.right, data)
    return root
```

```
def search(root, data):
    if root is None:
        return False
    if root.key == data:
        return True
    if data < root.key:
        return search(root.left, data)
    else:
        return search(root.right, data)
```

```
D Inorder traversal:
def inorder(root):
    if root is None:
        return
    inorder(root.left)
    print(root.key)
    inorder(root.right)
```

```
H Preorder traversal:
def preorder(root):
    if root is None:
        return
    print(root.key)
    preorder(root.left)
    preorder(root.right)
```

```
Postorder traversal:
def postorder(root):
    if root is None:
        return
    postorder(root.left)
    postorder(root.right)
    print(root.key)
```

2

2

Python program to insert element in binary tree 2

```
class Node:
    def __init__(self, data):
        self.key = data
        self.left = None
        self.right = None
```

A function to do inorder tree traversal 2

```
def printInorder(root):
    if root:
        # First recur on left child
        printInorder(root.left)

        # then print the data of node
        print(root.val),

        # now recur on right child
        printInorder(root.right)
```

A function to do postorder tree traversal 2

```
def printPostorder(root):
    if root:
        # First recur on left child
        printPostorder(root.left)

        # the recur on right child
        printPostorder(root.right)

        # now print the data of node
        print(root.val),
```

A function to do preorder tree traversal 2

```
def printPreorder(root):
    if root:
        # First print the data of node
        print(root.val),

        # Then recur on left child
```

2

```

printPreorder(root.left)

# Finally recur on right child
printPreorder(root.right)

"""function to insert element in binary tree """
def insert(temp, key):
    q = []
    q.append(temp)
    # Do level order traversal until we find
    # an empty place.
    while len(q):
        temp = q[0]
        q.pop(0)
        if not temp.left:
            temp.left = newNode(key)
            break
        else:
            q.append(temp.left)
        if not temp.right:
            temp.right = newNode(key)
            break
        else:
            q.append(temp.right)

# function to delete the given deepest node (d_node) in binary tree
def deleteDeepest(root, d_node):
    q = []
    q.append(root)
    while len(q):
        temp = q.pop(0)
        if temp is d_node:
            temp = None
            return
        if temp.right:
            if temp.right is d_node:
                temp.right = None
                return
            else:
                q.append(temp.right)
        if temp.left:
            if temp.left is d_node:
                temp.left = None
                return
            else:
                q.append(temp.left)

# function to delete element in binary tree
def deletion(root, key):
    if root == None:
        return None
    if root.left == None and root.right == None:
        if root.key == key:
            return None
        else:
            return root
    key_node = None
    q = []

```

```

q.append(root)
while(len(q)):
    temp =q.pop(0)
    iftemp.data ==key:
        key_node =temp
    iftemp.left:
        q.append(temp.left)
    iftemp.right:
        q.append(temp.right)
ifkey_node :
    x =temp.data
    deleteDeepest(root,temp)
    key_node.data =x
returnroot

##
# Driver code
if__name__ == '__main__':
    root =newNode(10)
    root.left =newNode(11)
    root.left.left =newNode(7)
    root.right =newNode(9)
    root.right.left =newNode(15)
    root.right.right =newNode(8)

    print"Preorder traversal of binary tree is"
    printPreorder(root)

    print"\nInorder traversal of binary tree is"
    printInorder(root)

    print"\nPostorder traversal of binary tree is"
    printPostorder(root)

    key =12
    insert(root, key)

    print"Preorder traversal of binary tree is"
    printPreorder(root)

    print"\nInorder traversal of binary tree is"
    printInorder(root)

    print"\nPostorder traversal of binary tree is"
    printPostorder(root)

    key =11
    root =deletion(root, key)

    print"Preorder traversal of binary tree is"
    printPreorder(root)

    print"\nInorder traversal of binary tree is"
    printInorder(root)

    print"\nPostorder traversal of binary tree is"
    printPostorder(root)

# Recursive Python program for level order traversal of Binary Tree

```


2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K

2
202009'N'P'19'N9!K



2
2
2
2
2
2

2

88

K

2

2
2
2
2

2

8

K

2

2
2
2
2
2
2

2

88

2

2

2

2

8

2
2
2
2
2

2

2

2

2

2

2

2

2

2

2

2

2

2

2

2

2

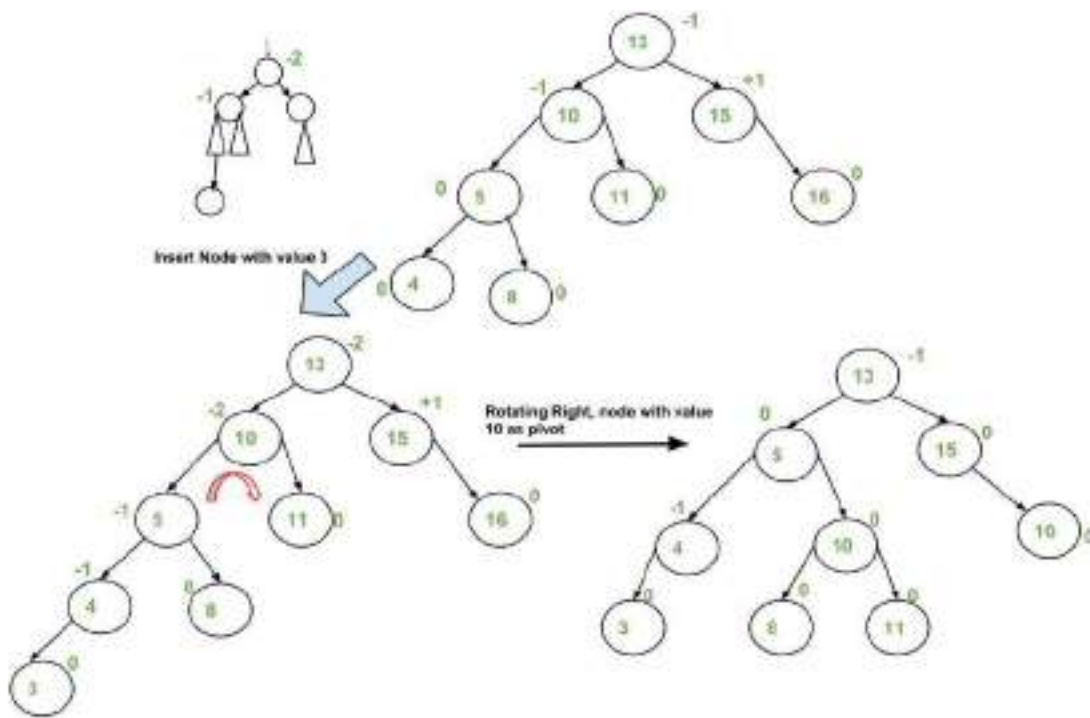
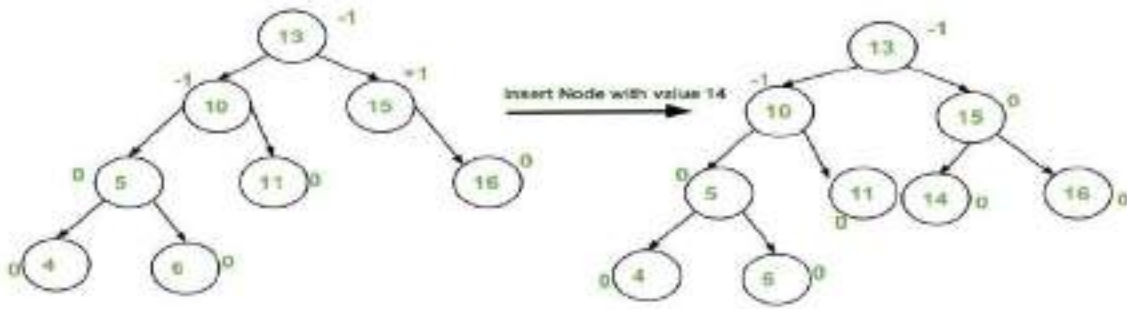
2

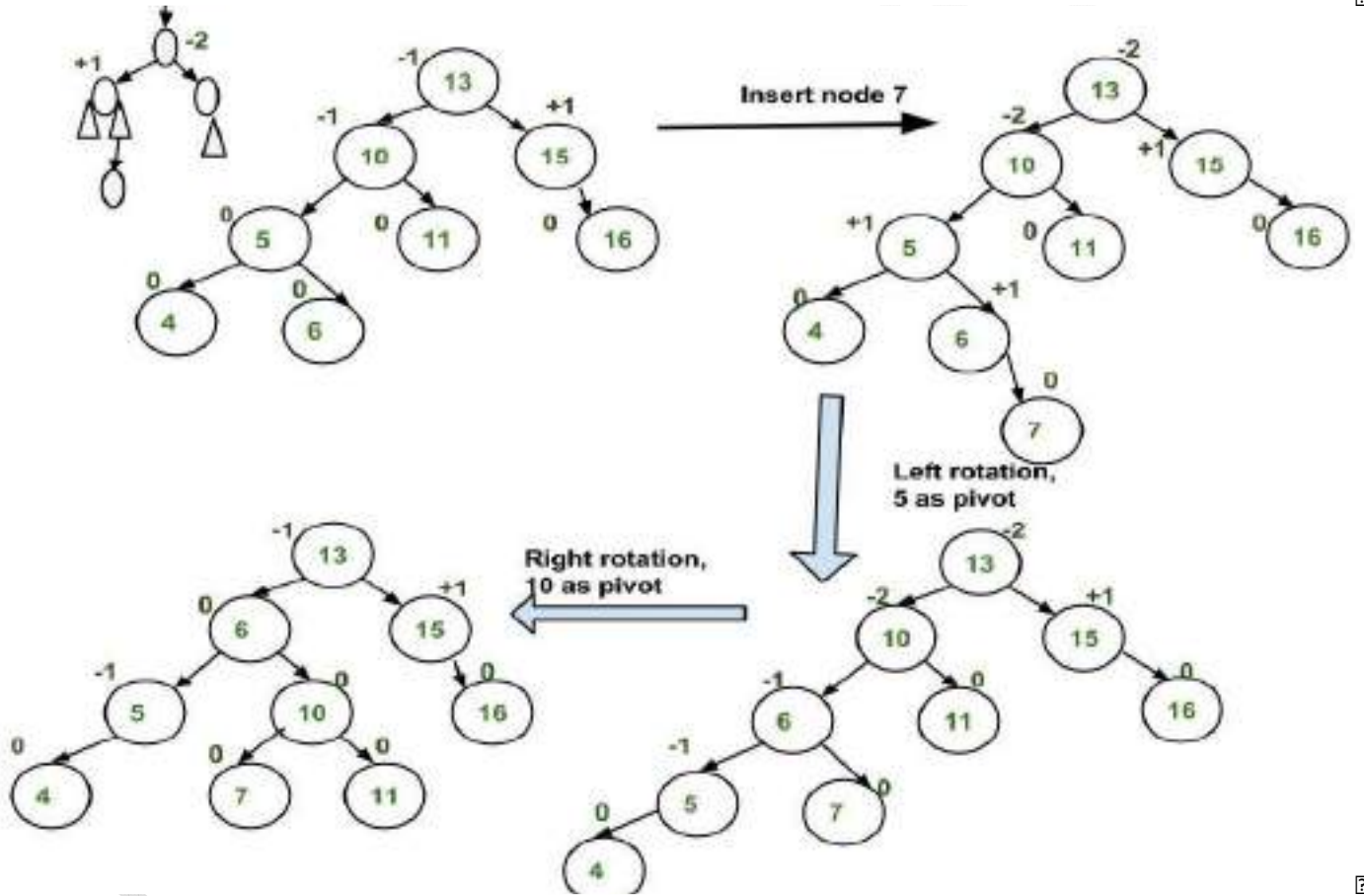
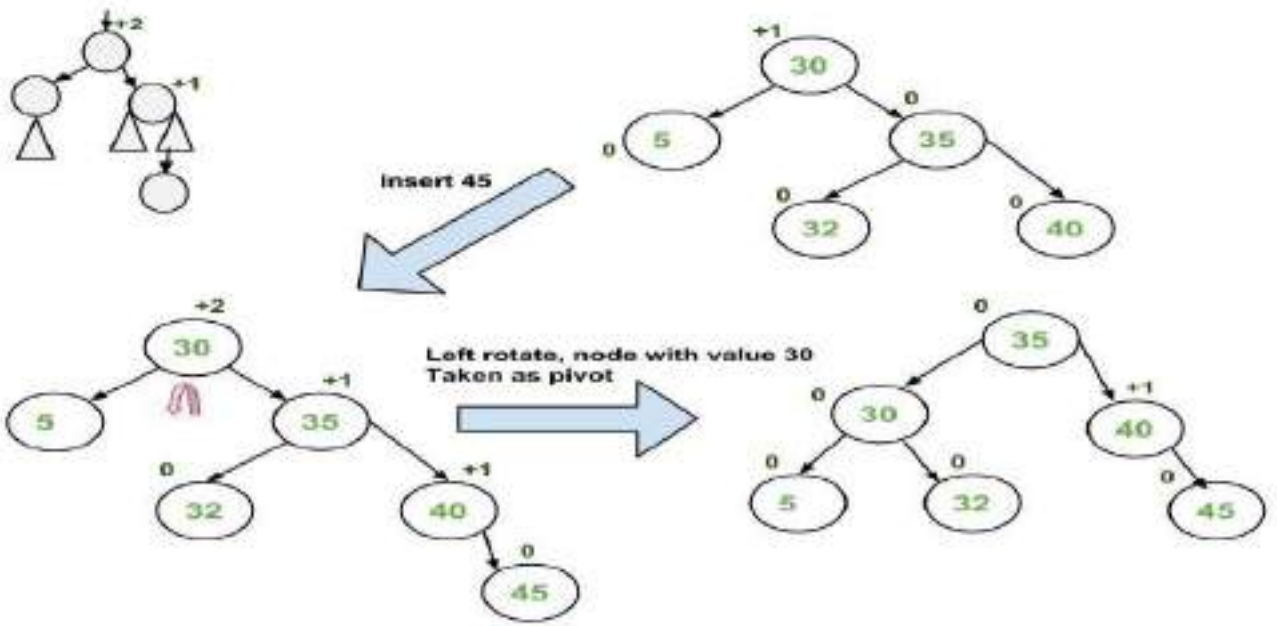
2

2

2

Insertion Examples:





2

ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

୧, ୩, ୪, ୫, ୬, ୭, ୮, ୯, ୧୦, ୧୧, ୧୨, ୧୩, ୧୪, ୧୫, ୧୬, ୧୭, ୧୮, ୧୯, ୨୦, ୨୧, ୨୨, ୨୩, ୨୪, ୨୫, ୨୬, ୨୭, ୨୮, ୨୯, ୩୦, ୩୧, ୩୨, ୩୩, ୩୪, ୩୫, ୩୬, ୩୭, ୩୮, ୩୯, ୪୦, ୪୧, ୪୨, ୪୩, ୪୪, ୪୫, ୪୬, ୪୭, ୪୮, ୪୯, ୫୦, ୫୧, ୫୨, ୫୩, ୫୪, ୫୫, ୫୬, ୫୭, ୫୮, ୫୯, ୬୦, ୬୧, ୬୨, ୬୩, ୬୪, ୬୫, ୬୬, ୬୭, ୬୮, ୬୯, ୭୦, ୭୧, ୭୨, ୭୩, ୭୪, ୭୫, ୭୬, ୭୭, ୭୮, ୭୯, ୮୦, ୮୧, ୮୨, ୮୩, ୮୪, ୮୫, ୮୬, ୮୭, ୮୮, ୮୯, ୯୦, ୯୧, ୯୨, ୯୩, ୯୪, ୯୫, ୯୬, ୯୭, ୯୮, ୯୯, ୧୦୦

ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

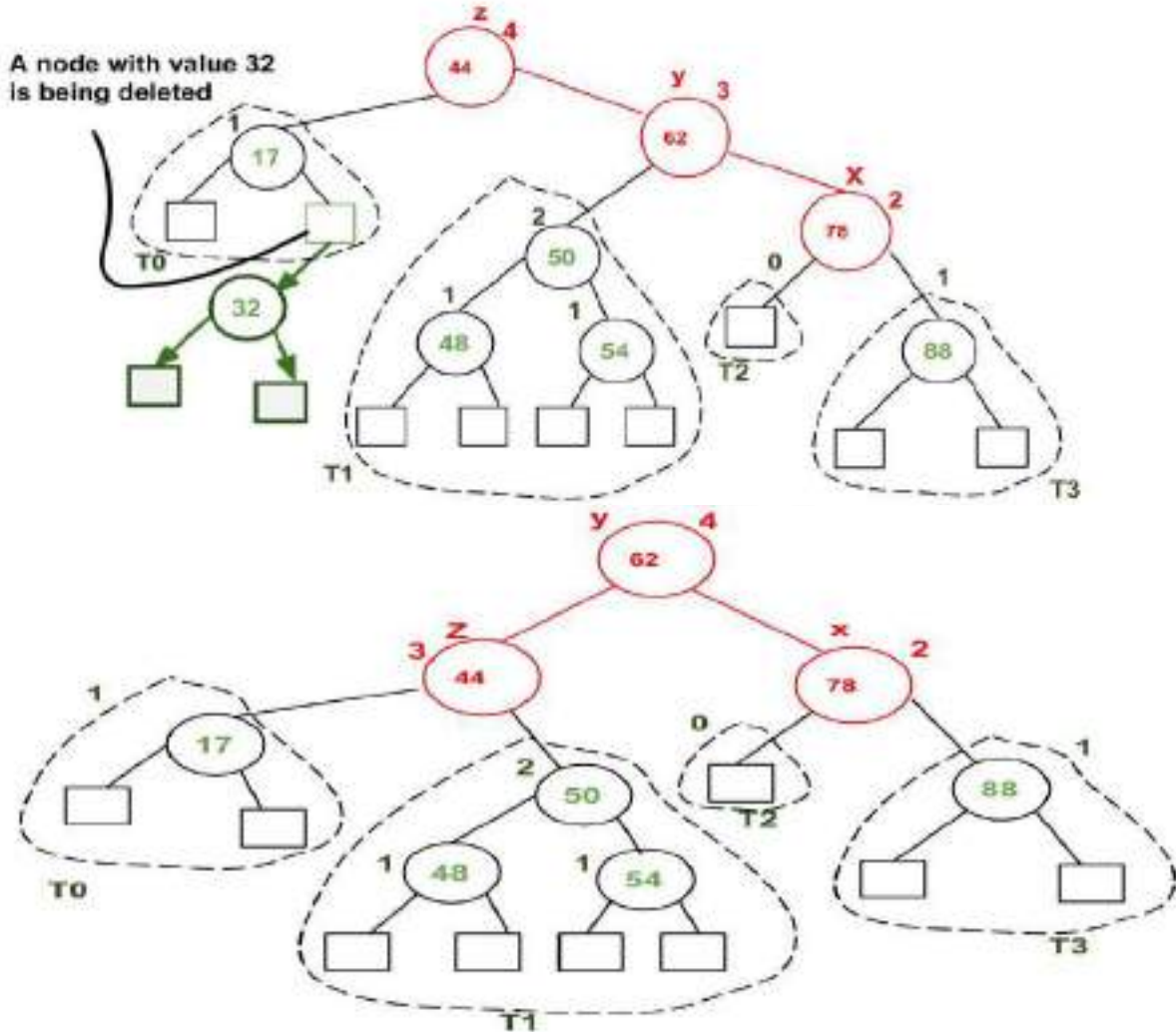
ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

2

Example of deletion from an AVL Tree:



2

2

ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

2

ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

ଉତ୍ତର [ଆବରଣର](#) କ୍ରମ

2

2

2

2 22222222 2222 22222222

2 22222222 2222 % 22222222

2

2 888 222222 22222222

2 245 22222222

2 245 22222222/ 2222

2

2 888 22222222 22222222

2 245 22222222 22222222 245 22222222 22222222 245 22222222 22222222 22222222

2 245 22222222 22222222 245 22222222 22222222 245 22222222 22222222 22222222

2

2 888 22222222# 22222222

2 22222222

2

2

2 888 22222222 22222222 22222222 22222222 22222222 22222222 22222222 22222222 22222222

2 888 22222222 22222222 22222222 22222222

2 22222222 2222 2222 222222222222 2222 2222 2222

2

2 22222222 2222 22222222

2 22222222 2222 % 22222222

2

2 888 222222 22222222

2 245 22222222

2 245 22222222/ 2222

2

2 888 22222222 22222222

2 245 22222222 22222222 245 22222222 22222222 245 22222222 22222222 22222222

2 245 22222222 22222222 245 22222222 22222222 245 22222222 22222222 22222222

2

2 888 22222222# 22222222

2 22222222

2

2

2 888 2222 2222 2222 2222 2222 2222 2222 2222 2222 2222 2222 2222 2222 2222

2 22222222 2222 2222 2222 2222 2222 2222 2222 2222 2222 2222 2222 2222 2222 2222

2

2 2222 2222 F -- 2222

2 22222222

2 22222222 22222222 22222222 22222222 22222222 22222222

2

2

2 22222222 2222 222222222222 2222 22222222 22222222

2

2 8 2222 222222 22222222 22222222 22222222 22222222 22222222 22222222

2 22222222 2222 F -- 2222

2 22222222# 22222222 22222222

2

2 2222 2222 22222222 (22222222

2 22222222 22222222 22222222 22222222 22222222 22222222

2 22222222 22222222 22222222 (22222222

2 22222222 22222222 22222222 22222222 22222222 22222222

2 22222222 F 22222222 22222222# 22222222

2 22222222 22222222

2

?

2) ...

2) ...

2) ...

2) ...

2) ...

2) ...

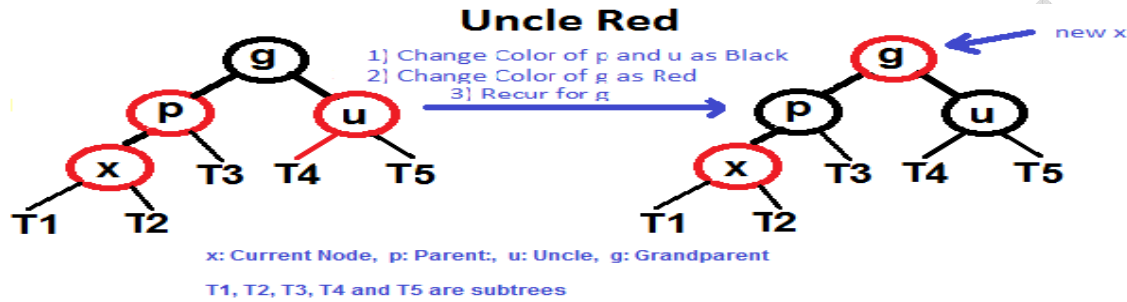
2) ...

2) ...

2) ...

2) ...

?



?

?

2) ...

2) ...

2) ...

2) ...

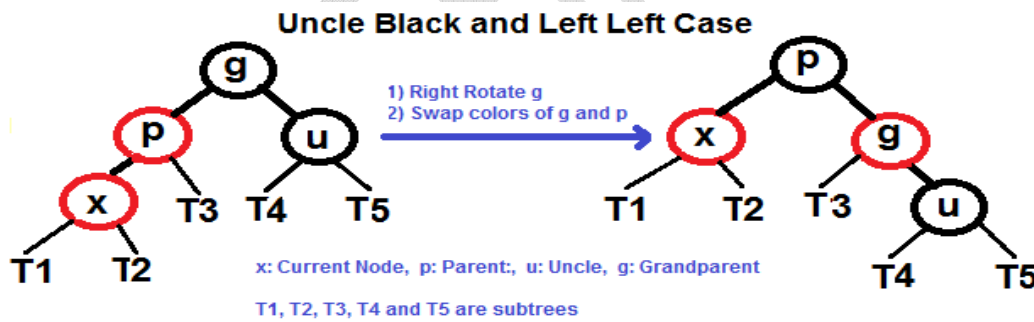
2) ...

2) ...

2) ...

2) ...

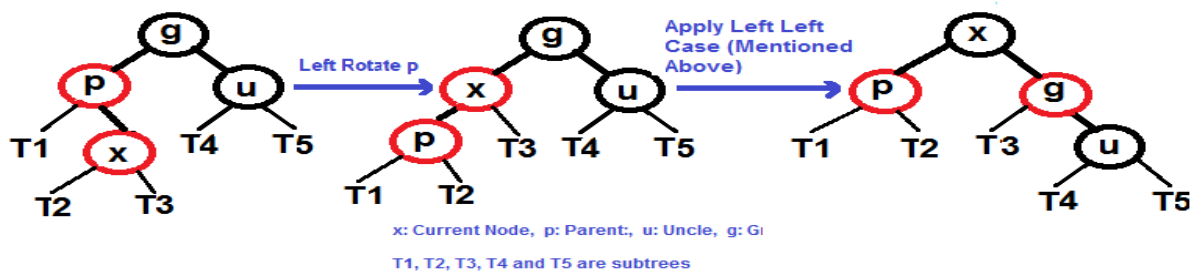
?



?

2) ...

Uncle Black and Left Right Case



?

2) ...

?

2
2 2 2 2

2 2 2 2 81 2020/21 2020/21

2 2 2 2 * 2020/21 2020/21 2020/21 82

2 2 2 2 2020/21 2020/21 2020/21

2 2 2 2 2020/21

2 2 2 2 2020/21 * 2020/21 2020/21 2020/21

2 2 2 2 2020/21 2020/21

2 2 2 2 2020/21 2020/21 2020/21

2 2 2 2 2020/21

2 2 2 2 81 2020/21 2020/21

2 2 2 2 2020/21 2020/21 2020/21 2020/21

2 2 2 2 - 2020/21 2020/21 2020/21 82

2 2 2 2 2020/21 - 2020/21 2020/21 2020/21 2020/21

2 2 2 2 2020/21 2020/21 2020/21 2020/21 2020/21

2 2 2 2 2020/21 2020/21

2 2 2 2 2020/21

2 2 2 2 2020/21

2 2 2 2 2020/21

2 2020/21 2020/21 - 2020/21

2020/21

2 88 2020/21 2020/21 2020/21 2020/21 2020/21 2020/21

2020/21 2020/21, 2020/21 2020/21 2020/21

2020/21

2 K 2020/21 2020/21 2020/21 2020/21

2 88 2020/21 2020/21 2020/21 2020/21

2 2020/21 2020/21 2020/21 2020/21

2 88 2020/21 2020/21 2020/21 2020/21

2 2020/21 2020/21 2020/21 2020/21

2020/21 2020/21 2020/21 2020/21

2020/21

2 88 2020/21 2020/21 2020/21 2020/21 2020/21

2020/21 2020/21, 2020/21 2020/21 2020/21 2020/21 2020/21

2020/21 2020/21, 2020/21 2020/21 2020/21 2020/21 2020/21

2020/21 2020/21, 2020/21 2020/21 2020/21 2020/21 2020/21

2 88 2020/21 2020/21

2020/21 2020/21

2020/21

2 * 2020/21 2020/21

2020/21

2 2020/21 2020/21 2020/21

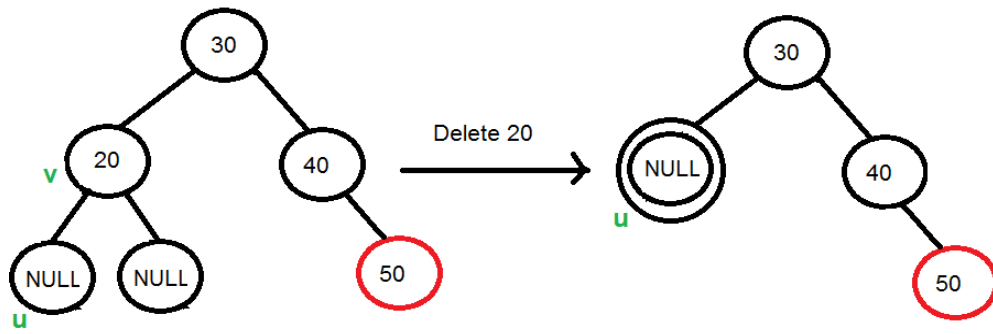
2 2020/21 2020/21 2020/21

2 2020/21 2020/21 2020/21

2 2020/21 2020/21 2020/21

2 2020/21 2020/21 2020/21

2020/21



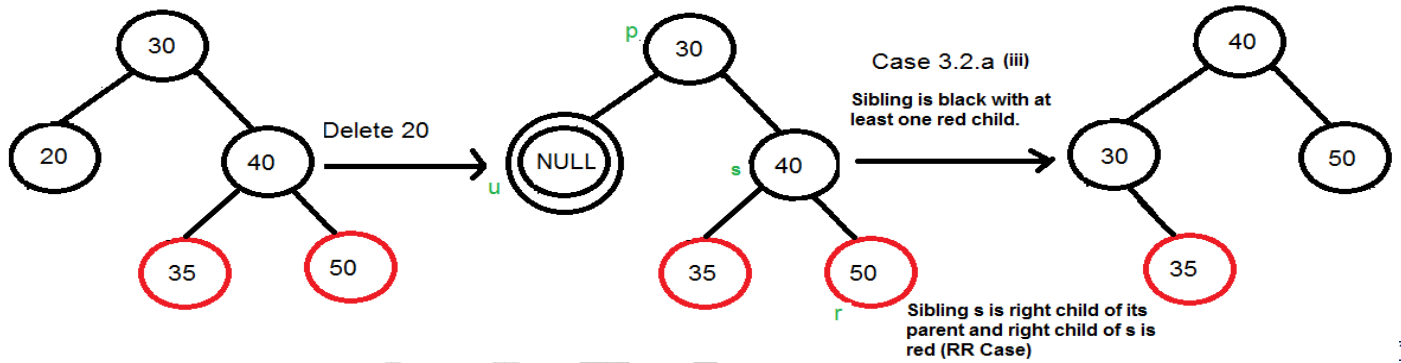
When 20 is deleted, it is replaced by a NULL, so the NULL becomes double black.
 Note that deletion is not done yet, this double black must become single black

Q. In a B+ tree, the leaf nodes are 10, 20, 30, 40, 50, 60, 70, 80, 90. The root node is 50. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black.

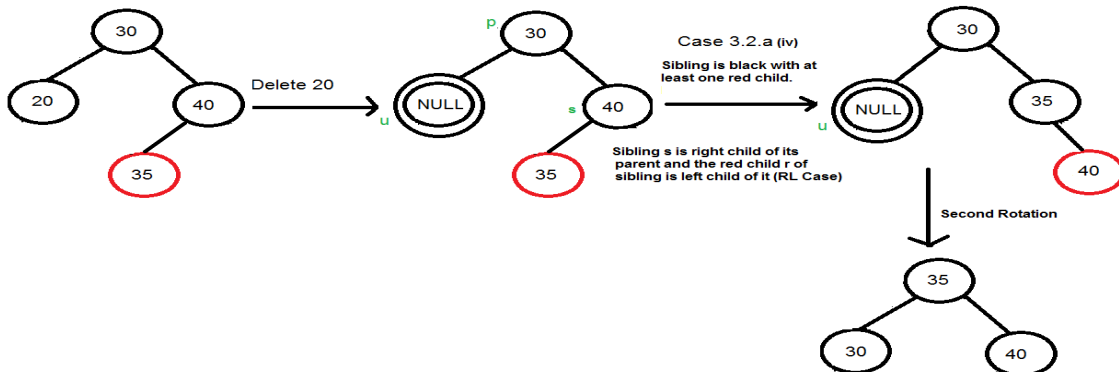
Q. In a B+ tree, the leaf nodes are 10, 20, 30, 40, 50, 60, 70, 80, 90. The root node is 50. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black.

Q. In a B+ tree, the leaf nodes are 10, 20, 30, 40, 50, 60, 70, 80, 90. The root node is 50. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black.

Q. In a B+ tree, the leaf nodes are 10, 20, 30, 40, 50, 60, 70, 80, 90. The root node is 50. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black.



Q. In a B+ tree, the leaf nodes are 10, 20, 30, 40, 50, 60, 70, 80, 90. The root node is 50. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black.



Q. In a B+ tree, the leaf nodes are 10, 20, 30, 40, 50, 60, 70, 80, 90. The root node is 50. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black. The leaf nodes 10, 20, 30, 40, 50, 60, 70, 80, 90 are red. The root node 50 is black.

2
2

88

K

88
66K F --
K F --

2
2

245
88
245
245
88
245

A
2

88
& 245

2
2

88

88
66K F --
K F --

2
2

245
245

A
2

88
245

66K F --
245
A
245

2
2

245
245

A
2

& 245
66K F --
66K F --

A
A

2
%

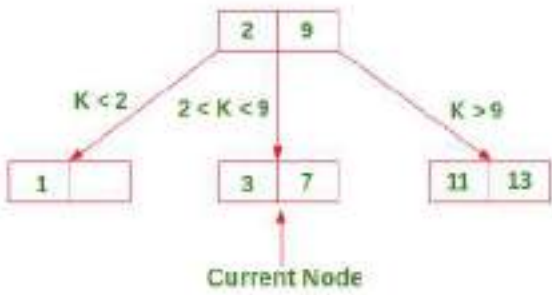
K

2
2

88
245
88

2
2

2



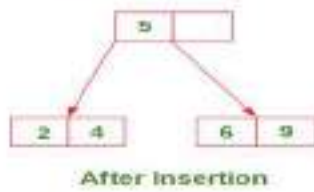
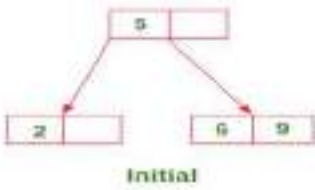
2

2

2

-
 -
 -

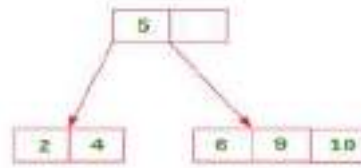
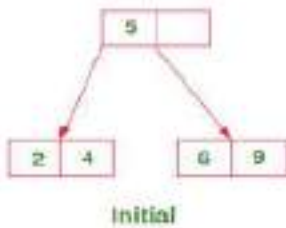
Insert 4 in the following 2-3 Tree:



2

-
 -
 -

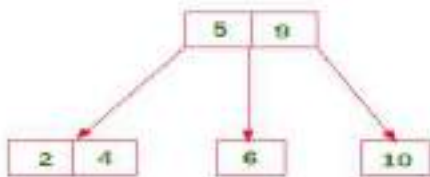
Insert 10 in the following 2-3 Tree:



Temporary Node with 3 data elements

2

2

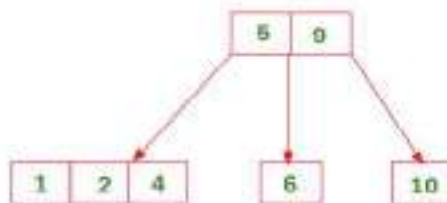
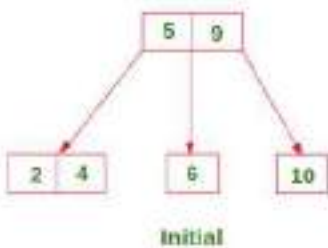


Move the middle element to parent and split the current Node

2

-
 -
 -

Insert 1 in the following 2-3 Tree:



Temporary Node with 3 data elements

2

2

2

