LAPORAN PRAKTIKUM ALGORITMA DASAR PEMROGRAMAN

JOBSHEET 14

SHAFIQA NABILA MAHARANI KHOIRUNNISA

244107020221

TI – 1B

14.2.1 Percobaan 1

```
Daftar semua mahasiswa (in order traversal):
NIM: 244160185Nama: Candra Kelas: C IPK: 3.21
NIM: 244160220Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160121Nama: Ali Kelas: A IPK: 3.57
NIM: 244160227Nama: Badar Kelas: B IPK: 3.85

Pencarian data mahasiswa:
Cari mahasiswa dengan ipk: 3.54 : Ditemukan
Cari mahasiswa dengan ipk: 3.22 : Tidak ditemukan

Daftar semua mahasiswa setelah penambahan 3 mahasiswa:
NIM: 244160185Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160220Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160121Nama: Ali Kelas: A IPK: 3.57
NIM: 244160170Nama: Fizi Kelas: B IPK: 3.6
NIM: 244160131Nama: Devi Kelas: A IPK: 3.72
NIM: 244160227Nama: Badar Kelas: B IPK: 3.85

PreOrder Traversal:
NIM: 244160121Nama: Ali Kelas: A IPK: 3.57
NIM: 244160185Nama: Candra Kelas: C IPK: 3.21
NIM: 244160220Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160205Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160227Nama: Badar Kelas: B IPK: 3.85
NIM: 244160131Nama: Devi Kelas: A IPK: 3.72
NIM: 244160170Nama: Fizi Kelas: B IPK: 3.6
```

```
PostOrder Traversal:
NIM: 244160205Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160220Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160185Nama: Candra Kelas: C IPK: 3.21
NIM: 244160170Nama: Fizi Kelas: B IPK: 3.6
NIM: 244160131Nama: Devi Kelas: A IPK: 3.72
NIM: 244160227Nama: Badar Kelas: B IPK: 3.85
NIM: 244160121Nama: Ali Kelas: A IPK: 3.57
Jika 2 anak, current =
NIM: 244160170Nama: Fizi Kelas: B IPK: 3.6

Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order traversal):NIM: 244160185Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160220Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160170Nama: Fizi Kelas: B IPK: 3.6
NIM: 244160131Nama: Devi Kelas: A IPK: 3.72
NIM: 244160227Nama: Badar Kelas: B IPK: 3.85
PS C:\Users\fika\MATKUL SEMESTER 2\praktikum-ASD\Jobsheet 14>
```

14.2.2 Pertanyaan Percobaan

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?

- Karena BST memiliki sifat terurut dimana nilai di left child selalu lebih kecil dari parent dan nilai di right child selalu lebih besar dari parent. Ini memungkinkan pencarian dilakukan dengan efisiensi O(log n) dalam kasus tree yang seimbang, karena setiap langkah pencarian bisa mengurangi kemungkinan pencarian menjadi setengahnya.

2. Untuk apakah di class Node, kegunaan dari atribut left dan right?

- Atribut left dan right digunakan untuk menyimpan referensi ke node anak kiri dan kanan dari node saat ini, yang membentuk struktur pohon biner dimana setiap node maksimal memiliki 2 anak.

3. a. Untuk apakah kegunaan dari atribut root di dalam class BinaryTree?

- Root adalah node paling atas/paling pertama dalam pohon yang menjadi titik awal semua operasi pada tree.

b. Ketika objek tree pertama kali dibuat, apakah nilai dari root?

- Saat tree pertama dibuat, root bernilai null (karena tree masih kosong).

4. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?

- Jika tree kosong, node baru akan langsung dijadikan sebagai root tanpa perlu membandingkan nilai dengan node lain.

5. Perhatikan method add(), di dalamnya terdapat baris program seperti di bawah ini. Jelaskan secara detil **untuk apa baris program tersebut?**

- parent = current: Menyimpan node saat ini sebagai parent sebelum berpindah ke child

- Jika IPK mahasiswa baru lebih kecil dari current, pindah ke left child

- Jika left child null, tambahkan node baru sebagai left child dari parent

- Jika IPK lebih besar, lakukan hal serupa untuk right child

- Proses ini berulang sampai menemukan posisi yang tepat untuk node baru

6. Jelaskan langkah-langkah pada method delete() saat menghapus sebuah node yang memiliki dua anak. Bagaimana method getSuccessor() membantu dalam proses ini?

- Cari successor (node terkecil di subtree kanan) menggunakan getSuccessor()

- Ganti node yang akan dihapus dengan successor

- Atur pointer left dari successor ke left child node yang dihapus

- Method getSuccessor() membantu menemukan pengganti yang tepat untuk mempertahankan sifat BST

4.3 Kegiatan Praktikum 2

```
Daftar semua mahasiswa (in order traversal):
NIM: 244160185Nama: Candra Kelas: C IPK: 3.21
NIM: 244160220Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160121Nama: Ali Kelas: A IPK: 3.57
NIM: 244160227Nama: Badar Kelas: B IPK: 3.85
```

14.3.2 Pertanyaan Percobaan

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class BinaryTreeArray?

- data: Array untuk menyimpan elemen-elemen tree

- idxLast: Indeks terakhir yang berisi data valid dalam array

2. Apakah kegunaan dari method populateData()?

- Untuk menginisialisasi data tree dari array yang sudah ada dan mengatur idxLast yang menunjukkan batas data valid.

3. Apakah kegunaan dari method traverseInOrder()?

- Untuk mengunjungi dan menampilkan semua node dalam tree secara in-order (left-root-right) dengan representasi array.

4. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masing-masing?

- Left child: 2*2 + 1 = indeks 5

- Right child: 2*2 + 2 = indeks 6

5. Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?

- Menunjukkan bahwa data valid dalam array ada sampai indeks ke-6 (7 elemen pertama), sedangkan indeks setelahnya (7,8,9) bernilai null.

6. Mengapa indeks 2*idxStart+1 dan 2*idxStart+2 digunakan dalam pemanggilan rekursif, dan apa kaitannya dengan struktur pohon biner yang disusun dalam array?

Ini adalah rumus matematis untuk menghitung posisi left dan right child dalam representasi array:

- Left child = 2*indeksParent + 1
- Right child = 2*indeksParent + 2
  Ini memungkinkan representasi pohon biner lengkap dalam array secara efisien.

Tugas Praktikum

1. **Buat method di dalam class BinaryTree00 yang akan menambahkan node dengan cara rekursif (addRekursif()).**

```
public void addRekursif(Mahasiswa20 mahasiswa) {
root = addRekursif(root, mahasiswa);
}

public Node20 addRekursif(Node20 current, Mahasiswa20 mahasiswa) {
    if (current == null) {
        return new Node20(mahasiswa);
}

if (mahasiswa.ipk < current.mahasiswa.ipk) {
    current.left = addRekursif(current.left, mahasiswa);
} else if (mahasiswa.ipk > current.mahasiswa.ipk) {
    current.right = addRekursif(current.right, mahasiswa);
}
```

2. **Buat method di dalam class BinaryTree00 untuk menampilkan data mahasiswa dengan IPK paling kecil dan IPK yang paling besar (cariMinIPK() dan cariMaxIPK()) yang ada di dalam binary search tree.**

```
public Mahasiswa20 cariMinIPK() {
if (isEmpty()) {
    return null;
}
Node20 current = root;
while (current.left != null) {
    current = current.left;
}
return current.mahasiswa;
}

public Mahasiswa20 cariMaxIPK() {
    if (isEmpty()) {
        return null;
    }
    Node20 current = root;
    while (current.right != null) {
        current = current.right;
    }
    return current.mahasiswa;
}
```

3. **Buat method dalam class BinaryTree00 untuk menampilkan data mahasiswa dengan IPK di atas suatu batas tertentu, misal di atas 3.50 (tampilMahasiswaIPKdiAtas(double ipkBatas)) yang ada di dalam binary search tree.**

```
public void tampilMahasiswalPKdiAtas(double ipkBatas) {
tampilMahasiswalPKdiAtas(root, ipkBatas);
}

public void tampilMahasiswalPKdiAtas(Node20 node, double ipkBatas) {
    if (node != null) {
        tampilMahasiswalPKdiAtas(node.left, ipkBatas);
        if (node.mahasiswa.ipk > ipkBatas) {
            node.mahasiswa.tampilInformasi();
        }
        tampilMahasiswalPKdiAtas(node.right, ipkBatas);
    }
}
```

4. **Modifikasi class BinaryTreeArray00 di atas, dan tambahkan : • method add(Mahasiswa data) untuk memasukan data ke dalam binary tree • method traversePreOrder()**

```
void populateData(Mahasiswa20[] dataMhs, int idxLast) {
    this.dataMahasiswa = dataMhs;
    this.idxLast = idxLast;
}

void traverseInOrder(int idxStart) {
    if (idxStart <= idxLast) {
        if (dataMahasiswa[idxStart] != null) {    Merge this if staten
            traverseInOrder(2 * idxStart + 1);
            dataMahasiswa[idxStart].tampilInformasi();
            traverseInOrder(2 * idxStart + 2);
        }
    }
}

// Method tambahan untuk tugas praktikum
void add(Mahasiswa20 data) {
    if (idxLast == dataMahasiswa.length - 1) {
        System.out.println(x:"Array penuh");    Replace this use of Sy
        return;
    }
    dataMahasiswa[++idxLast] = data;
}

void traversePreOrder(int idxStart) {
    if (idxStart <= idxLast) {
        if (dataMahasiswa[idxStart] != null) {    Merge this if staten
            dataMahasiswa[idxStart].tampilInformasi();
            traversePreOrder(2 * idxStart + 1);
            traversePreOrder(2 * idxStart + 2);
        }
    }
}
```