JOBSHEET TREE

ALGORITMA DAN STRUKTUR DATA

Burhnauddin ihsan

244107020189

TI 1E/06

PROGRAM STUDI D-IV TEKNIK INFORMATIKA JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

PERCOBAAN 1

1. Kode program

- Mahasiswa25

```
public class Mahasiswa06 {

        String nim, nama, kelas;
        double ipk;

        Mahasiswa06() {
        }

        Mahasiswa06(String nim, String nama, String kelas, double
ipk) {
            this.nim = nim;
            this.nama = nama;
            this.kelas = kelas;
            this.ipk = ipk;
        }

        void tampilInformasi() {
            System.out.println("Nama: " + this.nama + " | NIM: " +
this.nim + " | Kelas: " + this.kelas + " | IPK: " + this.ipk);
        }

}
```

- Node

```
public class Node {
    Mahasiswa06 mhs;
    Node left;
    Node right;

    Node () {

    }

    Node (Mahasiswa06 mhs) {
        this.mhs = mhs;
        left = right = null;
    }
}
```

- BinaryTree06

```
public class BinaryTree06 {
    Node06 root;

    public BinaryTree06() {
        root = null;
    }

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

    public void addRecursive(Mahasiswa06 mhs) {
        Node06 newNode = new Node06(mhs);
        root = addHelper(root, newNode);
```

```
public Node06 addHelper(Node06 root, Node06 node) {
        if (root == null) {
            root = node;
            return root;
        }

        if (node.mhs.ipk < root.mhs.ipk) {
            root.left = addHelper(root.left, node);
        } else {
            root.right = addHelper(root.right, node);
        }

        return root;
    }

    public void add (Mahasiswa06 mhs) {
        Node06 newNode = new Node06(mhs);
        if (isEmpty()) {
            root = newNode;
        } else {
            Node06 current = root;
            Node06 parent = null;
            while (true) {
                parent = current;
                if (mhs.ipk < current.mhs.ipk) {
                    current = current.left;
                    if (current == null) {
                        parent.left = newNode;
                        return;
                    }
                } else if (mhs.ipk > current.mhs.ipk) {
                    current = current.right;
                    if (current == null) {
                        parent.right = newNode;
                        return;
                    }
                }
            }
        }
    }

    Mahasiswa06 cariMinIpk() {
        Node06 current = root;

        while (current.left != null) {
            current = current.left;
        }

        return current.mhs;
    }

    Mahasiswa06 cariMaxIpk() {
        Node06 current = root;

        while (current.right != null) {
            current = current.right;
        }

        return current.mhs;
    }

    boolean find(double ipk) {
        boolean result = false;
        Node06 current = root;
```

```
while (current != null) {
            if (current.mhs.ipk == ipk) {
                return true;
            } else if (ipk > current.mhs.ipk) {
                current = current.right;
            } else {
                current = current.left;
            }
        }
        return result;
    }

    void traversePreOrder(Node06 node) {
        if (node == null) {
            return;
        }

        node.mhs.tampilInformasi();
        traversePreOrder(node.left);
        traversePreOrder(node.right);
    }

    void traverseInOrder(Node06 node) {
        if (node == null) {
            return;
        }

        traverseInOrder(node.left);
        node.mhs.tampilInformasi();
        traverseInOrder(node.right);
    }

    void traversePostOrder(Node06 node) {
        if (node == null) {
            return;
        }

        traversePostOrder(node.left);
        traversePostOrder(node.right);
        node.mhs.tampilInformasi();
    }

    void traverseIpk(Node06 node, double ipkBatas) {
        if (node == null) {
            return;
        }

        if (node.mhs.ipk >= ipkBatas) {
        traverseIpk(node.left, ipkBatas);
        node.mhs.tampilInformasi();
        }

        traverseIpk(node.right, ipkBatas);
    }

    void tampilMahasiswaIPKdiAtas(double ipkBatas) {
        System.out.println("Mahasiswa dengan ipk di atas " + ipkBatas
+ " : ");
        traverseIpk(root, ipkBatas);
    }

    Node06 getSuccessor(Node06 node) {
        Node06 successor = node.right;
        Node06 successorParent = node;
```

```
while (successor.left != null) {
            successorParent = successor;
            successor = successor.left;
        }

        if (successor != node.right) {
            successorParent.left = successor.right;
            successor.right = node.right;

        }
        return successor;
    }

    void delete(double ipk) {
        if (isEmpty()) {
            System.out.println("Binary tree kosong");
            return;
        }

        Node06 parent = root;
        Node06 current = root;
        boolean isLeftChild = false;

        while (current != null) {
            if (current.mhs.ipk == ipk) {
                break;
            } else if (ipk < current.mhs.ipk) {
                parent = current;
                current = current.left;
                isLeftChild = true;
            } else {
                parent = current;
                current = current.right;
            }
        }

        if (current == null) {
            System.out.println("data tidak ditemukan");
            return;
        } else {
            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) {
            if (current == root) {
                root = current.right;
            } else {
                if (isLeftChild) {
                    parent.left = current.right;
                } else {
                    parent.right = current.left;
                }
            }
        } else if(current.right == null) {
            if (current == root) {
                root = current.left;
            } else {
```

```
    parent.right = current.left;
                }
            }
        } else {
            Node06 successor = getSuccessor(current);
            System.out.println("Jika 2 anak, current = ");
            successor.mhs.tampilInformasi();
            if (current == root) {
                root = successor;
            } else {
                if(isLeftChild) {
                    parent.left = successor;
                } else {
                    parent.right = successor;
                }
            }
            successor.left = current.left;
        }
        }
    }
}
```

- BinaryTreeMain06

```
public class BinaryTreeMain06 {
    public static void main(String[] args) {
          BinaryTree06 bst = new BinaryTree06();

        bst.addRecursive(new Mahasiswa06("244160121", "Ali", "A",
3.57));
        bst.addRecursive(new Mahasiswa06("244160221", "Badar", "B",
3.85));
        bst.addRecursive(new Mahasiswa06("24416018S", "Candra", "C",
3.21));
        bst.addRecursive(new Mahasiswa06("244160220", "Dewi", "B",
3.54));

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

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

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

        bst.addRecursive(new Mahasiswa06("244160131", "Devi", "A",
3.72));
        bst.addRecursive(new Mahasiswa06("24416020S", "Ehsan", "D",
3.37));
        bst.addRecursive(new Mahasiswa06("244160170", "Fizi", "B",
3.46));

        System.out.println("\nDaftar semua mahasiswa setelah
penambahan 3 mahasiswa:");
        System.out.println("InOrder Traversal:");
        bst.traverseInOrder(bst.root);
```

```
 System.out.println("\nPostOrder Traversal:");
        bst.traversePostOrder(bst.root);

        System.out.println("\nPenghapusan data mahasiswa");
        bst.delete(3.57);
        System.out.println("Daftar semua mahasiswa setelah
penghapusan 1 mahasiswa (in order traversal):");
        bst.traverseInOrder(bst.root);
        System.out.println();

        Mahasiswa06 ipkTerbesar = bst.cariMaxIpk();
        System.out.println("mahasiswa dengan ipk terbesar : " +
ipkTerbesar.nama + " dengan ipk " + ipkTerbesar.ipk);
        Mahasiswa06 ipkTerkecil = bst.cariMinIpk();
        System.out.println("mahasiswa dengan ipk terkecil : " +
ipkTerkecil.nama + " dengan ipk " + ipkTerkecil.ipk);
         System.out.println();
        bst.tampilMahasiswaIPKdiAtas(3.4);

        /* Tree


           3.57
          /    \
       3.21     3.85
          \     /
          3.54 3.72
         /
      3.37
         \
         3.46


          */

    }
}
```

2. Hasil dari kode program

```
Daftar semua mahasiswa (in order traversal):
Nama: Candra | NIM: 24416018S | Kelas: C | IPK: 3.21
Nama: Dewi | NIM: 24416022O | Kelas: B | IPK: 3.54
Nama: Ali | NIM: 244160121 | Kelas: A | IPK: 3.57
Nama: Badar | NIM: 24416022l | 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:
Nama: Candra | NIM: 24416018S | Kelas: C | IPK: 3.21
Nama: Ehsan | NIM: 24416020S | Kelas: D | IPK: 3.37
Nama: Fizi | NIM: 24416017O | Kelas: B | IPK: 3.46
Nama: Dewi | NIM: 24416022O | Kelas: B | IPK: 3.54
Nama: Ali | NIM: 244160121 | Kelas: A | IPK: 3.57
Nama: Devi | NIM: 244160131 | Kelas: A | IPK: 3.72
Nama: Badar | NIM: 24416022l | Kelas: B | IPK: 3.85

PreOrder Traversal:
Nama: Ali | NIM: 244160121 | Kelas: A | IPK: 3.57
Nama: Candra | NIM: 24416018S | Kelas: C | IPK: 3.21
Nama: Dewi | NIM: 24416022O | Kelas: B | IPK: 3.54
Nama: Ehsan | NIM: 24416020S | Kelas: D | IPK: 3.37
Nama: Fizi | NIM: 24416017O | Kelas: B | IPK: 3.46
Nama: Badar | NIM: 24416022l | Kelas: B | IPK: 3.85
Nama: Devi | NIM: 244160131 | Kelas: A | IPK: 3.72
```

3. **PERTANYAAN**

1. Karena data yang disimpan dalam binary search tree sudah diurutkan, jadi akan mempermudah proses pencarian, proses traverse tidak perlu menelusuri seluruh bagian tree sehingga dapat menghemat waktu.

2. attribute left berfungsi untuk menyimpan alamat memori node di bagian kiri node tersebut (anak kiri) sedangkan right berfungsi sebaliknya.

3. attribute root pada class BinaryTree berfungsi untuk menandai bagian teratas dari sebuah tree (parent utama)
4. value dari root akan diinisasi dengan node yang baru diinputkan
5. variabel parent digunakan untuk menyimpan nilai parent dari variabel current, nilai dari parent ini akan terus menyesuaikan berdasarkan current, terdapat 2 kondisi yang memungkinkan yaitu saat nilai ipk dari mahasiswa yang diinputkan lebih kecil maka nilai dari parent.left akan diisi dengan newNode dan sebaliknya. Hal ini dilakukan sesuai dengan aturan binary search tree
6. Method getSuccessor() membantu dalam proses ini dengan cara mencari node pengganti (successor) yang nilainya paling mendekati dan lebih besar dari node yang ingin dihapus, yaitu node paling kiri di subtree kanan. Setelah successor ditemukan, node tersebut akan menggantikan posisi node yang dihapus. Jika node yang dihapus adalah root, maka root akan diganti dengan successor. Jika bukan root, maka parent dari node yang dihapus akan menunjuk ke successor. Agar struktur binary search tree tetap valid, anak kiri dari node yang dihapus disambungkan ke kiri successor, karena successor tidak memiliki anak kiri (sebab dia adalah node paling kiri di subtree kanan). Dengan cara ini, pohon tetap terjaga urutannya setelah proses penghapusan selesai.

PERCOBAAN 2

1. Kode program
   - BinaryTreeArray06

```
public class BinaryTreeArray06 {
    Mahasiswa06[] dataMhs;
    int idxLast;

    public BinaryTreeArray06() {
        this.dataMhs = new Mahasiswa06[10];
    }

    void populateData (Mahasiswa06 dataMhs[], int idxLast) {
        this.dataMhs = dataMhs;
        this.idxLast = idxLast;
    }

    void addData (Mahasiswa06 mhs) {
        if (dataMhs[0] == null) {
            this.dataMhs[0] = mhs;
            idxLast++;
            return;
        }
```

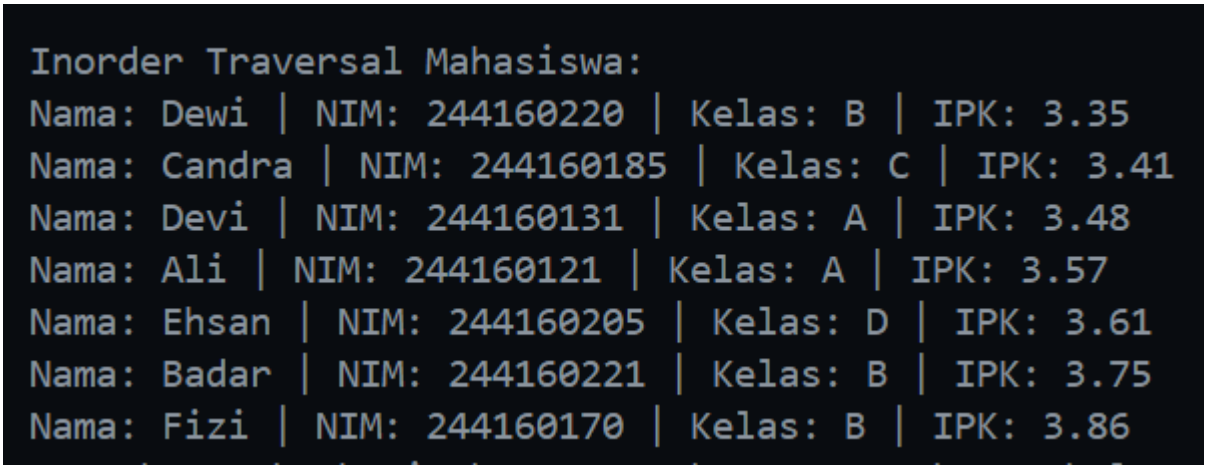
```
  int idx = 0;
        while (idx < dataMhs.length && dataMhs[idx] != null) {
            if (mhs.ipk < dataMhs[idx].ipk) {
                idx = 2 * idx + 1;
            } else {
                idx = 2 * idx + 2;
            }
        }

        if (idx >= dataMhs.length) {
            System.out.println("Array penuh, tidak bisa menambahkan
data.");
            return;
        }


        dataMhs[idx] = mhs;
        idxLast++;
    }

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

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

- BinaryTreeArrayMain06

```
public class BinaryTreeArrayMain06 {
    public static void main(String[] args) {

BinaryTreeArray06 bta = new BinaryTreeArray06();
Mahasiswa06 mhs1 = new Mahasiswa06("244160121", "Ali", "A", 3.57);
Mahasiswa06 mhs2 = new Mahasiswa06("244160185", "Candra", "C", 3.41);
Mahasiswa06 mhs3 = new Mahasiswa06("244160221", "Badar", "B", 3.75);
Mahasiswa06 mhs4 = new Mahasiswa06("244160220", "Dewi", "B", 3.35);

Mahasiswa06 mhs5 = new Mahasiswa06("244160131", "Devi", "A", 3.48);
Mahasiswa06 mhs6 = new Mahasiswa06("244160205", "Ehsan", "D", 3.61);
Mahasiswa06 mhs7 = new Mahasiswa06("244160170", "Fizi", "B", 3.86);

// Mahasiswa06[] dataMahasiswas = {mhs1, mhs2, mhs3, mhs4, mhs5, mhs6,
mhs7, null, null, null};
// int idxLast = 6;
// bta.populateData(dataMahasiswas, idxLast);

bta.addData(mhs1);
bta.addData(mhs2);
```

```
bta.addData(mhs3);
bta.addData(mhs4);
bta.addData(mhs5);
bta.addData(mhs6);
bta.addData(mhs7);

System.out.println("\nInorder Traversal Mahasiswa: ");
bta.traverseInOrder(0);

System.out.println("\nPreorder Traversal Mahasiswa: ");
bta.traversePreOrder(0);
    }
}
```

2. Hasil dari kode program



3. **PERTANYAAN**

1. idxLast digunakan untuk menyimpan index terakhir dari dataMhs, attribute ini diperlukan untuk membatasi traverse supaya berhenti sampai di idxLast
2. method populateData digunakan untuk menyimpan array dataMhs yang sudah diisi oleh beberapa objek dan menyimpan informasi idxLast
3. method traverseInOrder digunakan untuk melakukan traverse secara berurutan dengan menggunakan formula yang sudah ditentukan, baik untuk anak kiri dan anak kanan
4. node pada index 2 left child = 2 * 2 + 1 = 5 right child = 2 * 2 + 2 = 6
5. untuk membatasi traverse saat digunakan di method traverseInOrder supaya tidak melebihi batas array yang terisi
6. karena dengan formula tersebut otomatis akan didapatkan nilai dari anak kiri dan anak kanan dari node tersebut.

TUGAS

1. method addrekursif

```
public void addRecursive(Mahasiswa06 mhs) {
    Node06 newNode = new Node06(mhs);
    root = addHelper(root, newNode);
}
```

```
Daftar semua mahasiswa setelah penambahan 3 mahasiswa:
InOrder Traversal:
Nama: Candra | NIM: 24416018S | Kelas: C | IPK: 3.21
Nama: Ehsan | NIM: 24416020S | Kelas: D | IPK: 3.37
Nama: Fizi | NIM: 244160170 | Kelas: B | IPK: 3.46
Nama: Dewi | NIM: 244160220 | Kelas: B | IPK: 3.54
Nama: Ali | NIM: 244160121 | Kelas: A | IPK: 3.57
Nama: Devi | NIM: 244160131 | Kelas: A | IPK: 3.72
Nama: Badar | NIM: 244160221 | Kelas: B | IPK: 3.85
```

2. method mencari minIpk dan maxIpk

```
Mahasiswa06 cariMinIpk() {
    Node06 current = root;

    while (current.left != null) {
        current = current.left;
    }

    return current.mhs;
}

Windsurf: Refactor | Explain | Generate Javadoc | ×
Mahasiswa06 cariMaxIpk() {
    Node06 current = root;

    while (current.right != null) {
        current = current.right;
    }

    return current.mhs;
}
```

```
mahasiswa dengan ipk terbesar : Badar dengan ipk 3.85
mahasiswa dengan ipk terkecil : Candra dengan ipk 3.21
```

3. method menampilkan mahasiswa dengan ipk diatas input tertentu

```java
void tampilMahasiswaIPKdiAtas(double ipkBatas) {
    System.out.println("Mahasiswa dengan ipk di atas " + ipkBatas + " : ");
    traverseIpk(root, ipkBatas);
}
```

```
Mahasiswa dengan ipk di atas 3.4 :
Nama: Fizi | NIM: 244160170 | Kelas: B | IPK: 3.46
Nama: Dewi | NIM: 244160220 | Kelas: B | IPK: 3.54
Nama: Devi | NIM: 244160131 | Kelas: A | IPK: 3.72
Nama: Badar | NIM: 244160221 | Kelas: B | IPK: 3.85
```

4. method addData

```java
public void add (Mahasiswa06 mhs) {
    Node06 newNode = new Node06(mhs);
    if (isEmpty()) {
        root = newNode;
    } else {
        Node06 current = root;
        Node06 parent = null;    The assigned value is never used
        while (true) {
            parent = current;
            if (mhs.ipk < current.mhs.ipk) {
                current = current.left;
                if (current == null) {
                    parent.left = newNode;
                    return;
                }
            } else if (mhs.ipk > current.mhs.ipk) {
                current = current.right;
                if (current == null) {
                    parent.right = newNode;
                    return;
                }
            }
        }
    }
}
```

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

5. method preOrderTraverse untuk binary tree array

```
void traversePreOrder(Node06 node) {
    if (node == null) {
        return;
    }

    node.mhs.tampilInformasi();
    traversePreOrder(node.left);
    traversePreOrder(node.right);
}
```

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