# LAPORAN TUGAS ALGORITMA DAN STRUKTUR DATA
# JOBSHEET 14


`

NAMA : RADITYA RIEFKI
KELAS : TI 1E
ABSEN : 23

**Implementasi Binary Search Tree menggunakan Linked List (100 Menit)**

**14.2.1 Percobaan 1**

Kode Program

Mahasiswa

```
package jobsheet14;

public class Mahasiswa23 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    public Mahasiswa23(){
    }
    public Mahasiswa23(String nim, String nama, String kelas, double ipk){
        this.nim = nim;
        this.nama = nama;
        this.kelas = kelas;
        this.ipk = ipk;
    }
    public void tampilInformasi(){
        System.out.println("NIM: "+ this.nim + " " +
        "Nama: "+ this.nama + " " +
        "Kelas: "+ this.kelas + " " +
        "IPK: "+ this.ipk);
    }
}
```

Node

```
package jobsheet14;

public class Node23 {
    Mahasiswa23 mahasiswa;
    Node23 left, right;

    public Node23(){
    }

    public Node23 (Mahasiswa23 mahasiswa){
        this.mahasiswa = mahasiswa;
        left = right = null;
    }
}
```

BinaryTree

```
package jobsheet14;

import org.w3c.dom.Node;

public class BinaryTree23 {
    Node23 root;

    public BinaryTree23(){
        root = null;
    }

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

    public void add(Mahasiswa23 mahasiswa){
        Node23 newNode = new Node23(mahasiswa);
        if (isEmpty()) {
            root = newNode;
        }else{
            Node23 current = root;
            Node23 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;
                    }
                }
            }
        }
    }
    boolean find (double ipk){
        boolean result = false;
        Node23 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;
    }

    void traversePreOrder(Node23 node){
        if (node != null) {
            node.mahasiswa.tampilInformasi();
            traversePreOrder(node.left);
            traversePreOrder(node.right);
        }
    }

    void traversInOrder (Node23 node){
        if (node != null) {
            traversInOrder(node.left);
            node.mahasiswa.tampilInformasi();
            traversInOrder(node.right);
        }
    }

    void traversePostOrder(Node23 node){
        if (node != null) {
            traversePostOrder(node.left);
            traversePostOrder(node.right);
            node.mahasiswa.tampilInformasi();
        }
    }

    Node23 getSuccessor(Node23 del){
        Node23 successor = del.right;
        Node23 succesorParent = del;
        while (successor.left != null) {
            succesorParent = successor;
            successor = successor.left;
        }
        if (successor != del.right) {
            succesorParent.left = successor.right;
            successor.right = del.right;
        }
        return successor;
    }

    void delete(double ipk){
        if (isEmpty()) {
            System.out.println("Binary tree kosong!");
            return;
        }
        //cari node (current) yang akan di hapus
        Node23 parent = root;
        Node23 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;
            }
        }
        //penghapusan
            if (current == null){
                System.out.println("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
                    Node23 successor = getSuccessor(current);
                    System.out.println("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;
                }
            }

    }
}
```

Main

```
package jobsheet14;

import org.w3c.dom.Node;

public class BinaryTree23 {
    Node23 root;

    public BinaryTree23(){
        root = null;
    }

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

    public void add(Mahasiswa23 mahasiswa){
        Node23 newNode = new Node23(mahasiswa);
        if (isEmpty()) {
            root = newNode;
        }else{
            Node23 current = root;
            Node23 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;
                    }
                }
```

```
            }
        }
    }
    boolean find (double ipk){
        boolean result = false;
        Node23 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;
    }

    void traversePreOrder(Node23 node){
        if (node != null) {
            node.mahasiswa.tampilInformasi();
            traversePreOrder(node.left);
            traversePreOrder(node.right);
        }
    }

    void traversInOrder (Node23 node){
        if (node != null) {
            traversInOrder(node.left);
            node.mahasiswa.tampilInformasi();
            traversInOrder(node.right);
        }
    }

    void traversePostOrder(Node23 node){
        if (node != null) {
            traversePostOrder(node.left);
            traversePostOrder(node.right);
            node.mahasiswa.tampilInformasi();
        }
    }

    Node23 getSuccessor(Node23 del){
        Node23 successor = del.right;
        Node23 succesorParent = del;
        while (successor.left != null) {
            succesorParent = successor;
            successor = successor.left;
        }
        if (successor != del.right) {
            succesorParent.left = successor.right;
            successor.right = del.right;
        }
```

```
        return successor;
    }

    void delete(double ipk){
        if (isEmpty()) {
            System.out.println("Binary tree kosong!");
            return;
        }
        //cari node (current) yang akan di hapus
        Node23 parent = root;
        Node23 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;
            }
        }
        //penghapusan
            if (current == null){
                System.out.println("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
                    Node23 successor = getSuccessor(current);
                    System.out.println("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;
                }
            }

    }
}
```

Output

```
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: Ali IPK: 3.57
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85

Pencarian data mahasiswa:
Cari mahasiswa dengan ipk: 3.54 :
Ditemukan
Cari Mahasiwa 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: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160121 Nama: Ali Kelas: Ali IPK: 3.57
NIM: 244160170 Nama: Fizi Kelas: E IPK: 3.65
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: Ali 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: 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160170 Nama: Fizi Kelas: E IPK: 3.65

PostOrder Traversal:
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: 244160170 Nama: Fizi Kelas: E IPK: 3.65
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
NIM: 244160121 Nama: Ali Kelas: Ali IPK: 3.57

Penghapusan data mahasiswa
Jika 2 anak, current =
NIM: 244160170 Nama: Fizi Kelas: E IPK: 3.65

Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order traversal):
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205 Nama: Ehsan Kelas: D IPK: 3.37
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.54
NIM: 244160170 Nama: Fizi Kelas: E IPK: 3.65
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72
NIM: 244160221 Nama: Badar Kelas: B IPK: 3.85
PS D:\CollegeFile\SMT 2\ALSD>
```

**14.2.2 Pertanyaan Percobaan**

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

Karena penempatan elemen yang urut dan dapat dilakukan dengan metode divide and conquer

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

Berguna sebagai pointer sama halnya dengan node di linked list

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

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

Root berguna sebagai awal dari semua operasi (sama dengan head)

Nilai root pertama kali dibuat adalah null

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

Proses yang pertama adalah pengecekan root apakah == null  jika iya setr root ke node terbary

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

```
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;
    }
}
```

Membandingan data ipk mahasiswa yang sudah ada di tree dengan node yang baru

Lalu mengecek apakah harus ke kiri atau ke kanan

Jika kosong menempatkan node baru di anak kiri atau anak kanan parent

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

Method getSuccesor() membantu untu menemukan pengganti yang tepat dan memindahkan data successor ke node yang di hapus juga memastikan tree tetap terurut

### 14.3 Kegiatan Praktikum 2

### Implementasi Binary Tree dengan Array (45 Menit)

#### 14.3.1 Tahapan Percobaan

Kode BinaryTreeArray

```
package jobsheet14;

import jobsheet10.P2Jobsheet10.Mahasiswa;

public class BinaryTreeArray23 {
    Mahasiswa23[] dataMahasiswa;
    int idxLast;

    public BinaryTreeArray23(){
    }
    void populateData (Mahasiswa23[] dataMahasiswa, int idxLast){
        this.dataMahasiswa = dataMahasiswa;
        this.idxLast = idxLast;
    }

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

            }

        }
    }

}
```

Kode Main

```
package jobsheet14;

public class BinaryTreeArrayMain23 {
    public static void main(String[] args) {
        BinaryTreeArray23 bta = new BinaryTreeArray23();

        Mahasiswa23 mhs1 = new Mahasiswa23("244160121", "Ali", "Ali", 3.57);
        Mahasiswa23 mhs2 = new Mahasiswa23("244160221", "Badar", "B", 3.41);
        Mahasiswa23 mhs3 = new Mahasiswa23("244160185", "Candra", "C",
3.75);
        Mahasiswa23 mhs4 = new Mahasiswa23("244160220", "Dewi", "B", 3.35);

        Mahasiswa23 mhs5 = new Mahasiswa23("244160131", "Devi", "A", 3.48);
        Mahasiswa23 mhs6 = new Mahasiswa23("244160205", "Ehsan", "D", 3.61);
        Mahasiswa23 mhs7 = new Mahasiswa23("244160170", "Fizi", "B", 3.86);

        Mahasiswa23[] datMahasiswa =
{mhs1,mhs2,mhs3,mhs4,mhs5,mhs6,mhs7,null,null,null};
        int idxLast = 6;
        bta.populateData(datMahasiswa, idxLast);
        System.out.println("\nInOrder Tranversal Mahasiswa: ");
        bta.traverseInOrder(0);
    }
}
```

**14.3.2 Pertanyaan Percobaan**

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

Data untuk menyimpan elemen tree

Idx last menjadi penunjuk index terakhir yang terisi

2. Apakah kegunaan dari method populateData()?

Mengisi array data dengan data awal

3. Apakah kegunaan dari method traverseInOrder()?

Melakukan traversal dari kiri – root – kanan secara rekursif dan menampilkan dara dalam urutan inroder

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 di indeks ke 5

Right child di indeks ke 6

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

Bermaksud jumlah node yang sudah di masukkan ke array artinya data array sampai indeks ke 6 dan mencegah traversal out of index

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?

Untuk traversal dan pencarian posisi anak secara konsisten tanpa pointer dan terurut

**14.4 Tugas Praktikum**

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

```
public void addRekursif(Mahasiswa23 data) {
    root = addRekursif(root, data);
}

private Node23 addRekursif(Node23 current, Mahasiswa23 data) {
    if (current == null) {
        return new Node23(data);
    }

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

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 Mahasiswa23 cariMinIPK() {
    if (root == null) {
        return null;
    }
    Node23 current = root;
    while (current.left != null) {
        current = current.left;
    }
    return current.mahasiswa;
}

public Mahasiswa23 cariMaxIPK() {
    if (root == null) {
        return null;
    }
    Node23 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 tampilMahasiswaIPKdiAtas(double ipkBatas) {
    tampilMahasiswaIPKdiAtas(root, ipkBatas);
}

private void tampilMahasiswaIPKdiAtas(Node23 current, double ipkBatas) {
    if (current != null) {
        tampilMahasiswaIPKdiAtas(current.left, ipkBatas);
        if (current.mahasiswa.ipk > ipkBatas) {
            System.out.println("NIM: " + current.mahasiswa.nim +
                               ", Nama: " + current.mahasiswa.nama +
                               ", IPK: " + current.mahasiswa.ipk);
        }
        tampilMahasiswaIPKdiAtas(current.right, ipkBatas);
    }
}
```

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

```
public void add(Mahasiswa23 data) {
    if (idxLast + 1 < dataMahasiswa.length) {
        idxLast++;
        dataMahasiswa[idxLast] = data;
    } else {
        System.out.println(x:"Tree array penuh!");
    }
}

public void traversePreOrder() {
    traversePreOrder(idxStart:0);
}

private void traversePreOrder(int idxStart) {
    if (idxStart <= idxLast && dataMahasiswa[idxStart] != null) {
        System.out.println("NIM: " + dataMahasiswa[idxStart].nim +
                           ", Nama: " + dataMahasiswa[idxStart].nama +
                           ", IPK: " + dataMahasiswa[idxStart].ipk);
        traversePreOrder(2 * idxStart + 1);
        traversePreOrder(2 * idxStart + 2);
    }
}
```