ADSA _LAB3

- BST Creation, Traversal
- Diameter of BST
- No of leaf nodes, Internal Node, height of BST

Open | bst1.c ~/Documents/ADSANEW_LAB/lab 3 | Save

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h> // Include math.h for fmax function

float c = 0, p = 0;

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

struct Node* create(int item) {
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = item;
    node->left = node->right = NULL;
    return node;
}

int maxheight(struct Node* root) {
    if (root == NULL) {
        return 0;
    } else {
        int lHeight = maxheight(root->left);
        int rHeight = maxheight(root->right);
        if (lHeight > rHeight) {
            return (lHeight + 1);
        } else {
            return (rHeight + 1);
        }
    }
}

int diameter(struct Node* root) {
    if (root == NULL)
        return 0;
    int lheight = maxheight(root->left);
    int rheight = maxheight(root->right);
    int ldiameter = diameter(root->left);
    int rdiameter = diameter(root->right);
    c++;

    // Use fmax from math.h to find the maximum
    return (int)fmax(fmax(ldiameter, rdiameter), lheight + rheight + 1);
}

void inorder(struct Node* root) {
    if (root == NULL)
        return;
    inorder(root->left);
    printf("%d ", root->data);
    inorder(root->right);
}

void preorder(struct Node* root) {
    if (root == NULL)
        return;
    printf("%d ", root->data);
    preorder(root->left);
    preorder(root->right);
}

void postorder(struct Node* root) {
    if (root == NULL)
        return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->data);
}
```

```c

struct Node* insert(struct Node* root, int item) {
    if (root == NULL)
        return create(item);
    if (item < root->data) {
        root->left = insert(root->left, item);
    } else {
        root->right = insert(root->right, item);
    }
    c++;
    return root;
}

int main() {
    struct Node* root = NULL;
    int num;
    char choice;

    do {
        printf("Enter an integer to insert into the binary tree: ");
        scanf("%d", &num);
        root = insert(root, num);
        p++;

        printf("Do you want to insert another integer? (y/n): ");
        scanf(" %c", &choice); // Note the space before %c to consume the newline character.

    } while (choice == 'y' || choice == 'Y');

    printf("The inorder traversal of the binary tree is\n");
    inorder(root);
    printf("\nThe preorder traversal of the binary tree is\n");
    preorder(root);
    printf("\nThe postorder traversal of the binary tree is\n");
    postorder(root);

    int max_h = maxheight(root);
    printf("\nHeight of the tree is %d\n", max_h);

    int dia = diameter(root);
    printf("Diameter of the tree is %d\n", dia);

    printf("The amortized cost of BST is %f\n", p / c);

    return 0;
}
```

```
rohithsaidatta@rohithsaidatta-VirtualBox: ~/Documents/ADSANEW_LAB/lab 3
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/ADSANEW_LAB/lab 3$ gcc -o bst bst1.c -lm
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/ADSANEW_LAB/lab 3$ ./bst
Enter an integer to insert into the binary tree: 45
Do you want to insert another integer? (y/n): y
Enter an integer to insert into the binary tree: 30
Do you want to insert another integer? (y/n): y
Enter an integer to insert into the binary tree: 50
Do you want to insert another integer? (y/n): y
Enter an integer to insert into the binary tree: 25
Do you want to insert another integer? (y/n): y
Enter an integer to insert into the binary tree: 35
Do you want to insert another integer? (y/n): y
Enter an integer to insert into the binary tree: 60
Do you want to insert another integer? (y/n): y
Enter an integer to insert into the binary tree: 4
Do you want to insert another integer? (y/n): n
The inorder traversal of the binary tree is
4 25 30 35 45 50 60
The preorder traversal of the binary tree is
45 30 25 4 35 50 60
The postorder traversal of the binary tree is
4 25 35 30 60 50 45
Height of the tree is 4
Diameter of the tree is 6
The amortized cost of BST is 0.388889
```

Analysis:

Time Complexity:

Insertion (create):

- Best Case: O(log n) for balanced trees.
- Worst Case: O(n) for completely unbalanced trees.
- Average Case: O(log n) with random insertions in balanced trees.

Traversal (inorder, preorder, postorder):

- Always O(n) because each node is visited once.

Height Calculation (maxheight):

- Best Case: O(log n) for balanced trees.
- Worst Case: O(n) for completely unbalanced trees.
- Average Case: O(log n) with random insertions in balanced trees.

Diameter Calculation (diameter):

- Best Case: O(n) for balanced trees.
- Worst Case: O(n) for completely unbalanced trees.
- Average Case: Varies but often close to O(n) with random insertions.