Red Black Tree

KARTHIK VENUGOPAL

insertion

IBM18CS043

```
insert ( data )
{
        newn =  new Node (data)

        root =  BSTInsert (root , newn )

        fix (root , newn);

    }


BSTInsert (root , newn)
{
        if (root == NULL)

            return newn

        if (newn->data < root->data)
            {
            root->left = BSTInsert (root->left , newn);
            root->left->parent = root;

            }

        else if (newn->data >= root->data)
            {
            root->right = BSTInsert (root->right , newn);
            root->right->parent = root;

            }

        return root;

    }
```

①

```
fix( root, newn)
{

    pn = NULL
    gpn = NULL
    while ( (newn != root) && (newn -> color != BLACK) &&
            (newn -> parent -> color = RED))

    {

        pn = pn -> parent;

        gpn = newn -> parent -> parent

        if (pn == gpn -> left)

        {

            uncle = gpn -> right
            if (uncle != NULL && uncle -> color == RED)

            {
                gpn -> color = RED,
                parent -> color = BLACK
                uncle -> color = BLACK
                newn = gpn
            }

            else
            {
```

```
if (newn = pn -> right)
{
    rotate Left (root, peren pn);

    newn = pn;

    Pn = newn -> parent
}

rotate right (root, gpn);

swap (pn -> color, gpn -> color);

pn = newn = pn
}
}

else
{
    st uncle = gpn -> left

    if ((uncle != NULL) && (uncle -> color == RED))
    {
        gpn -> color = RED

        pn -> color = BLACK

        uncle -> color = BLACK

        newn = gn
    }
    else
    {
```

```
if (newn = pn->left)
{
    rotateRight (root, pn)

    newn = pn

    pn = newn->parent;

}

    rotateleft (root , gpn)

    swap ( pn->color, gpn->color)

                newn = pn

        }

    }
    }

    root->color = BLACK

}

rotateRight (root, newn)
{
        nleft = newn->left

        newn->lef = nleft->rigt;

        if (newn->left != NULL)

            newn->left->parent = newn
```

new nleft-> parent = newn-> parent

if (newn->parent == NULL)

   root = nleft

else if (newn == newn->parent->left)

~~else parent~~ newn->parent->left = hleft

else

     newn->parent->right = nleft

   nleft->right = newn;

   newn->parent = nleft

}

rotateleft (root , newn)

{

   nright = newn->right

   newn->right = nright->left

   if (newn->right != NULL)

     newn->right->parent = newn;

   nright->parent = newn->parent

   if (newn->parent == NULL)

     root = nright

```
else if (newn == newn -> parent->l.s1)

    newn -> parent -> left =    n right

else

        newn -> parent -> right = nright

nright -> left = newn

newn -> parent = nright

}
```