Harshit Hiremath,
1BM18CS036
18/11/20

ADS Lab 8 - Red-Black Tree

Insertion algorithm:-

1. Perform BST insert, set col. of new node x=RED
2. If x is root, ther col of x = BLACK
3. If color of x's parent != BLACK or x is not root:
   a. If x's uncle is RED
      i. Change parent col & uncle col = BLACK
      ii. Grandparent Col = RED
      iii. Change x's col = grandparent's col.
   b. Else (x's uncle is BLACK)
      a) Determine
         i. Left-left case
         ii. Left-right case
         iii. Right-right case
         iv. right-left case
      b) Change x = x's parent

Basic structs & nodes:

```
struct Node {
    int data;
    bool color;
    Node *left, *right, *parent;

    Node (int data) {
        this->data = data
        left = right = parent = NULL;
        this->color = RED;
    }
};
class RBTree {
    private: Node* root;
        void rotateLeft(Node *&, Node *&)
        void rotateRight(Node *&, Node *&)
        void fixViolation(Node *&, Node *&)
```

```
public:
    RBTree() { root = NULL; }
    void insert(const int &n);
    void inorder();
};
```

Utility functions :

```
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;
}
void RBTree::rotateLeft(Node *&root, Node *&pt){
    Node* pt_right = pt->right;
    pt->right = pt_right->left;
    if (pt->right != NULL) pt->right->parent = pt;
    pt_right->parent = pt_parent;
    if (pt->parent == NULL) root = pt_right;
    else if (pt == pt->parent->left) pt->parent->left = pt_right;
    else pt->parent->right = pt_right;
    pt_right->left = pt;
    pt_parent = pt_right;
}
void RBTree::rotateRight(Node *&root, Node *&pt){
    Node* pt_left = pt->left;
    pt->left = pt_left->right;
    if( pt->left != NULL) pt->left->parent = pt;
    pt_left->parent = pt->parent;
    if( pt->parent == NULL) root = pt_left;
```

1RM18CS036
Harshit Hiremath

```
    else if ( pt == pt->parent->left)
        pt->parent->left = pt_left;
    else    pt->parent->right = pt_left;
    pt_left->right = pt;
    pt->parent = pt_left;
}

void RBTree::fixViolation(Node *&root, Node *&pt) {
Node* parent_pt = NULL; *grand_parent_pt = NULL;
E:
while( (pt != root) && (pt->color != BLACK) && (pt->parent->color
                                                == RED)) {
    parent_pt = pt->parent;
    grandparent_pt = pt->parent->parent;
    if (parent_pt == grand_parent_pt->left) {
        Node* uncle_pt = grand_parent_pt->right;
        if (uncle_pt != NULL && uncle_pt->color == RED) {
            grand_parent_pt->color = RED;
            parent_pt->color = BLACK;
            uncle_pt->color = BLACK;
            pt = grand_parent_pt;
        }
        else{
            if (pt == parent_pt->right) {
                rotateleft(root, parent_pt);
                pt = parent_pt;
                parent_pt = pt->parent;
            }
            rotateRight(root, grand_parent_pt);
            swap(parent_pt->color, grand_parent_pt->color);
            pt = parent_pt;
        }
    }
    else {
        Node *uncle_pt = grand_parent_pt->left;
        if ((uncle_pt != NULL) && (uncle_pt->color == RED)){
            grand_parent_pt->color = RED;
            parent_pr->color = BLACK;
            uncle_pt->color = BLACK;
            pt = grand_parent_pt;
        }
```

```
            else {
                if (pt == parent_pt -> left) {
                    rotateRight (root, parent_pt);
                    pt = parent_pt;
                    parent_pt = pt -> parent;
                }
                rotateLeft (root, grandparent_pt)
                swap (parent_pt -> color, grand_parent_pt -> color);
                pt = parent_pt;
            }
        }
    }
    root -> color = BLACK;
}
void RBTree :: insert (const int &data) {
    Node *pt = new Node (data);
    root = BSTInsert (root, pt);
    fixViolation (root, pt);
}
```