Experiment No:6c

Title: Implementation of Search operation in Binary Search Tree in C.

Problem Statement: Implementation of Search operation in Binary Search Tree in C.

Insert()

Search()

Algorithm:

1. Start
2. IF ROOT -> DATA = ITEM OR ROOT = NULL  
       Return ROOT  
      ELSE  
      IF ROOT < ROOT -> DATA  
      Return search(ROOT -> LEFT, ITEM)  
     ELSE  
      Return search(ROOT -> RIGHT,ITEM)  
     [END OF IF]  
     [END OF IF]
3. Stop

CODE:

#include<iostream>

using namespace std;

struct node {

int d;

node \*left;

node \*right;

};

node\* CreateNode(int d) {

node \*newnode = new node;

newnode->d = d;

newnode->left = NULL;

newnode->right = NULL;

return newnode;

}

node\* InsertIntoTree(node\* root, int d) {

node \*temp = CreateNode(d);

node \*t = new node;

t = root;

if(root == NULL)

root = temp;

else {

while(t != NULL) {

if(t->d < d) {

if(t->right == NULL) {

t->right = temp;

break;

}

t = t->right;

} else if(t->d > d) {

if(t->left == NULL) {

t->left = temp;

break;

}

t = t->left;

}

}

}

return root;

}

void Search(node \*root, int d) {

int depth = 0;

node \*temp = new node;

temp = root;

while(temp != NULL) {

depth++;

if(temp->d == d) {

cout<<"\nitem found at depth: "<<depth;

return;

} else if(temp->d > d)

temp = temp->left;

else

temp = temp->right;

}

cout<<"\n item not found";

return;

}

int main() {

char ch;

int n, i, a[10] = {93, 53, 45, 2, 7, 67, 32, 26, 71, 76};

node \*root = new node;

root = NULL;

for (i = 0; i < 10; i++)

root = InsertIntoTree(root, a[i]);

up:

cout<<"\nEnter the Element to be searched: ";

cin>>n;

Search(root, n);

cout<<"\n\n\tDo you want to search more...enter choice(y/n)?";

cin>>ch;

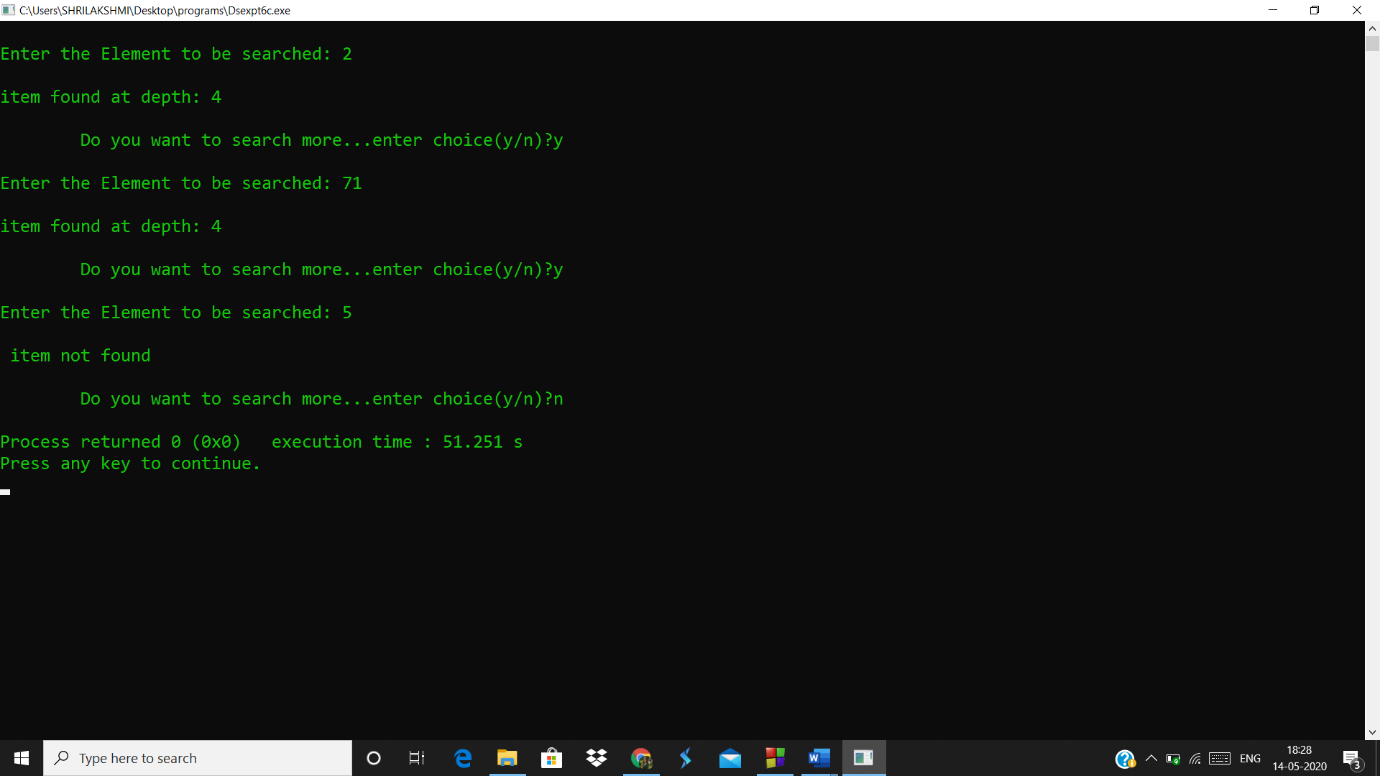
if(ch == 'y' || ch == 'Y')

goto up;

return 0;

}

Output:



Analysis(Limitations):

* The searching in binary tree is very fast as compared to one or two dimensional array.
* The time complexity increases as the nodes increase since to add a node we have to traverse until the leaf node on any one side.