LAPORAN PRAKTIKUM STRUKTUR DATA

BINARY TREE DAN GRAFH

OLEH:

AHMAD ZAHRAN

2411532004

DOSEN PENGAMPU:

DR. WAHYUDI, S.T, M.T

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

A. Pendahuluan

Struktur data merupakan fondasi penting dalam dunia pemrograman, khususnya dalam pengolahan dan pengurutan data. Salah satu bentuk implementasinya adalah penggunaan algoritma binary tree dan grafh.

B. Tujuan

Tujuan dilakukannya praktikum ini adalah sebagai berikut:

1. Memahami Struktur Data BINARY TREE DAN GRAFH pada java

C. Langkah Kerja Praktikum

a. Kelas Node

1. Pertama buat class Node, dan buat attributnya, yaitu ada ‘data’ untuk menyimpan nilai node dan Node left, Node Right adalah pointer ke kiri dan kanan dalam struktur binary tree

```
public class Node {
    //Ahmad Zahran

    int data;
    Node left;
    Node right;
```

2. Lalu buat konstruktor node dengan data, dan left, right di set null

```
public Node(int data) {
    this.data = data;
    left = null;
    right = null;
}
```

3. Method Set dan Get, pertama Set untuk menetapkan anak pada tree jika tidak ada, lalu get untuk memanggil Node saat ini

```
public void setLeft(Node node) {
    if (left == null)
        left = node;
}

public void setRight(Node node) {
    if (right == null)
        right = node;
}

public Node getLeft() {
    return left;
}

public Node getRight() {
    return right;
}

public int getData() {
    return data;
}
```

4. Membuat Method untuk printPreorder, Postorder, dan Inorder

```
void printPreorder(Node node) {
    if (node == null)
        return;
    System.out.print(node.data + " ");
    printPreorder(node.left);
    printPreorder(node.right);
}

void printPostorder(Node node) {
    if (node == null)
        return;
    printPostorder(node.left);
    printPostorder(node.right);
    System.out.print(node.data + " ");
}

void printInorder(Node node) {
    if (node == null)
        return;
    printInorder(node.left);
    System.out.print(node.data + " ");
    printInorder(node.right);
}
```

5. Membuat Method untuk menampilkan visualisasi struktur tree

```
public String print() {
    return this.print("", true, "");
}

public String print(String prefix, boolean isTail, String sb) {
    if (right != null) {
        right.print(prefix + (isTail ? "|   " : "    "), false, sb);
    }
    System.out.println(prefix + (isTail ? "\\-- " : "/-- ") + data);
    if (left != null) {
        left.print(prefix + (isTail ? "    " : "|   "), true, sb);
    }
    return sb;
```

b. Kelas BTree

1. Pertama Buat kelas, dan atribut, yaitu ada Root untuk node paling atas dari tree, dan currentNode untuk melacak node yang sedang aktif, lalu inisiasi tree dengan root null

```
public class BTree {
    private Node root;
    private Node currentNode;
    public BTree() {
        root = null;
    }
```

2. Method search untuk memulai searching dari root, dan search data di seluruh pohon dan jika ditemukan kita return true

```
  public boolean search (int data) {
      return search(root, data);
  }
  public boolean search (Node node, int data) {
      if (node.getData() == data)
          return true;
      if (node.getLeft() != null)
          if (search (node.getLeft(), data))
              return true;
      if (node.getRight() != null)
          if (search (node.getRight(), data))
              return true;
      return false;
```

3. Method print Inorder, Preorder, dan Postorder

```
 public void printInorder() {
     root.printInorder(root);
 }
 public void printPreorder() {
     root.printPreorder(root);
 }
 public void printPostorder() {
     root.printPostorder(root);
 }
```

4. Untuk mengecek apakah pohon kosong?

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

5. Fungsi rekursif yang menghitung jumlah node dalam pohon

```
public int countNodes() {
    return countNodes(root);
}

private int countNodes(Node node) {
    int count = 1;
    if (node == null) {
        return 0;
    }else {
        count += countNodes(node.getLeft());
        count += countNodes(node.getRight());
        return count;
    }
```

6. Mengatur dan mengakses current Node

```
public Node getCurrent() {
    return currentNode;
}
public Node setCurrent( Node node) {
    return currentNode = node;
```

7. Dan terakhir Meletakkan root baru

```
public void setRoot(Node root) {
    this.root = root;
}
```

c. Kelas TreeMain

1. Pertama buat pohon dari class BTree

```
BTree tree = new BTree ();
System.out.println("Jumlah simpul awal pohon: ");
System.out.println(tree.countNodes());
```

2. Lalu kita buat root dan tampilkan

```
tree.setRoot(root);
System.out.println("Jumlah simpul jika a=hanya ada root");
System.out.println(tree.countNodes());
```

3. Lalu tambahkan Simpul lainnya

```
Node node2 = new Node (2);
Node node3 = new Node (3);
Node node4 = new Node (4);
Node node5 = new Node (5);
Node node6 = new Node (6);
Node node7 = new Node (7);
```

4. Dan atur Struktur Pohon

```
root.setLeft(node2);
node2.setLeft(node4);
node2.setRight(node5);
node3.setLeft(node6);
root.setRight(node3);
node3.setRight(node7);
```

5. Membuat dan print Node aktif (currentNode)

```
//set root
tree.setCurrent(tree.getRoot());
System.out.println("menampilkan simpul terakhir: ");
System.out.println(tree.getCurrent().getData());
```

6. Print jumlah node setelah semua node dimasukkan

```
System.out.println(tree.getCurrent().getData());
System.out.println("Jumlah simpul; setelah simpul 7 ditambahkan");
```

7. Tampilkan hasil dari traversal, Inorder, Postorder, dan Preorder

```
System.out.println(tree.countNodes());
System.out.println("Inorder: ");
tree.printInorder();
System.out.println("\n Preorder: ");
tree.printPreorder();
System.out.println("\n Postorder: ");
tree.printPostorder();
```

8. Dan terakhir print pohon secara visual

```
System.out.println("\n menampilkan simpul dalam bentuk pohon: ");
tree.print();
```

d. Kelas GraphTranversal

1. Pertama Deklarasi dan inisialisasi Graf

```
public class GraphTraversal {
    private Map<String, List<String>> graph = new HashMap<>();
```

2. Menambahkan Edge

```
public void addEdge(String node1, String node2) {
    graph.putIfAbsent(node1, new ArrayList<>());
    graph.putIfAbsent(node2, new ArrayList<>());
    graph.get(node1).add(node2);
    graph.get(node2).add(node1);
```

3. Menampilkan Isi Graf

```
public void printGraph() {
    System.out.println("Graf Asal (Adjacency List):");
    for (String node : graph.keySet()) {
        System.out.print(node + " -> ");
        List<String> neighbors = graph.get(node);
        System.out.println(String.join(", ", neighbors));
    }
    System.out.println();
}
```

4. Untuk Penelusuran DFS (Rekursif)

```
public void dfs(String start) {
    Set<String> visited = new HashSet<>();
    System.out.println("Penelusuran DFS:");
    dfsHelper(start, visited);
    System.out.println();
}

private void dfsHelper(String current, Set<String> visited) {
    if (visited.contains(current)) return;
    visited.add(current);
    System.out.print(current + " ");
    for (String neighbor : graph.getOrDefault(current, new ArrayList<>())) {
        dfsHelper(neighbor, visited);
```

5. Penelusuran BFS (Iteratif)

```
public void bfs(String start) {
    Set<String> visited = new HashSet<>();
    Queue<String> queue = new LinkedList<>();
    queue.add(start);
    visited.add(start);

    System.out.println("Penelusuran BFS:");
    while (!queue.isEmpty()) {
        String current = queue.poll();
        System.out.print(current + " ");
        for (String neighbor : graph.getOrDefault(current, new ArrayList<>())) {
            if (!visited.contains(neighbor)) {
                queue.add(neighbor);
                visited.add(neighbor);
            }
```

6. Main Method

```
public static void main(String[] args) {
    GraphTraversal graph = new GraphTraversal();

    // Contoh graf: A-B, A-C, B-D, B-E
    graph.addEdge("A", "B");
    graph.addEdge("A", "C");
    graph.addEdge("B", "D");
    graph.addEdge("B", "E");

    System.out.println("Graf Asal adalah: ");
    graph.printGraph();

    // Lakukan penelusuran
    graph.dfs("A");
    graph.bfs("A");
}
```

D. Kesimpulan

Melalui praktikum ini, mahasiswa memperoleh pemahaman yang lebih dalam tentang bagaimana struktur data Binary Tree dan Grafh bekerja. Dan bagaimana cara kerja penelusuran bekerja dengan visualisasi tiap-tiap langkah dari penelusuran tersebut