Name: Shah Siddh Tejaskumar

Reg.no.: 20BCE1937

Slot: L55 + L56

--------------------------------------------------------------------------------------------------------------------------------

BST creation AND Traversal

#include <iostream>

using namespace std;

class tree

{

public:

    int data;

    tree \*left;

    tree \*right;

};

tree \*createNode(int data) // creating the nodes

{

    tree \*node;

    node = (tree \*)malloc(sizeof(tree));

    node->data = data;

    node->left = NULL;

    node->right = NULL;

    return node;

}

tree \*create\_tree(int l, int h, int arr[])  // function to create the BST

{

    if (l > h)  // end points or if empty condition

        return NULL;

    int mid = (l + h) / 2;

    tree \*root;

    root = createNode(arr[mid]);

    root->left = create\_tree(l, mid - 1, arr);

    // calling the same function to create left sub\_tree

    root->right = create\_tree(mid + 1, h, arr);

    // calling th same function to create right sub\_tree

    return root;

}

void preOrder(tree \*root)  // Pre-Oreder Traversal

{

    if (root != NULL)

    {

        cout << root->data << " ";

        preOrder(root->left);

        preOrder(root->right);

    }

}

void postOrder(tree \*root)  // Post-Oreder Traversal

{

    if (root != NULL)

    {

        postOrder(root->left);

        postOrder(root->right);

        cout << root->data << " ";

    }

}

void inOrder(tree \*root)   // In-Oreder Traversal

{

    if (root != NULL)

    {

        inOrder(root->left);

        cout << root->data << " ";

        inOrder(root->right);

    }

}

void sort(int n, int \*arr)  // using the insertion sort to sort an arry

{

    int temp, precedence;

    for (int i = 1; i < n; i++)

    {

        temp = arr[i];

        precedence = i - 1;

        while (precedence >= 0 && arr[precedence] > temp)

        {

            arr[precedence + 1] = arr[precedence];

            precedence--;

        }

        arr[precedence + 1] = temp;

    }

}

int main()

{

    int n;

    cout << "Enter ele.: ";

    cin >> n;

    int arr[n];

    for (int i = 0; i < n; i++)

    {

        cin >> arr[i];

    }

    sort(n, arr);

    tree \*BST;

    BST = create\_tree(0, n-1, arr);

    cout << "Pre-Order Traversal: ";

    preOrder(BST);

    cout << endl;

    cout << "Post-Order Traversal: ";

    postOrder(BST);

    cout << endl;

    cout << "In-Order Traversal: ";

    inOrder(BST);

    cout << endl;

    return 0;

}

// all the traversal will have Time Complextity = O(n)

// Space Complexity in BEST case ~= log(n)  (Balanced Tree, which we are constructing here)

// Space Complexity in WORST case ~= O(n)  (eg. Skew tree)