ADS - DAY 8

18-NOV-2018
Dhanraj K
1BM18CS027

Red-Black tree

```
//Node
  -> attributes  -> int data;
                    Node * parent;
                    Node * left;
                    Node * right;
                    int color;

// For balancing the tree after insertion

void   insertFix( NodePtr . K){

       NodePtr u;
       while (K->Parent->color ==1 ){
             if ( K-> Parent == K->Parent-> Parent ->right) {
                      u= K-> Parent -> Parent->left;
                      if (u-> color ==1){
                          u -> color = 0;
                          K -> Parent -> color =0 ;
                          K -> Parent -> parent -> color = 1;
                          K= K-> parent ->parent;

             }else {
                      if (K== K ->Parent -> left) {
                              K= K-> Parent;
                              right Rotate (K);

               }
               K -> Parent -> color =0;
               K -> Parent -> Parent -> color =1;
               left Rotate ( K->Parent -> Parent);
         }
}else {
        u=K-> parent -> parent -> right;
        if (u-> color ==1){
              u-> color =0;
              K-> Parent -> color =0;
              K-> Parent -> Parent-> color=1;
               K= K-> Parent -> Parent;
        } else{
              if (K== K-> parent->right){
                      K= K-> parent;
                      ~~lef~~ left Rotate(K);
              }
              K-> Parent ->color =0;
              K-> parent ->parent->color=1;
              right Rotate(K-> Parent ->Parent);
        }
    }
    if (K== root) {
            break;
    }
  }
  root -> color =0;
}
```

//insertion

```
Void insert (int Key) {
      NodePtr node= new Node;
      node-> parent = nullptr;
      node-> data = Key;
      node-> left = TNULL;
      node-> right = TNULL;
node->color = 1;
NodePtr y = nullptr;
Nodeptr x = this->root;

while (x != TNULL) {
        y = x;
        if (node->data < x->data){
            x = x->left;
        } else{
            x = x->right
        }
    }

    node->parent = y;
    if (y == nullptr) {
            root = node;
    } else if (node->data < y->data){
            y->left = node;
        } else {
            y->right = node;
    }
    if (node->parent == nullptr){
            node->color = 0;
            return;
    }
    if (node->parent->parent == nullptr){
            return;
    }
    insertFix(node);
}
```