LAPORAN PRAKTIKUM

PERTEMUAN 9

Tree

Nama :

Andika Rifki Pratama (2311104011)

Dosen :

Yudha Islami Sulistya,
S.Kom.,M.Kom.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

B. Soal Unguided

1. Modifikasi guided tree diatas dengan program menu menggunakan input data tree dari user dan berikan fungsi tambahan untuk menampilkan node child dan descendant dari node yang diinputkan!
2. Buatlah fungsi rekursif is_valid_bst(node, min_val, max_val) untuk memeriksa apakah suatu pohon memenuhi properti Binary Search Tree. Uji fungsi ini pada berbagai pohon, baik yang valid maupun tidak valid sebagai BST.
3. Buatlah fungsi rekursif cari_simpul_daun(node) untuk menghitung jumlah simpul daun dalam Binary Tree. Simpul daun adalah node yang tidak memiliki anak kiri maupun kanan.

Unguided > © Unguided3.cpp > ...

```cpp
#include <iostream>
using namespace std;

// BINARY TREE

struct Pohon
{
    char data;
    Pohon *left, *right, *parent;
};

Pohon *root, *baru;

void init()
{
    root = NULL;
}

bool isEmpty()
{
    return root == NULL;
}

void buatNode(char data)
{
    if (isEmpty())
    {
        root = new Pohon{data, NULL, NULL, NULL};
        cout << "\nNode " << data << " berhasil dibuat jadi root" << endl;
    }
    else
    {
        cout << "\nPohon sudah dibuat." << endl;
    }
}

// Tambah kiri
Pohon *insertLeft(char data, Pohon *node)
{
    if (isEmpty())
    {
        cout << "\nBuat tree dulu." << endl;
        return NULL;
    }

    if (node->left != NULL)
    {
        cout << "\nNode " << node->data << " sudah ada child kiri." << endl;
        return NULL;
    }

    baru = new Pohon{data, NULL, NULL, node};
    node->left = baru;
    cout << "\nNode " << data << " berhasil ditambahkan ke child kiri " << node->data << endl;
    return baru;
}

// Tambah kanan
Pohon *insertRight(char data, Pohon *node)
{
    if (isEmpty())
    {
        cout << "\nBuat tree dulu!" << endl;
        return NULL;
    }

    if (node->right != NULL)
    {
        cout << "\nNode " << node->data << " sudah ada child kanan." << endl;
        return NULL;
    }

    baru = new Pohon{data, NULL, NULL, node};
    node->right = baru;
    cout << "\nNode " << data << " berhasil ditambahkan ke child kanan " << node->data << endl;
    return baru;
}

// Update Tree
void update(char data, Pohon *node)
{
    if (isEmpty())
    {
        cout << "\nBuat tree dulu!" << endl;
        return;
    }

    if (!node)
    {
        cout << "\nNode yang ingin diganti tidak ada." << endl;
        return;
    }

    char temp = node->data;
    node->data = data;
    cout << "\nNode " << temp << " berhasil diubah jadi " << data << endl;
```

```cpp
void retrieve(Pohon *node)
{
    if (isEmpty())
    {
        cout << "\nBuat tree dulu." << endl;
        return;
    }

    if (!node)
    {
        cout << "\nNode yang ditunjuk tidak ada." << endl;
        return;
    }
    cout << "\nData node: " << node->data << endl;
}

// Search Data
void find(Pohon *node)
{
    if (isEmpty())
    {
        cout << "\nBuat tree dulu." << endl;
        return;
    }

    if (!node)
    {
        cout << "\nNode yang ditunjuk tidak ada." << endl;
        return;
    }

    cout << "\nData Node : " << node->data << endl;
    cout << "Root : " << root->data << endl;
    cout << "Parent : " << (node->parent ? node->parent->data : '(Tidak ada parent)') << endl;

    if (node->parent)
    {
        if (node->parent->left == node && node->parent->right)
            cout << "Sibling : " << node->parent->right->data << endl;
        else if (node->parent->right == node && node->parent->left)
            cout << "Sibling : " << node->parent->left->data << endl;
        else
            cout << "Sibling : (Tidak ada sibling)" << endl;
    }
}
bool is_valid_bst(Pohon *node, char min_val, char max_val)
{
    if (node == NULL)
        return true;

    if (node->data <= min_val || node->data >= max_val)
        return false;

    return is_valid_bst(node->left, min_val, node->data) && is_valid_bst(node->right, node->data, max_val);
}

int cari_simpul_daun(Pohon *node)
{
    if (node == NULL)
        return 0;

    if (node->left == NULL && node->right == NULL)
        return 1;

    return cari_simpul_daun(node->left) + cari_simpul_daun(node->right);
}

void inputNode()
{
    char data, parentData;
    int choice;
    Pohon *current = root;

    if (isEmpty())
    {
        cout << "Masukkan nama node untuk dijadikan root: ";
        cin >> data;
        buatNode(data);
    }

    do
    {
        cout << "\n== Pilihan Input Node == " << endl;
        cout << "1. Tambah Child Kiri" << endl;
        cout << "2. Tambah Child Kanan" << endl;
        cout << "3. Update Node" << endl;
        cout << "4. Retrieve Node" << endl;
        cout << "5. Find Node" << endl;
        cout << "6. Cek apakah BST?" << endl;
        cout << "7. Jumlah simpul daun" << endl;
        cout << "8. Keluar" << endl;
        cout << "Pilih: ";
        cin >> choice;

        if (choice == 1 || choice == 2 || choice == 3 || choice == 4 || choice == 5)
```

```
            cout << "Masukkan data node parent: ";
            cin >> parentData;

            if (parentData == root->data)
                current = root;
            else if (root->left && root->left->data == parentData)
                current = root->left;
            else if (root->right && root->right->data == parentData)
                current = root->right;
            else
                cout << "Parent node tidak ditemukan!" << endl;

            if (!current) continue;
        }

        switch (choice)
        {
        case 1:
            cout << "Masukkan data untuk child kiri: ";
            cin >> data;
            insertLeft(data, current);
            break;
        case 2:
            cout << "Masukkan data untuk child kanan: ";
            cin >> data;
            insertRight(data, current);
            break;
        case 3:
            cout << "Masukkan data baru untuk update node: ";
            cin >> data;
            update(data, current);
            break;
        case 4:
            retrieve(current);
            break;
        case 5:
            find(current);
            break;
     case 6:
            if (is_valid_bst(root, CHAR_MIN, CHAR_MAX))
                cout << "\nPohon adalah Binary Search Tree yang valid." << endl;
            else
                cout << "\nPohon bukan Binary Search Tree yang valid." << endl;
            break;
    case 7:
            cout << "\nJumlah simpul daun: " << cari_simpul_daun(root) << endl;
            break;
    case 8:
            cout << "Keluar." << endl;
            break;
        default:
            cout << "Pilihan tidak valid!" << endl;
        }

    } while (choice != 8);
}

int main()
{
    init();
    inputNode();
    return 0;
}
```

Penjelasan :

Ditambahkan fungsi inputNode untuk memunculkan menu yang dapat dipilih oleh user. Pada method ini terdapat fungsi untuk memeriksa apakah tree kosong atau tidak, kemudian apabila tidak kosong maka akan memunculkan menu dari menambah child, mengupdate, memunculkan bahkan mencari node dan BST hingga jumlah simpul daun. Kemudian terdapat penggunaan if else untuk memeriksa apakah terdapat data dari parent node. Selanjutnya terdapat fungsi switch case yang digunakan untuk memunculkan pilihan dan juga untuk memilih menu. Kemudian terdapat method is_valid_bst untuk ememriksa apakah nodenya BST atau tidak dengan cara memeriksa apakah data yang lebih dari value minimum.