Red-Black Tree Insertion

P. SAI DEEKSHITH
IBM18CC148

```
Void RBTree :: insert (const int & data) {
    Node * ptr = new Node (data);
    root = BSTInsert (root, ptr);
    fixViolation (root, ptr);
}

Node * BSTInsert (Node * root, Node * pt) {
    if (root == NULL) return pt;
    if (pt -> data < root -> data) {
        root -> left = BSTInsert (root->left, pt);
        root -> left -> parent = root;
    }
    else if (pt -> data > root -> data) {
        root -> right = BSTInsert (root -> right
                                   pt);
        root -> right -> parent = root;
    }
    return root;
}
```

Case A : Parent of pt is left child of Grand Parent of pt.

Case 1 : The uncle of pt is also red only Recoloring required

Case 2 : pt is right child of its parent left rotation required.

Case 3 : pt is left child of its parent right rotation required.

Case B : Parent of pt is right child of Grand-parent of pt.

Case 1 : The uncle of pt is also red only recoloring is required.

Case 2 : pt is left child of its parent Right rotation is Required.

Case 3 : pt is right child of its parent then left rotation is Required.