LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

TREE

AHMAD SOFYAN BADAWI

244107020473

KELAS TI-1B

PRODI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

Percobaan 1

Langkah-langkah:

1. Di dalam class Mahasiswa04, deklarasikan atribut sesuai dengan diagram class Mahasiswa di atas. Tambahkan juga konstruktor dan method sesuai diagram di atas.
2. Di dalam class Node04, tambahkan atribut data, left dan right, serta konstruktor default dan berparameter.
3. Di dalam class BinaryTree04, tambahkan atribut root.

```
Node04 root;
```

4. Tambahkan konstruktor dan method isEmpty() di dalam class BinaryTree04.

```
BinaryTree04() {
    root = null;
}

public boolean isEmpty() {
    return root == null;
}
```

5. Tambahkan method add() di dalam class BinaryTree04. Node ditambahkan di binary search tree pada posisi sesuai dengan besar nilai IPK mahasiswa. Di bawah ini proses penambahan node tidak dilakukan secara rekursif, agar lebih mudah dilihat alur proses penambahan node dalam tree. Sebenarnya dengan proses rekursif penulisan kode akan lebih efisien.

```
public void add(Mahasiswa04 mahasiswa) {
    Node04 newNode = new Node04(mahasiswa);
    if (isEmpty()) {
        root = newNode;
    } else {
        Node04 current = root;
        Node04 parent = null;
        while (true) {
            parent = current;
            if (mahasiswa.ipk < current.mahasiswa.ipk) {
                current = current.left;
                if (current == null) {
                    parent.left = newNode;
                    return;
                }
            } else {
                current = current.right;
                if (current == null) {
                    parent.right = newNode;
                    return;
                }
            }
        }
    }
}
```

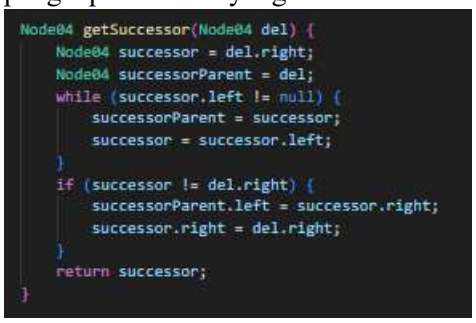
6. Tambahkan method find()

```
public boolean find(double ipk) {
    boolean result = false;
    Node04 current = root;
    while (current != null) {
        if (current.mahasiswa.ipk == ipk) {
            result = true;
            break;
        } else if (ipk > current.mahasiswa.ipk) {
            current = current.right;
        } else {
            current = current.left;
        }
    }
    return result;
}
```

7. Tambahkan method traversePreOrder(), traverseInOrder() dan traversePostOrder(). Method traverse digunakan untuk mengunjungi dan menampilkan node-node dalam binary tree, baik dalam mode pre-order, in-order maupun post-order.

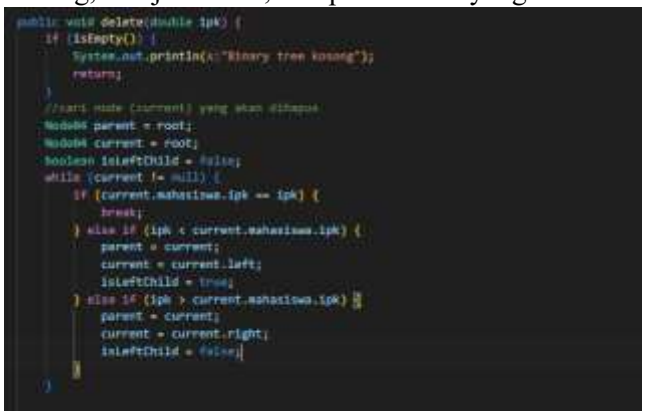
```
public void traversePreOrder(Node04 node) {
    if (node != null) {
        node.mahasiswa.tampilInformasi();
        traversePreOrder(node.left);
        traversePreOrder(node.right);
    }
}

public void traverseInOrder(Node04 node) {
    if (node != null) {
        traverseInOrder(node.left);
        node.mahasiswa.tampilInformasi();
        traverseInOrder(node.right);
    }
}

public void traversePostOrder(Node04 node) {
    if (node != null) {
        traversePostOrder(node.left);
        traversePostOrder(node.right);
        node.mahasiswa.tampilInformasi();
    }
}
```

8. Tambahkan method getSuccessor(). Method ini akan digunakan ketika proses penghapusan node yang memiliki 2 child.



```
Node04 getSuccessor(Node04 del) {
    Node04 successor = del.right;
    Node04 successorParent = del;
    while (successor.left != null) {
        successorParent = successor;
        successor = successor.left;
    }
    if (successor != del.right) {
        successorParent.left = successor.right;
        successor.right = del.right;
    }
    return successor;
}
```

9. Tambahkan method delete(). Di dalam method delete tambahkan pengecekan apakah tree kosong, dan jika tidak, cari posisi node yang akan dihapus.



```
public void delete(double ipk) {
    if (isEmpty()) {
        System.out.println("Binary tree kosong");
        return;
    }
    //cari node (current) yang akan dihapus
    Node04 parent = root;
    Node04 current = root;
    boolean isLeftChild = false;
    while (current != null) {
        if (current.mahasiswa.ipk == ipk) {
            break;
        } else if (ipk < current.mahasiswa.ipk) {
            parent = current;
            current = current.left;
            isLeftChild = true;
        } else if (ipk > current.mahasiswa.ipk) {
            parent = current;
            current = current.right;
            isLeftChild = false;
        }
    }
```

10. Kemudian tambahkan proses penghapusan di dalam method delete() terhadap node current yang telah ditemukan.

```
    //penghapusan
    if (current == null) {
        System.out.println(x:"Data tidak ditemukan");
        return;
    } else {
        //jika tidak ada anak (leaf), maka node dihapus
        if (current.left == null && current.right == null) {
            if (current == root) {
                root = null;
            } else {
                if (isLeftChild) {
                    parent.left = null;
                } else {
                    parent.right = null;
                }
            }
        } else if (current.left == null) { //jika hanya punya 1 anak (kanan)
            if (current == root) {
                root = current.right;
            } else {
                if (isLeftChild) {
                    parent.left = current.right;
                } else {
                    parent.right = current.right;
                }
            }
```

```
        } else if (current.right == null) { //jika hanya punya 1 anak (kiri)
            if (current == root) {
                root = current.left;
            } else {
                if (isLeftChild) {
                    parent.left = current.left;
                } else {
                    parent.right = current.left;
                }
            }
        } else { //jika punya 2 anak
            Node04 successor = getSuccessor(current);
            System.out.println(x:"Jika 2 anak, current = ");
            successor.mahasiswa.tampilInformasi();
            if (current == root) {
                root = successor;
            } else {
                if (isLeftChild) {
                    parent.left = successor;
                } else {
                    parent.right = successor;
                }
            }
            successor.left = current.left;
        }
    }
```

11. Buka class BinaryTreeMain04 dan tambahkan method main() kemudian tambahkan kode berikut ini:

```java
public static void main(String[] args) {
    BinaryTree04 bst = new BinaryTree04();

    bst.add(new Mahasiswa04(nim:"244160121", nama:"Ali", kelas:"A", ipk:3.57));
    bst.add(new Mahasiswa04(nim:"244160221", nama:"Badar", kelas:"B", ipk:3.85));
    bst.add(new Mahasiswa04(nim:"244160185", nama:"Candra", kelas:"C", ipk:3.21));
    bst.add(new Mahasiswa04(nim:"244160220", nama:"Dewi", kelas:"B", ipk:3.54));

    System.out.println(x:"\nDaftar semua mahasiswa (in order traversal): ");
    bst.traverseInOrder(bst.root);

    System.out.println(x:"\nPencarian data mahasiswa: ");
    System.out.print(s:"Cari mahasiswa dengan ipk: 3.54 :");
    String hasilCari = bst.find(ipk:3.54) ? " Ditemukan" : "Tidak ditemukan";
    System.out.println(hasilCari);

    System.out.print(s:"Cari mahasiswa dengan ipk: 3.22 : ");
    hasilCari = bst.find(ipk:3.23) ? " Ditemukan" : "Tidak ditemukan";
    System.out.println(hasilCari);

    bst.add(new Mahasiswa04(nim:"244160131", nama:"Devi", kelas:"A", ipk:3.72));
    bst.add(new Mahasiswa04(nim:"244160205", nama:"Ehsan", kelas:"D", ipk:3.37));
    bst.add(new Mahasiswa04(nim:"244160170", nama:"Fizi", kelas:"B", ipk:3.46));
    System.out.println(x:"\nDaftar semua mahasiswa setelah penambahan 3 mahasiswa: ");
    System.out.println(x:"InOrder Traversal:");
    bst.traverseInOrder(bst.root);
    System.out.println(x:"\nPreOrder Traversal:");
    bst.traversePreOrder(bst.root);
    System.out.println(x:"\nPostOrder Traversal:");
    bst.traversePostOrder(bst.root);

    System.out.println(x:"\nPenghapusan data mahasiswa dengan ipk 3.57");
    bst.delete(ipk:3.57);
    System.out.println(x:"\nDaftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order): ");
    bst.traverseInOrder(bst.root);
```

12. Compile dan jalankan class BinaryTreeMain04 untuk mendapatkan simulasi jalannya program binary tree yang telah dibuat.

Verifikasi Hasil Percobaan

```
Daftar semua mahasiswa (in order traversal):
NIM : 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM : 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM : 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM : 244160221 Nama: 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:
InOrder Traversal:
NIM : 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM : 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM : 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM : 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM : 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM : 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM : 244160221 Nama: Badar Kelas: B IPK: 3.85

PreOrder Traversal:
NIM : 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM : 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM : 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM : 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM : 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM : 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM : 244160131 Nama: Devi Kelas: A IPK: 3.72

PostOrder Traversal:
NIM : 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM : 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM : 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM : 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM : 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM : 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM : 244160121 Nama: Ali Kelas: A IPK: 3.57

Penghapusan data mahasiswa dengan ipk 3.57
Jika 2 anak, current =
NIM : 244160131 Nama: Devi Kelas: A IPK: 3.72

Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order):
NIM : 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM : 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM : 244160170 Nama: Fizi Kelas: B IPK: 3.46
NIM : 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM : 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM : 244160221 Nama: Badar Kelas: B IPK: 3.85
```

Pertanyaan

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?
   Jawab:
   Karena binary search tree memiliki sifat terurut dimana nilai di subtree kiri selalu lebih kecil dari root dan nilai di subtree kanan selalu lebih besar. Ini memungkinkan pencarian dengan kompleksitas O(log n) dengan membagi area pencarian menjadi dua setiap langkah

2. Untuk apakah di class Node, kegunaan dari atribut left dan right?
   Jawab:
   Atribut left dan right digunakan untuk menyimpan referensi ke node anak kiri dan kanan dari node saat ini, sehingga membentuk struktur pohon biner

3. a. Untuk apakah kegunaan dari atribut root di dalam class BinaryTree?
   b. Ketika objek tree pertama kali dibuat, apakah nilai dari root?
   Jawab:
   a. Root adalah node paling atas/paling pertama dalam pohon biner yang menjadi titik awal semua operasi pada pohon
   b. Nilai root awalnya adalah null (karena diinisialisasi sebagai null dalam konstruktor)

4. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?
   Jawab:
   Node baru akan langsung menjadi root tree karena tree kosong (root == null)

5. Perhatikan method add(), di dalamnya terdapat baris program seperti di bawah ini. Jelaskan secara detil untuk apa baris program tersebut?
   Jawab:
   Kode ini mencari posisi yang tepat untuk node baru berdasarkan nilai IPK
   - Menyimpan node saat ini sebagai parent
   - Jika IPK baru lebih kecil, bergerak ke kiri
   - Jika IPK baru lebih besar, bergerak ke kanan
   - Jika ditemukan posisi kosong (null), node baru dimasukkan sebagai anak kiri/kanan dari parent
6. Jelaskan langkah-langkah pada method delete() saat menghapus sebuah node yang memiliki dua anak. Bagaimana method getSuccessor() membantu dalam proses ini?
   Jawab:
   - Menyimpan node saat ini sebagai parent
   - Jika IPK baru lebih kecil, bergerak ke kiri
   - Jika IPK baru lebih besar, bergerak ke kanan
   - Jika ditemukan posisi kosong (null), node baru dimasukkan sebagai anak kiri/kanan dari parent

Percobaan 2

Langkah-langkah:

1. Di dalam percobaan implementasi binary tree dengan array ini, data tree disimpan dalam array dan langsung dimasukan dari method main(), dan selanjutnya akan disimulasikan proses traversal secara inOrder.
2. Buatlah class BinaryTreeArray04 dan BinaryTreeArrayMain04
3. Buat atribut data dan idxLast di dalam class BinaryTreeArray04. Buat juga method populateData() dan traverseInOrder().

```
int idxLast;
Mahasiswa04[] dataMahasiswa;

BinaryTreeArray04() {
    this.dataMahasiswa = new Mahasiswa04[10];
}

void populateData(Mahasiswa04 dataMhs[], int idxLast) {
    this.dataMahasiswa = dataMhs;
    this.idxLast = idxLast;
}

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

4. Kemudian dalam class BinaryTreeArrayMain04 buat method main() dan tambahkan kode seperti gambar berikut ini di dalam method main().

```
BinaryTreeArray04 bta = new BinaryTreeArray04();

Mahasiswa04 mhs1 = new Mahasiswa04(nim:"244160121", nama:"Ali", kelas:"A", ipk:3.57);
Mahasiswa04 mhs2 = new Mahasiswa04(nim:"244160221", nama:"Badar", kelas:"B", ipk:3.41);
Mahasiswa04 mhs3 = new Mahasiswa04(nim:"244160185", nama:"Candra", kelas:"C", ipk:3.75);
Mahasiswa04 mhs4 = new Mahasiswa04(nim:"244160220", nama:"Dewi", kelas:"B", ipk:3.35);

Mahasiswa04 mhs5 = new Mahasiswa04(nim:"244160131", nama:"Devi", kelas:"A", ipk:3.48);
Mahasiswa04 mhs6 = new Mahasiswa04(nim:"244160205", nama:"Ehsan", kelas:"D", ipk:3.61);
Mahasiswa04 mhs7 = new Mahasiswa04(nim:"244160170", nama:"Fizi", kelas:"B", ipk:3.86);
Mahasiswa04 mhs8 = new Mahasiswa04(nim:"244160171", nama:"Kak Rose", kelas:"C", ipk:3.66);

Mahasiswa04[] dataMahasiswas = {mhs1, mhs2, mhs3, mhs4, mhs5, mhs6, mhs7, null, null, null};
int idxLast = 6;
bta.populateData(dataMahasiswas, idxLast);
System.out.println(x:"\nInOrder Traversal Mahasiswa: ");
bta.traversInOrder(idxStart:0);
```

5. Jalankan class BinaryTreeArrayMain04

Verifikasi Hasil Percobaan

```
InOrder Traversal Mahasiswa:
NIM : 244160220 Nama: Dewi Kelas: B IPK: 3.35
NIM : 244160221 Nama: Badar Kelas: B IPK: 3.41
NIM : 244160131 Nama: Devi Kelas: A IPK: 3.48
NIM : 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM : 244160205 Nama: Ehsan Kelas: D IPK: 3.61
NIM : 244160185 Nama: Candra Kelas: C IPK: 3.75
NIM : 244160170 Nama: Fizi Kelas: B IPK: 3.86
```

Pertanyaan

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class BinaryTreeArray?
   Jawab:
   data: Array untuk menyimpan elemen-elemen tree
   idxLast: Indeks terakhir yang berisi data valid dalam array

2. Apakah kegunaan dari method populateData()?
   Jawab:
   Untuk mengisi data tree dari array yang sudah ada dan menetapkan indeks terakhir yang valid

3. Apakah kegunaan dari method traverseInOrder()?
   Jawab:
   Untuk melakukan traversal in-order (kiri-root-kanan) pada tree yang disimpan dalam array

4. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masing-masing?
   Jawab:
   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?
   Jawab:
   Menunjukkan bahwa data valid berada sampai indeks 6 dalam array (7 elemen pertama)

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?
   Jawab:
   Ini adalah rumus untuk menghitung posisi anak dalam array
   Left child = 2*parent_index + 1
   Right child = 2*parent_index + 2

Tugas Praktikum

1. Buat method di dalam class BinaryTree04 yang akan menambahkan node dengan cara rekursif (addRekursif()).

```java
public void addRekursif(Mahasiswa04 mahasiswa) {
    if (isEmpty()) {
        root = new Node04(mahasiswa);
    } else {
        Node04 current = root;
        while (true) {
            if (mahasiswa.ipk < current.mahasiswa.ipk) {
                if (current.left == null) {
                    current.left = new Node04(mahasiswa);
                    break;
                } else {
                    current = current.left;
                }
            } else if (mahasiswa.ipk > current.mahasiswa.ipk) {
                if (current.right == null) {
                    current.right = new Node04(mahasiswa);
                    break;
                } else {
                    current = current.right;
                }
            } else {
                break; // IPK sudah ada
            }
        }
    }
}
```

2. Buat method di dalam class BinaryTree04 untuk menampilkan data mahasiswa dengan IPK paling kecil dan IPK yang paling besar (cariMinIPK() dan cariMaxIPK()) yang ada di dalam binary search tree.

```java
public void cariMinIPK() {
    if (isEmpty()) {
        System.out.println(x:"Tree kosong");
    }

    Node04 current = root;
    while (current.left != null) {
        current = current.left;
    }
    System.out.println(x:"\nMahasiswa dengan IPK terendah: ");
    current.mahasiswa.tampilInformasi();
}

public void cariMaxIPK() {
    if (isEmpty()) {
        System.out.println(x:"Tree kosong");
    }

    Node04 current = root;
    while (current.right != null) {
        current = current.right;
    }
    System.out.println(x:"\nMahasiswa dengan IPK tertinggi: ");
    current.mahasiswa.tampilInformasi();
}
```

3. Buat method dalam class BinaryTree04 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.

```java
public void tampilMahasiswaIPKdiAtas(double ipkBatas) {
    if (isEmpty()) {
        System.out.println(x:"Tree kosong");
        return;
    }

    System.out.println("\nMahasiswa dengan IPK di atas " + ipkBatas + ":");
    Node04 current = root;
    Node04 prev = null;

    while (current != null) {
        if (current.left == null) {
            // Proses node saat ini
            if (current.mahasiswa.ipk > ipkBatas) {
                current.mahasiswa.tampilInformasi();
            }
            current = current.right;
        } else {
            // Temukan predecessor
            prev = current.left;
            while (prev.right != null && prev.right != current) {
                prev = prev.right;
            }

            if (prev.right == null) {
                prev.right = current;
                current = current.left;
            } else {
                prev.right = null;
                // Proses node saat ini
                if (current.mahasiswa.ipk > ipkBatas) {
                    current.mahasiswa.tampilInformasi();
                }
                current = current.right;
            }
        }
    }
}
```

4. Modifikasi class BinaryTreeArray04 di atas, dan tambahkan :
   - method add(Mahasiswa data) untuk memasukan data ke dalam binary tree

```java
void add(Mahasiswa04 data) {
    if (idxLast < dataMahasiswa.length - 1) {
        idxLast++;
        dataMahasiswa[idxLast] = data;
    } else {
        System.out.println("Kapasitas (array) sudah penuh, tidak bisa menambahkan data");
    }
}
```

   - method traversePreOrder()

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

```
PreOrder Traversal Mahasiswa:
NIM : 244160121 Nama: Ali Kelas: A IPK: 3.57
NIM : 244160221 Nama: Badar Kelas: B IPK: 3.41
NIM : 244160220 Nama: Dewi Kelas: B IPK: 3.35
NIM : 244160171 Nama: Kak Rose Kelas: C IPK: 3.66
NIM : 244160131 Nama: Devi Kelas: A IPK: 3.48
NIM : 244160185 Nama: Candra Kelas: C IPK: 3.75
NIM : 244160205 Nama: Ehsan Kelas: D IPK: 3.61
NIM : 244160170 Nama: Fizi Kelas: B IPK: 3.86
PS C:\Users\AHMAD SOFYAN BADAWI\OneDrive\Documents\Prak
```