Laporan Praktikum

Kelompok 5

BAB 5 Binary Tree dan Binary Search Tree

Struktur Data Kelas E081

Disusun oleh :

Azka Avicenna Rasjid / 20081010115

Farkhan / 20081010060

Kuncoro Ariadi / 20081010096

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”

JAWA TIMUR

2022

LEMBAR PERNYATAAN ORISINALITAS PRAKTIKUM

Yang bertanda tangan di bawah ini :

Nama Anggota 1 : Azka Avicenna Rasjid

NPM : 20081010115

Angkatan : 2020

Nama Anggota 2 : Farkhan

NPM : 20081010060

Angkatan : 2020

Nama Anggota 3 : Kuncoro Ariadi

NPM : 20081010096

Angkatan : 2020

Dengan ini menyatakan bahwa praktikum yang kami buat merupakan benar-benar hasil praktikum kami dan bukan merupakan tiruan/plagiarisme atau hasil karya orang lain. Apabila dikemudian hari kami melanggar pernyataan ini maka kami bersedia menerima sanksi yang diberikan.

Demikian pernyataan ini dibuat dengan sebenar-benarnya tanpa adanya paksaan dari pihak manapun.

Surabaya, 16 Desember 2022

Anggota 1	Anggota 2	Anggota 3
Azka Avicenna Rasjid	Farkhan	Kuncoro Ariadi
NPM. 20081010060	NPM. 20081010115	NPM. 20081010096

SOAL PRAKTIKUM

1. Buat program Binary Search Tree(BST) dan AVL Tree sesuai dengan topik kalian masing-masing. Untuk operasi yang ditampilkan adalah:

   a. Insert menggunakan BST

   b. Delete menggunakan AVL Tree

   c. Search

   d. View

KODE PROGRAM

Bahasa pemrograman yang digunakan: Bahasa yang digunakan adalah C++

No 1 :

```
 1  #include<bits/stdc++.h>
 2  #include <conio.h>
 3  using namespace std;
 4
 5  // STRUCT KOMIK
 6  struct komik {
 7      int kode_komik, tahun;
 8      char judul[50], penerbit[50], pengarang[50];
 9      struct komik *next;
10  };
11  struct komik* front = NULL;
12  struct komik* rear = NULL;
13  struct komik* temp;
14
15  // buat node AVL tree
16  class Node
17  {
18         public:
19         int key;
20         Node *left;
21         Node *right;
22         int height;
23  };
```

Kode di atas diawali library yang dibutuhkan. Baris ke 5-10 mendefinisikan struct komik yang berisi kode komik, tahun, judul, penerbit, dan pengarang. Kemudian baris ke 11-13 mendefinisikan struct komik front dan rear adalah NULL dan struct komik adalah temp. Baris ke 16-23 mendefinisikan class Node untuk AVL tree.

```
24  Node* newNode(int key) {
25         Node* node = new Node();
26         node->key = key;
27         node->left = NULL;
28         node->right = NULL;
29         node->height = 1; // node baru ditambahkan
30                                    // di daun
31         return(node);
32  }
33  // fungsi untuk mendapatkan tinggi node
34  int height(Node *N) {
35         if (N == NULL)
36                return 0;
37         return N->height;
```

```
}
// fungsi untuk melakukan rotasi kanan
Node *rightRotate(Node *y) {
       Node *x = y->left;
       Node *T2 = x->right;
       // Lakukan rotasi
       x->right = y;
       y->left = T2;
       // Update tinggi
       y->height = max(height(y->left),
                                height(y->right)) + 1;
       x->height = max(height(x->left),
                                height(x->right)) + 1;
       // Return new root
       return x;
}
// fungsi untuk melakukan rotasi kiri
Node *leftRotate(Node *x) {
       Node *y = x->right;
       Node *T2 = y->left;
       // Lakukan rotasi
       y->left = x;
       x->right = T2;
       // Update tinggi
       x->height = max(height(x->left),
                                height(x->right)) + 1;
       y->height = max(height(y->left),
                                height(y->right)) + 1;
       // Return new root
       return y;
}
// fungsi untuk mendapatkan balance factor
int getBalance(Node *N) {
       if (N == NULL)
              return 0;
       return height(N->left) - height(N->right);
}
// fungsi min value node
Node * minValueNode(Node* node) {
    Node* current = node;
    /* loop down to find the leftmost leaf */
    while (current->left != NULL)
        current = current->left;
    return current;
}
// fungsi insert BST
Node* insertBST(Node* node, int key) {
       /* 1. normal BST insert */
       if (node == NULL)
              return(newNode(key));
       if (key < node->key)
              node->left = insertBST(node->left, key);
       else if (key > node->key)
              node->right = insertBST(node->right, key);
```

```
 92        else // nilai key sama dengan node yang ada
 93              return node;
 94        /* 2. Update tinggi node */
 95        node->height = 1 + max(height(node->left),
 96                                        height(node->right));
 97    return node;
 98  }
 99  // START SHOW ALL
100  void tampil(int kode_komik) {
101      int flag = 0;
102      temp = front;
103      while (temp != NULL) {
104          if (temp->kode_komik == kode_komik) {
105              flag = 1;
106              break;
107          }
108          temp = temp->next;
109      }
110      if (flag == 1) {
111          cout << "\nKode Komik          : " << temp->kode_komik << endl;
112          cout << "Judul Komik           : " << temp->judul << endl;
113          cout << "Penerbit Komik        : " << temp->penerbit << endl;
114          cout << "Pengarang Komik       : " << temp->pengarang << endl;
115          cout << "Tahun Terbit Komik    : " << temp->tahun << endl;
116      }
117
118      cout << endl;
119  }
```

Kode di atas berisi definisi dari fungsi-fungsi. Baris ke 24-32 berisi definisi node newnode. Baris ke 34-38 berisi fungsi height untuk mendapatkan tinggi node. Baris ke 39-53 berisi fungsi right rotate untuk melakukan rotasi kanan. Baris ke 54-68 berisi fungsi left rotate untuk melakukan rotasi kiri. Baris ke 69-74 berisi fungsi getbalance untuk mendapatkan balance node. Baris ke 75-82 berisi fungsi minValueNode untuk minimal value node. Baris 83-98 berisi fungsi insertBST untuk insert BST. Baris ke 99-119 berisi fungsi tampil untuk menampilkan data yang telah disimpan.

```
120  // fungsi untuk melakukan preorder traversal
121  void preOrder(Node *root) {
122        if(root != NULL)
123        {
124          // panggil fungsi tampil
125          tampil(root->key);
126               preOrder(root->left);
127               preOrder(root->right);
128        }
129  }
130
131  // fungsi untuk melakukan inorder traversal
132  void inOrder(Node *root) {
```

```
    if(root != NULL) {
        inOrder(root->left);
        // panggil fungsi tampil
        tampil(root->key);
        inOrder(root->right);
    }
}

// fungsi untuk melakukan postorder traversal
void postOrder(Node *root) {
    if(root != NULL) {
        postOrder(root->left);
        postOrder(root->right);
        // panggil fungsi tampil
        tampil(root->key);
    }
}
// END SHOW ALL

// START DELETE
Node* deleteNode(Node* root, int kode_komik) {
    // STEP 1: NORMAL BST DELETE
    if (root == NULL)
        return root;

    // Jika kunci yang akan dihapus lebih kecil dari
    // root->key, maka akan berada di subtree kiri
    if (kode_komik < root->key)
        root->left = deleteNode(root->left, kode_komik);

    // Jika kunci yang akan dihapus lebih besar dari
    // root->key, maka akan berada di subtree kanan
    else if (kode_komik > root->key)
        root->right = deleteNode(root->right, kode_komik);

    // Jika kunci yang akan dihapus sama dengan root->key
    else {
        // node dengan satu anak atau tanpa anak
        if ((root->left == NULL) || (root->right == NULL)) {
            Node *temp = root->left ? root->left : root->right;

            // Tidak ada anak
            if (temp == NULL) {
                temp = root;
                root = NULL;
            }
            else // Ada satu anak
                *root = *temp; // Copy isi dari anak
            free(temp);
        }
        else {
            // node dengan dua anak: dapatkan inorder successor
            // (nilai terkecil di subtree kanan)
            Node* temp = minValueNode(root->right);
```

```
187
188             // Copy isi inorder successor ke node yang akan dihapus
189             root->key = temp->key;
190
191             // Hapus inorder successor
192             root->right = deleteNode(root->right, temp->key);
193         }
194     }
195
196     // Jika tree hanya memiliki satu node
197     if (root == NULL)
198         return root;
199
200     // STEP 2: UPDATE TINGGI NODE
201     root->height = 1 + max(height(root->left),
202                            height(root->right));
203
204     // STEP 3: GET BALANCE FACTOR
205     int balance = getBalance(root);
206
207     // Jika node tidak balance, ada 4 kasus
208
209     // Left Left Case
210     if (balance > 1 && getBalance(root->left) >= 0)
211         return rightRotate(root);
212
213     // Left Right Case
214     if (balance > 1 && getBalance(root->left) < 0) {
215         root->left = leftRotate(root->left);
216         return rightRotate(root);
217     }
218
219     // Right Right Case
220     if (balance < -1 && getBalance(root->right) <= 0)
221         return leftRotate(root);
222
223     // Right Left Case
224     if (balance < -1 && getBalance(root->right) > 0) {
225         root->right = rightRotate(root->right);
226         return leftRotate(root);
227     }
228
229     return root;
230 }
231 // END DELETE
```

Kode di atas berisi definisi dari fungsi-fungsi yang akan dijalankan. Baris ke 120-129 berisi fungsi preorder untuk melakukan preorder traversal. Baris 131-139 berisi fungsi inOrder untuk melakukan inorder traversal. Baris ke 141-149 berisi fungsi postOrder untuk melakukan postorder traversal. Pada baris ke 153 memulai node delete dengan parameter node root serta int kode komik, update tinggi node, dan get balance factor. Node delete terdiri dari beberapa kasus yaitu normal BST Delete, kunci yang dihapus lebih kecil dari root key, kunci yang dihapus lebih besar dari root key, kunci yang dihapus sama dengan root key, node dengan dua

anak, dan tree dengan satu node. Kemudian update tinggi node dan get balance factor yang jika node tidak balance akan terjadi 4 kasus.

```
232 void tambah(int kode_komik) {
233     char judul[50];
234     char penerbit[50];
235     char pengarang[50];
236     int tahun;
237 
238     cout << ">> Masukkan Judul Komik          : "; cin >> judul;
239     cout << ">> Masukkan Penerbit Komik : "; cin >> penerbit;
240     cout << ">> Masukkan Pengarang Komik      : "; cin >> pengarang;
241     cout << ">> Masukkan Tahun Terbit Komik   : "; cin >> tahun;
242 
243     if (rear == NULL) {
244         rear = (struct komik *)malloc(sizeof(struct komik));
245         rear->next = NULL;
246         rear->kode_komik = kode_komik;
247         strcpy(rear->judul, judul);
248         strcpy(rear->penerbit, penerbit);
249         strcpy(rear->pengarang, pengarang);
250         rear->tahun = tahun;
251         front = rear;
252     } else {
253         temp=(struct komik *)malloc(sizeof(struct komik));
254         rear->next = temp;
255         temp->kode_komik = kode_komik;
256         strcpy(temp->judul, judul);
257         strcpy(temp->penerbit, penerbit);
258         strcpy(temp->pengarang, pengarang);
259         temp->tahun = tahun;
260         temp->next = NULL;
261         rear = temp;
262     }
263     cout << "\nKomik Berhasil Ditambahkan" << endl << endl;
264 }
265 
266 // fungsi cari komik
267 void cari() {
268     int kode_komik;
269     int flag = 0;
270     temp = front;
271     if (temp == NULL) {
272         cout << "Gudang kosong, masukkan data komik terlebih dahulu..." <<
273 endl;
274         return;
275     }
276     cout << "Masukkan Kode Komik : ";
277     cin >> kode_komik;
278     while (temp != NULL) {
279         if (temp->kode_komik == kode_komik) {
280             flag = 1;
```

```
            break;
        }
        temp = temp->next;
    }
    if (flag == 1) {
        cout << "\nKode Komik          : " << temp->kode_komik << endl;
        cout << "Judul Komik           : " << temp->judul << endl;
        cout << "Penerbit Komik        : " << temp->penerbit << endl;
        cout << "Pengarang Komik       : " << temp->pengarang << endl;
        cout << "Tahun Terbit Komik    : " << temp->tahun << endl;
    } else {
        cout << "Kode Komik tidak ditemukan..." << endl;
    }
    cout << endl;
}
```

Kode di atas berisi definisi dari fungsi-fungsi yang dibutuhkan. Baris ke 232-264 berisi fungsi tambah untuk fungsi dan tampilan ketika user memilih opsi tambah data dengan metode BST. Baris ke 266-295 berisi fungsi cari untuk fungsi dan tampilan ketika user ingin memilih komik yang diinginkan.

```
int main() {
    int pilihan;

    cout << "==============================" << endl;
       cout << "=== Program Pendataan Komik ===" << endl;
       cout << "==============================" << endl << endl;

       cout << "DISUSUN OLEH KELOMPOK 5 " << endl;
       cout << "Azka Avicenna Rasjid    [20081010115]" << endl;
       cout << "Farkhan                 [20081010060]" << endl;
       cout << "Kuncoro Ariadi          [20081010096]" << endl << endl;

       Node *root = NULL;

    do {
        cout << "Pilihan menu yang tersedia : " << endl;
        cout << "1. Tambah Komik (BST)" << endl;
        cout << "2. Hapus Komik (AVL)" << endl;
        cout << "3. Cari Komik" << endl;
        cout << "4. Lihat Komik" << endl;
        cout << "5. Keluar" << endl << endl;
        cout << ">> Masukkan Pilihan : "; cin >> pilihan;

        cout << endl;
        switch (pilihan) {
            case 1:
                // insert BST
                int kode_komik;
                cout << ">> Masukkan Kode Komik           : "; cin >>
kode_komik;
```

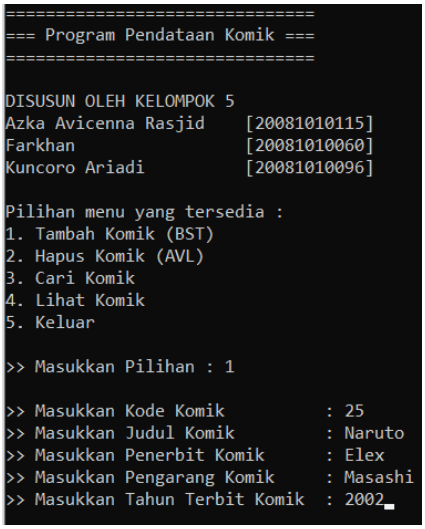
```
                tambah(kode_komik);
                root = insertBST(root, kode_komik);
                break;
            case 2:
                // delete AVL
                int kode_komik_hapus;
                cout << "Masukkan kode komik yang akan dihapus: "; cin >>
kode_komik_hapus;
                root = deleteNode(root, kode_komik_hapus);
                getch();
                break;
            case 3: cari(); getch(); break;
            case 4:
                // lakukan pilihan traversal
                int pilihan_traversal;
                cout << "Pilihan traversal yang tersedia : " << endl;
                cout << "1. Preorder" << endl;
                cout << "2. Inorder" << endl;
                cout << "3. Postorder" << endl << endl;
                cout << ">> Masukkan Pilihan : "; cin >> pilihan_traversal;
                cout << endl;
                switch (pilihan_traversal) {
                    case 1:
                        cout << "Data komik yang ada di gudang: " << endl;
                        preOrder(root);
                        getch();
                        break;
                    case 2:
                        cout << "Data komik yang ada di gudang: " << endl;
                        inOrder(root);
                        getch();
                        break;
                    case 3:
                        cout << "Data komik yang ada di gudang: " << endl;
                        postOrder(root);
                        getch();
                        break;
                    default:
                        cout << "Pilihan tidak tersedia" << endl;
                }
                break;
            case 5:
                cout << "Terima Kasih Telah Menggunakan Program Ini..." <<
endl;
                getch();
                break;
            default:
                cout << "Pilihan tidak tersedia" << endl;
        }
        system("cls");
    } while (pilihan != 5);

    return 0;
}
```

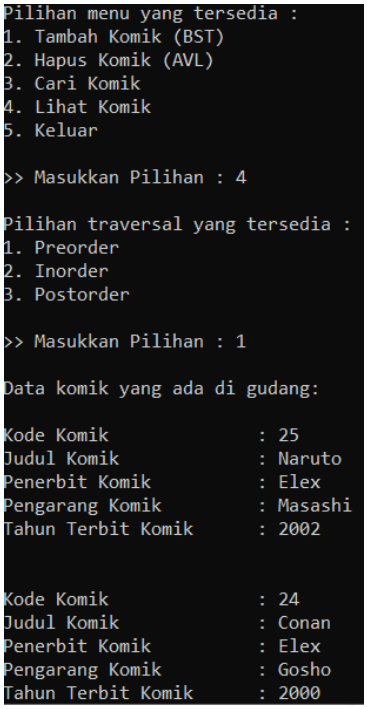
380	

Kode di atas berisi fungsi main dari program. Diawali dengan deklarasi int pilihan kemudian kode untuk tampilan pada user. Kemudian ada definisi node *root adalah NULL. Pada baris ke 310-380 berisi tampilan utama untuk user yang berisi fungsi do dan switch case yang user akan diminta untuk memilih fungsi yang diinginkan beserta metode yang ingin digunakan.

Output:

No 1 :





Link google drive:
https://drive.google.com/drive/folders/1Fh3Kw3U7davV4uUy7TrslpYo_PLBx7Uc?usp=share_link