LAPORAN HASIL PRAKTIKUM

ALGORITMA DAN STRUKTUR DATA

JOBSHEET 14

ATHAULLA HAFIZH

244107020030

TI 1 E

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2024

2. Praktikum

Percobaan 1

Kode program class Mahasiswa05

```
package Pertemuan14;


public class Mahasiswa05 {
    String nim;
    String nama;
    String kelas;
    double ipk;


    public Mahasiswa05() {
    }


    public Mahasiswa05(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);
    }
}
```

Kode program class Node05

```
package Pertemuan14;


public class Node05 {
    Mahasiswa05 mahasiswa;
```

```
    Node05 left, right;

    public Node05() {
    }

    public Node05(Mahasiswa05 mahasiswa) {
        this.mahasiswa = mahasiswa;
        left = right = null;
    }
}
```

Kode program class BinaryTree05

```
package Pertemuan14;

public class BinaryTree05 {
    Node05 root;

    public BinaryTree05() {
        root = null;
    }

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

    public void add(Mahasiswa05 mahasiswa) {
        Node05 newNode = new Node05(mahasiswa);
        if (isEmpty()) {
            root = newNode;
        } else {
            Node05 current = root;
            Node05 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;
        Node05 current = root;
        while (current != null) {
            if (current.mahasiswa.ipk == ipk) {
                result = true;
            } else if (ipk > current.mahasiswa.ipk) {
                current = current.right;
            } else {
                current = current.left;
            }
        }
        return result;
    }

    void traversePreOrder(Node05 node) {
        if (node != null) {
            node.mahasiswa.tampilInformasi();
            traversePreOrder(node.left);
```

```
            traversePreOrder(node.right);
        }
    }

    void traverseInOrder(Node05 node) {
        if (node != null) {
            traverseInOrder(node.left);
            node.mahasiswa.tampilInformasi();
            traverseInOrder(node.right);
        }
    }

    void traversePostOrder(Node05 node) {
        if (node != null) {
            traversePostOrder(node.left);
            traversePostOrder(node.right);
            node.mahasiswa.tampilInformasi();
        }
    }

    Node05 getSuccessor(Node05 del) {
        Node05 successor = del.right;
        Node05 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;
    }
```

```
    void delete(double ipk) {
        if (isEmpty()) {
            System.out.println("Binary tree kosong");
            return;
        }
        //cari node (current) yang akan dihapus
        Node05 parent = root;
        Node05 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 ada 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 ada
anak kiri
                if (current == root) {
                    root = current.left;
                } else {
                    if (isLeftChild) {
                        parent.left = current.left;
                    } else {
                        parent.right = current.left;
                    }
                }
            } else { //jika ada dua anak
                Node05 successor = getSuccessor(current);
                System.out.println("Jika 2 anak, current = ");
                successor.mahasiswa.tampilInformasi();
                if (current == root) {
                    root = successor;
                } else {
                    if (isLeftChild) {
                        parent.left = successor; //set anak kiri
dari parent
                    } else {
                        parent.right = successor; //set anak kanan
```

```
                    }
                }
                successor.left = current.left; //set anak kiri dari
successor
            }
        }
    }
}
```

Kode program class BinaryTreeMain05

```
package Pertemuan14;


public class BinaryTreeMain05 {
    public static void main(String[] args) {
        BinaryTree05 bst = new BinaryTree05();


        bst.add(new Mahasiswa05("244160121", "Ali", "A", 3.57));
        bst.add(new Mahasiswa05("244160221", "Badar", "B", 3.85));
        bst.add(new Mahasiswa05("244160185", "Candra", "C", 3.21));
        bst.add(new Mahasiswa05("244160220", "Dewi", "B", 3.54));


        System.out.println("\nDaftar 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.add(new Mahasiswa05("244160131", "Devi", "A", 3.72));
```

```
        bst.add(new Mahasiswa05("244160205", "Ehsan", "P", 3.37));
        bst.add(new Mahasiswa05("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("\nPreOrder Traversal:");
        bst.traversePreOrder(bst.root);
        System.out.println("\nPostOrder Traversal:");
        bst.traversePostOrder(bst.root);


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

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: 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: P 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: P 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: P 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
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:
Jika 2 anak, current =
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72

Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order traversal):
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205 Nama: Ehsan Kelas: P 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. Binary tree biasa: Seperti daftar acak, harus cek satu-satu (lambat)
   Binary search tree: Seperti kamus terurut, bisa langsung tahu harus ke kiri/kanan (cepat)

2. left = penunjuk ke "anak kiri" (nilai lebih kecil)
   right = penunjuk ke "anak kanan" (nilai lebih besar)

3. a. root adalah "pintu masuk" tree, semua akses dimulai dari root
   b. null (kosong) karena belum ada mahasiswa sama sekali

4. Mahasiswa pertama otomatis jadi root karena dia satu-satunya

5. Langkah-langkah sederhana:
   Cek apakah pohon kosong?
   - Kalau kosong → mahasiswa baru jadi root

   Kalau sudah ada isi, mulai dari root:

   - Bandingkan IPK mahasiswa baru dengan IPK yang ada
   - IPK lebih kecil? → Jalan ke KIRI
   - IPK lebih besar/sama? → Jalan ke KANAN

   Terus jalan sampai ketemu tempat kosong:
   - Kalau sampai ujung dan kosong → TARUH DI BAGIAN TERSEBUT

6. Langkah mencari successor:
   - Mulai dari anak kanan (yang pasti lebih besar dari yang dihapus)
   - Terus ke kiri sampai mentok (cari yang terkecil di sebelah kanan)
   - Inilah successor - dia adalah yang "paling kecil dari yang besar"

   Kenapa successor sempurna?
   - Dia lebih besar dari SEMUA yang ada di sebelah kiri
   - Dia lebih kecil dari SEMUA yang ada di sebelah kanan
   - Jadi aturan BST tetap berlaku

   Method getSuccessor() tugasnya:

   Cari pengganti yang tepat dan rapikan semua "hubungan keluarga" agar pohon tetap rapi dan aturan BST tidak rusak.

Percobaan 2

Kode program class BinaryTreeArray05

```
package Pertemuan14;


public class BinaryTreeArray05 {
    Mahasiswa05[] dataMahasiswa;
    int idxLast;


    public BinaryTreeArray05() {
        this.dataMahasiswa = new Mahasiswa05[10];
    }


    void populateData (Mahasiswa05 dataMhs[], int idxLast) {
```

```
        this.dataMahasiswa = dataMhs;
        this.idxLast = idxLast;
    }

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

Kode program class BinaryTreeArrayMain05

```
package Pertemuan14;


public class BinaryTreeArrayMain05 {
    public static void main(String[] args) {
        BinaryTreeArray05 bta = new BinaryTreeArray05();


        Mahasiswa05 mhs1 = new Mahasiswa05("244160121", "Ali", "A",
3.57);
        Mahasiswa05 mhs2 = new Mahasiswa05("244160185", "Candra",
"C", 3.41);
        Mahasiswa05 mhs3 = new Mahasiswa05("244160221", "Badar",
"B", 3.75);
        Mahasiswa05 mhs4 = new Mahasiswa05("244160220", "Dewi", "B",
3.35);
        Mahasiswa05 mhs5 = new Mahasiswa05("244160131", "Devi", "A",
3.48);
        Mahasiswa05 mhs6 = new Mahasiswa05("244160205", "Ehsan",
"D", 3.61);
        Mahasiswa05 mhs7 = new Mahasiswa05("244160170", "Fizi", "B",
3.86);
```

```
        Mahasiswa05[] dataMahasiswa = {mhs1, mhs2, mhs3, mhs4, mhs5,
mhs6, mhs7, null, null, null};

        int idxLast = 6;

        bta.populateData(dataMahasiswa, idxLast);

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

        bta.traverseInOrder(0);

    }

}
```

Output

```
Inorder Traversal Mahasiswa:
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.35
NIM: 244160185 Nama: Candra Kelas: C 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: 244160221 Nama: Badar Kelas: B IPK: 3.75
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.86
```

Pertanyaan

1. Kegunaan atribut data dan idxLast di class BinaryTreeArray:
- dataMahasiswa: Array untuk menyimpan data mahasiswa dalam bentuk binary tree
- idxLast: Menyimpan indeks terakhir yang berisi data (bukan null) dalam array

2. Kegunaan method populateData():
Method ini berfungsi untuk:
- Mengisi array dataMahasiswa dengan data mahasiswa yang sudah disiapkan
- Menyimpan informasi indeks terakhir (idxLast) untuk mengetahui batas data yang valid

3. Kegunaan method traverseInOrder():
Method ini melakukan traversal InOrder pada binary tree yang disimpan dalam array:
- Kunjungi left child terlebih dahulu
- Tampilkan data node saat ini
- Kunjungi right child
- Menghasilkan output data dalam urutan tertentu

4. Posisi left child dan right child untuk node di indeks 2:
Jika node berada di indeks 2, maka:
- Left child di indeks: 2 * 2 + 1 = 5
- Right child di indeks: 2 * 2 + 2 = 6

5. Kegunaan statement int idxLast = 6:
Statement ini menentukan bahwa data mahasiswa yang valid dalam array hanya sampai indeks 6. Artinya:
- Indeks 0-6 berisi data mahasiswa
- Indeks 7-9 berisi null (kosong)
- Digunakan sebagai batas dalam traversal agar tidak memproses data kosong

6. Kegunaan rumus 2idxStart+1 dan 2idxStart+2:
Rumus ini adalah formula standar Binary Tree Array:
- 2*idxStart+1: Mencari indeks left child dari node di indeks idxStart
- 2*idxStart+2: Mencari indeks right child dari node di indeks idxStart

Tugas

Kode program yang digunakan terdapat pada repository saya pada Praktikum14 yang terbaru

Output BinaryTreeMain05

```
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: P 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: 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: P 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: P 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:
Jika 2 anak, current =
NIM: 244160131 Nama: Devi Kelas: A IPK: 3.72

Daftar semua mahasiswa setelah penghapusan 1 mahasiswa (in order traversal):
NIM: 244160185 Nama: Candra Kelas: C IPK: 3.21
NIM: 244160205 Nama: Ehsan Kelas: P 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

Mahasiswa dengan ipk terbesar : Badar dengan ipk 3.85
Mahasiswa dengan ipk terkecil : Candra dengan ipk 3.21

Mahasiswa dengan IPK di atas 3.4:
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
```

Output BinaryTreeArrayMain05

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

Inorder Traversal Mahasiswa:
NIM: 244160220 Nama: Dewi Kelas: B IPK: 3.35
NIM: 244160185 Nama: Candra Kelas: C 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: 244160221 Nama: Badar Kelas: B IPK: 3.75
NIM: 244160170 Nama: Fizi Kelas: B IPK: 3.86

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