THỰC HÀNH CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Họ tên: Khương Thanh Bình

MSSV:3123411034

Bài Tập Cây AVL

Cài đặt cây AVL chứa các số nguyên ( thêm, xóa, và các thao tác cây BST )

CODE

#include <iostream>

using namespace std;

struct Node

{

int info;

Node\* left;

Node\* right;

int height;

int count;

};

int myMax(int a, int b) {

return (a > b) ? a : b;

}

Node\* minValueNode(Node\* node) {

Node\* current = node;

while (current && current->left != NULL)

current = current->left;

return current;

}

int GetHeight(Node\* p) {

if (p == NULL)

return 0;

return p->height;

}

int GetBalance(Node\* p) {

if (p == NULL)

return 0;

return GetHeight(p->left) - GetHeight(p->right);

}

Node\* RightRotate(Node\* y) {

Node\* x = y->left;

Node\* T2 = x->right;

x->right = y;

y->left = T2;

y->height = myMax(GetHeight(y->left), GetHeight(y->right)) + 1;

x->height = myMax(GetHeight(x->left), GetHeight(x->right)) + 1;

return x;

}

Node\* LeftRotate(Node\* x) {

Node\* y = x->right;

Node\* T2 = y->left;

y->left = x;

x->right = T2;

x->height = myMax(GetHeight(x->left), GetHeight(x->right)) + 1;

y->height = myMax(GetHeight(y->left), GetHeight(y->right)) + 1;

return y;

}

Node\* insert(Node\* node, int key) {

if (node == NULL) {

Node\* newNode = new Node();

newNode->info = key;

newNode->left = newNode->right = NULL;

newNode->height = 1;

return newNode;

}

if (key < node->info)

node->left = insert(node->left, key);

else if (key > node->info)

node->right = insert(node->right, key);

else

return node;

node->height = myMax(GetHeight(node->left), GetHeight(node->right)) + 1;

int balance = GetBalance(node);

if (balance > 1 && key < node->left->info)

return RightRotate(node);

if (balance < -1 && key > node->right->info)

return LeftRotate(node);

if (balance > 1 && key > node->left->info) {

node->left = LeftRotate(node->left);

return RightRotate(node);

}

if (balance < -1 && key < node->right->info) {

node->right = RightRotate(node->right);

return LeftRotate(node);

}

return node;

}

Node\* Find(Node\* root, int key) {

if (root == NULL || root->info == key)

return root;

if (key < root->info)

return Find(root->left, key);

else

return Find(root->right, key);

}

Node\* Delete(Node\* root, int key) {

if (root == NULL)

return root;

if (key < root->info)

root->left = Delete(root->left, key);

else if (key > root->info)

root->right = Delete(root->right, key);

else {

if (root->left == NULL) {

Node\* temp = root->right;

delete root;

return temp;

}

else if (root->right == NULL) {

Node\* temp = root->left;

delete root;

return temp;

}

Node\* temp = minValueNode(root->right);

root->info = temp->info;

root->right = Delete(root->right, temp->info);

}

root->height = myMax(GetHeight(root->left), GetHeight(root->right)) + 1;

int balance = GetBalance(root);

if (balance > 1 && GetBalance(root->left) >= 0)

return RightRotate(root);

if (balance < -1 && GetBalance(root->right) <= 0)

return LeftRotate(root);

if (balance > 1 && GetBalance(root->left) < 0) {

root->left = LeftRotate(root->left);

return RightRotate(root);

}

if (balance < -1 && GetBalance(root->right) > 0) {

root->right = RightRotate(root->right);

return LeftRotate(root);

}

return root;

}

void DisplayNode(Node\* p, int i) {

for (int j = 1; j <= i; j++)

printf(" ");

printf("{%d}:%d\n", p->info, p->height);

}

void PreOrder(Node\* theRoot, int i) {

if (theRoot != NULL) {

DisplayNode(theRoot, i);

PreOrder(theRoot->left, i + 3);

PreOrder(theRoot->right, i + 3);

}

}

int main() {

Node\* root = NULL;

cout << "Test Case 1: Chen va Xoa cac phan tu" << endl;

root = insert(root, 10);

root = insert(root, 20);

root = insert(root, 30);

root = insert(root, 25);

root = insert(root, 5);

root = insert(root, 15);

cout << "Cay AVL truoc khi xoa:" << endl;

PreOrder(root, 0);

cout << "Tim kiem node co gia tri 15..." << endl;

Node\* foundNode = Find(root, 15);

if (foundNode)

cout << "Da tim thay node co gia tri " << foundNode->info << endl;

else

cout << "Khong tim thay node co gia tri 15!" << endl;

cout << "Xoa node co gia tri 20..." << endl;

root = Delete(root, 20);

cout << "Cay AVL sau khi xoa:" << endl;

PreOrder(root, 0);

root = NULL;

cout << "\nTest Case 2: Chen cac phan tu va tim kiem phan tu khong ton tai" << endl;

root = insert(root, 15);

root = insert(root, 10);

root = insert(root, 20);

root = insert(root, 25);

root = insert(root, 5);

root = insert(root, 30);

cout << "Cay AVL sau khi chen cac phan tu:" << endl;

PreOrder(root, 0);

cout << "Tim kiem node co gia tri 35..." << endl;

foundNode = Find(root, 35);

if (foundNode)

cout << "Da tim thay node co gia tri " << foundNode->info << endl;

else

cout << "Khong tim thay node co gia tri 35!" << endl;

return 0;

}

Run Code

->A screenshot of a computer program

AI-generated content may be incorrect.