VERSION 1.0

FEBRUARI 8, 2024

[STRUKTUR DATA]

MODUL 5, TREE

DISUSUN OLEH:
IZZA IHSAN FATHONY
MOCH IQBAL ARIZKI WIDYANSYAKH

DIAUDIT OLEH:
MUHAMMAD ILHAM PERDANA. S.TR.T., M.T.

LAB. INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

[STRUKTUR DATA]

PERSIAPAN MATERI

Mahasiswa diharapkan mempelajari materi sebelum mengerjakan tugas, materi yang tercakup antara lain:

1. Tree
2. Binary Tree
3. Binary Search Tree

TUJUAN

Mahasiswa mampu menguasai dan menjelaskan konsep dari Struktur Data Tree.

TARGET MODUL

Mahasiswa mampu memahami dan menerapkan Tree beserta contohnya:

PERSIAPAN SOFTWARE/APLIKASI

1. Java Development Kit
2. Java Runtime Environment
3. IDE (Intellij IDEA, Eclipse, Netbeans, dll)

REFERENSI MATERI

Geekgorgeeks:

https://www.geeksforgeeks.org/introduction-to-binary-tree-data-structure-and-algorithm-tutorials/

Youtube (Indonesia):

https://www.youtube.com/watch?v=YISN5L-7Nz0&ab_channel=IsnaAlfi

Youtube (English):

https://www.youtube.com/watch?v=FxaZu9KeKyI&ab_channel=GreatLearning

Javapoint:

https://www.javatpoint.com/tree

Edureka:

https://www.edureka.co/blog/java-binary-tree

Note: Dari referensi tersebut mungkin terdapat sedikit perbedaan satu sama yang lain, cukup pahami konsepnya dan terapkan pada kasus di modul ini.

MATERI POKOK

Mengapa belajar Tree dan Binary Tree:

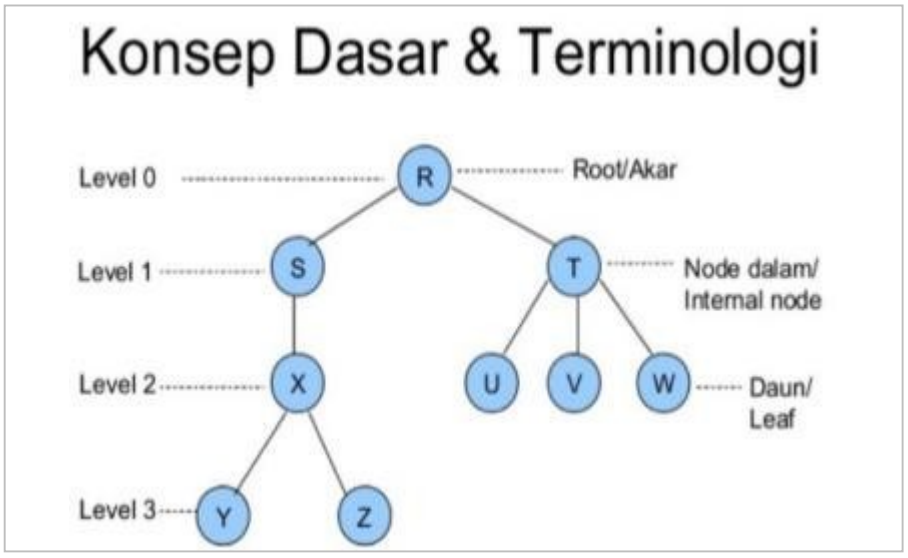
Sebelum masuk ke materi, kita harus mengetahui tujuan belajar kita, tujuan kita mempelajari Tree dan Binary Tree adalah untuk memperdalam struktur data kita, Binary Tree dan Tree banyak diimplementasikan dalam beberapa algoritman salah satunya seperti Binary Sort dan lain-lainnya. Struktur data ini juga banyak diimplementasikan pada aplikasi Backend. Selain itu jenis struktur data ini sering sekali dijadikan studi kasus dalam Technical Interview ketika kita ingin bekerja pada suatu Perusahaan khususnya untuk posisi Backend Developer.

Penjelasan:

1. Tree

Tree merupakan salah satu bentuk struktur data tidak linear yang menggambarkan hubungan yang bersifat hierarki antara elemen-elemen. Tree didefinisikan sebagai kumpulan simpul (node) dengan salah satu simpul yang dijadikan akar (root). Simpul lainnya terbagi menjadi himpunan yang saling tak berhubungan satu sama lain (subtree).

Node-node tersebut dihubungkan oleh sebuah vector. Setiap node dapat memiliki 0 atau lebih node anak (child). Sebuah node yang memiliki node anak disebut node induk (parent). Sebuah node anak hanya memiliki satu node induk. Sesuai konvensi ilmu computer, tree tumbuh ke bawah, tidak seperti pohon di dunia nyata yang tumbuh ke atas. Dengan demikian, node anak akan digambarkan berada dibawah node induknya. Berikut adalah contoh ilustrasi bentuk tree.



Istilah Pada Tree

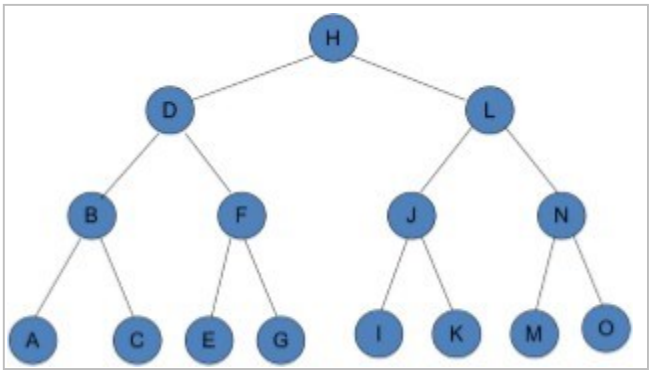
Istilah	Keterangan
Predecessor	Node yang berada di atas node tertentu
Successor	Node yang berada di bawah tertentu
Ancestor	Seluruh node yang terletak sebelum node tertentu dan terletak pada jalur yang sama
Descendant	Seluruh node yang terletak setelah node tertentu dan terletak pada jalur yang sama
Parent	Predecessor satu level di atas suatu node
Child	Successor satu level di bawah suatu node
Sibling	Node-node yang memiliki parent yang sama
Subtree	Suatu node beserta descendant-nya
Size	Banyaknya node dalam suatu tree
Height	Banyaknya tingkatan dalam suatu tree
Root	Node khusus yang tidak memiliki predecessor
Leaf	Node-node dalam tree yang tidak memiliki successor
Degree	Banyaknya child dalam suatu

2. Binary Tree

Binary Tree adalah tree dengan syarat bahwa tiap node hanya boleh memiliki maksimal 2 subtree dan kedua subtree tersebut harus dipisah. Sesuai dengan definisi tersebut, maka tiap node dalam Binary Tree hanya boleh memiliki paling banyak dua child. Berikut adalah jenis-jenis Binary Tree:

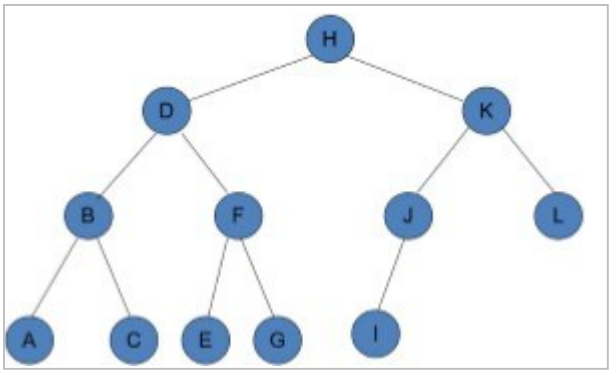
a. Full Binary Tree

Merupakan Binary Tree yang tiap node-nya (kecuali leaf) memiliki dua child dan tiap subtree harus mempunyai Panjang path yang sama. Berikut contohnya:



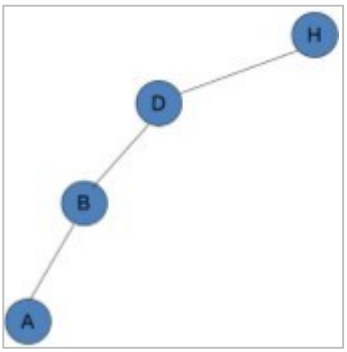
b. Complete Binary Tree

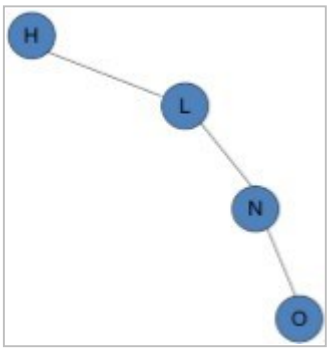
Complete Binary Tree ini hampir sama dengan Full Binary Tree, namun siap subtree boleh memiliki panjang path yang berbeda. Node kecuali leaf memiliki 0 atau 2 child. Berikut contohnya:



c. Skewed Binary Tree

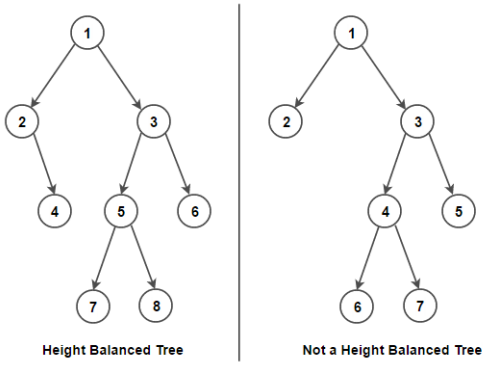
Merupakan Binary Tree yang semua node-nya (kecuali leaf) hanya memiliki satu child. Berikut contohnya:





d. Balanced Binary Tree

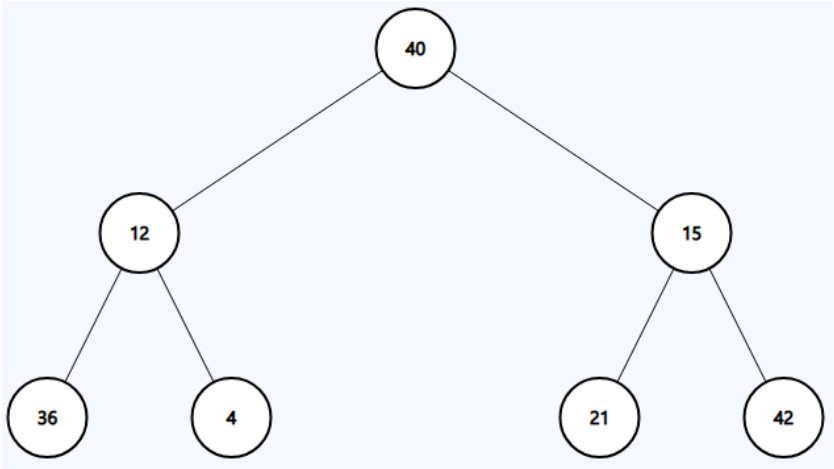
Balanced Binary Tree merupakan tree yang memiliki perbedaan jumlah node pada subtree kiri dan kanannya maksimal satu atau tinggi nya maksimal harus berselisih satu saja. Berikut contohnya:



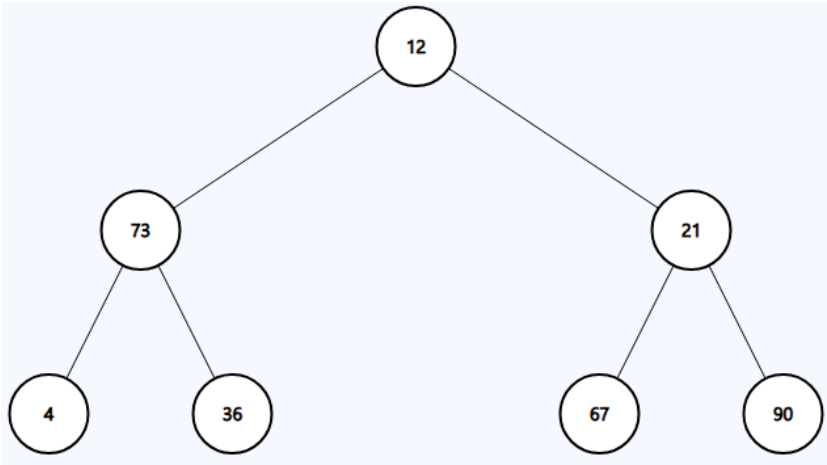
3. Binary Search Tree

Binary Search Tree merupakan jenis khusus dari Binary Tree yang menyimpan data dalam urutan terurut. Binary Search Tree akan memudahkan pencarian, penambahan dan penghapusan data. Ciri khas yang ada dalam Binary Search ada pada di setiap node-nya. Semua nilai di subtree kanan pasti lebih besar dari nilai node di subtree kiri. Binary Search Tree sangat berguna dalam aplikasi yang memerlukan pencarian data yang cepat dan efisien. Dibawah ini merupakan gambar perbedaan diantara Binary Tree dengan Binary Search Tree.

Binary Tree :



Binary Search Tree :

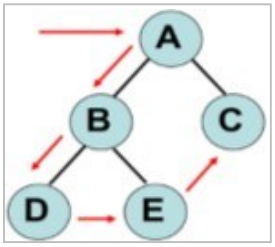


4. Binary Tree Traversal

Binary Tree Traversalah adalah proses mengunjungi node tepat satu kali dan tiap node hanya boleh memiliki maksimal 2 subtree yang disebut sub pohon kiri (left subtree) dan sub pohon kanan (right subtree). Dengan melakukan kunjungan secara lengkap, maka akan didapatkan

urutan informasi secara linier yang tersimpan dalam sebuah Binary Tree. Terdapat 3 metode dalam Binary Tree Traversal, yaitu:

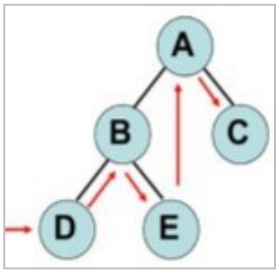
A. PreOrder



Urutan PreOrder:

- Cetak isi simpul yang dikunjungi (root)
- Kunjungi cabang kiri
- Kunjungi cabang kanan

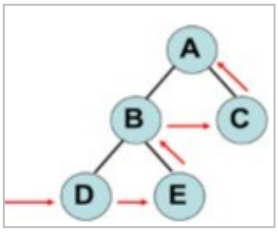
B. InOrder



Urutan InOrder:

- Kunjungi cabang kiri
- Cetak isi simpul yang dikunjungi (root)
- Kunjungi cabang kanan

C. PostOrder



Urutan PostOrder:

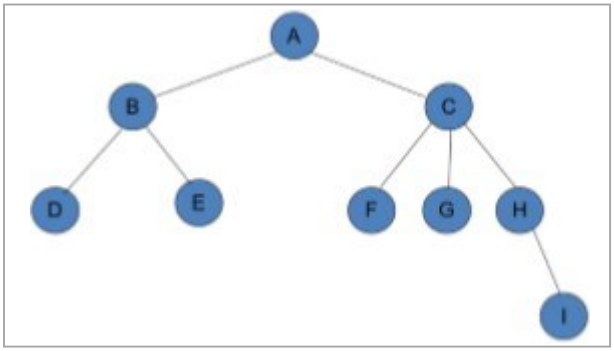
- Kunjungi cabang kiri
- Kunjungi cabang kanan
- Cetak isi simpul yang dikunjungi (root)

Kenapa Menggunakan Tree?

Tree merupakan teori yang sangat bermanfaat dalam struktur data karena dapat digunakan sebagai struktur dalam penyimpanan data yang baik dalam berbagai kasus. Dapat digunakan untuk menyimpan dan mencari data dalam teknik pemrograman serta bagaimana cara struktur data Tree dapat dimanfaatkan untuk menyimpan dan mencari data dengan cepat dan efisien serta mengurangi bug dalam komputer.

Contoh dan Istilah Dalam Binary Tree:

Disediakan gambar tree sepeperti berikut:



Dengan melihat gambar diatas, dapat diketahui bahwa:

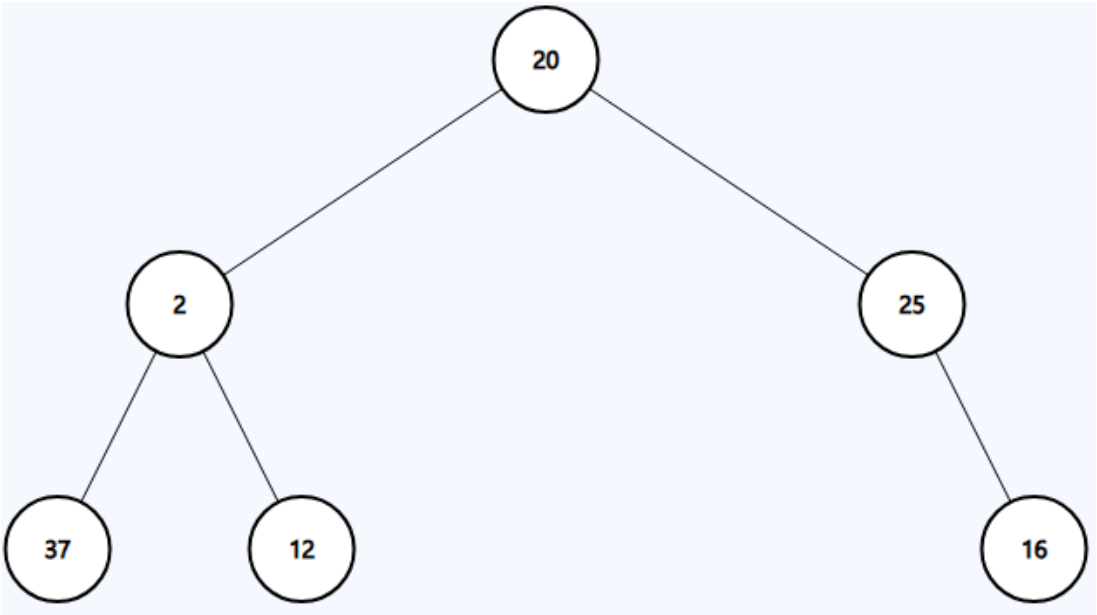
a. Predecessor (F) = **A, B, C**
b. Successor (B) = **D, E, F, G, H, I**
c. Ancestor (F) = **C, A**
d. Descendant (B) = **D, E**
e. Parent (I) = **H**
f. Child (C) = **F, G, H**
g. Sibling (G) = **F, H**
h. Degree (C) = **3**
i. Height = **4**
j. Root = **A**
k. Leaf = **D, E, F, G, I**
l. Size = **9**

*Jika dilihat dari soal diatas, yang menjadi node patokan adalah didalam tanda kurung ().

CODELAB

LATIHAN 1

Membuat Code Binary Tree Seperti Gambar Dibawah Ini



Selanjutnya Membuat Tree Traversal Code-nya

a. Membuat class Node.java untuk deklarasikan node

```java
public class Node {
    int data;
    Node left;
    Node right;

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

b. Membuat class BinaryTree.java dan menambahkan proses traversal kedalam (PreOrder, InOrder, dan PostOrder)

```java
public class BinaryTree {
    public Node root;

    public BinaryTree() {
        root = null;
    }

    // Fungsi untuk menambahkan node secara manual ke tree
    public void addRoot(int data) {
        root = new Node(data);
    }

    public void inOrder(Node node) {
        if (node != null) {
            inOrder(node.left);
            System.out.println(node.data + " ");
            inOrder(node.right);
        }
    }
```

```
    public void preOrder(Node node) {
        if (node != null) {
            System.out.println(node.data + " ");
            preOrder(node.left);
            preOrder(node.right);
        }
    }

    public void postOrder(Node node) {
        if (node != null) {
            postOrder(node.left);
            postOrder(node.right);
            System.out.println(node.data + " ");
        }
    }

    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();

        // Menentukan struktur tree secara manual
        tree.addRoot(20); // Root
        tree.root.left = new Node(2); // Menambahkan node ke kiri root
        tree.root.right = new Node(25); // Menambahkan node ke kanan root
        tree.root.left.left = new Node(37); // Menambahkan node ke kiri dari node kiri root
        tree.root.left.right = new Node(12); // Menambahkan node ke kiri dari node kiri root
        tree.root.right.right = new Node(16); // Menambahkan node ke kanan dari node kanan root

        System.out.println("\nPre Order: ");
        tree.preOrder(tree.root);
        System.out.println("\nIn Order: ");
        tree.inOrder(tree.root);
        System.out.println("\nPost Order: ");
        tree.postOrder(tree.root);
    }
}
```

LATIHAN 2

Contoh Kasus Merubah Urutan Data Menjadi Binary Search Tree

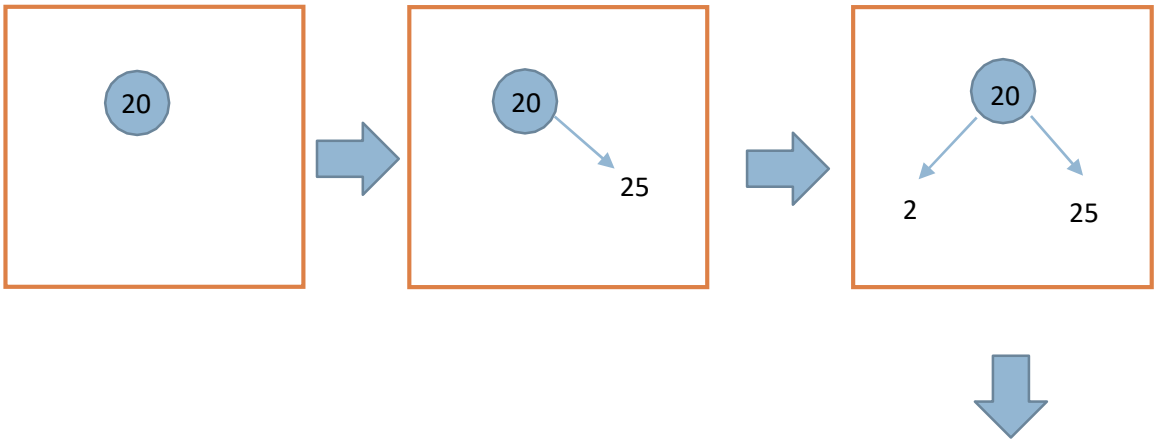
Aturan umum:

a. Jika binary Tree masih kosong, maka nilai pertama langsung menjadi root
b. Jika nilai berikutnya yang akan dimasukkan lebih kecill (<) dari node yang sudah ada, maka diletakan di bagian kiri bawah node yang sudah ada
c. Jika nilai berikutnya yang akan dimasukkan lebih besar atau sama dengan (>=) dari node yang sudah ada maka diletakkan di bagian kanan bawah node yang sudah ada tersebut
d. Pengecekan untuk peletakkan node dilakukan berulang-ulang sampai nilai yang akan dimasukkan tersebut menjadi leaf

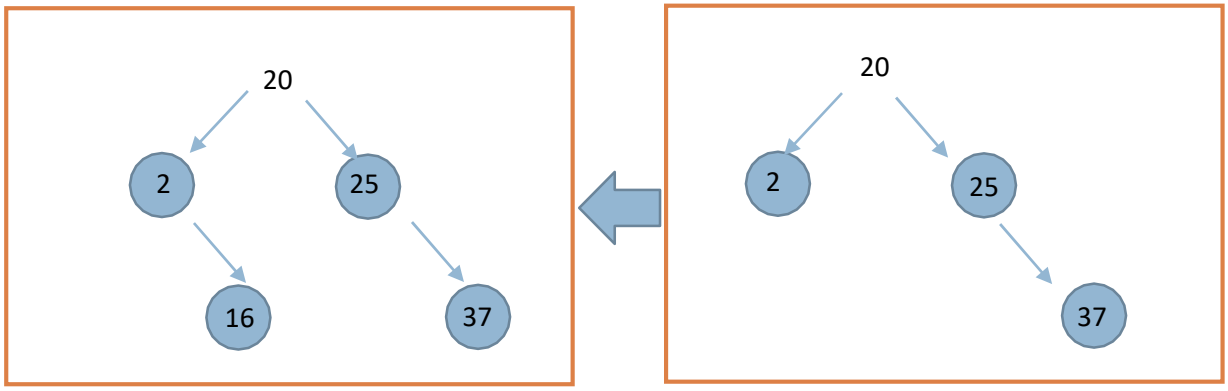
Contoh:

Urutan data: 20 2 25 37 16

Jika diubah menjadi Binary Search Tree yaitu:



HASIL AKHIR:



NB: Jika data berupa selain angka, maka pengurutannya berdasarkan nilai ASCII

Tree Traversal Source Code

Dari data pohon yang sudah diatas, dapat dibuat dalam program untuk menentukan Tree Traversal seperti berikut:

a. Membuat class Node. Java untuk deklarasikan node

```
public class Node {
    int data;
    Node left;
    Node right;

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

b. Membuat class BinaryTree.java dan menambahkan proses traversal kedalam tree (PreOrder, InOrder, dan PostOrder)

```java
public class BinaryTree {
    public Node root;

    public void NewNode(int data) {
        root = NewNode(root, new Node(data));
    }

    private Node NewNode(Node root, Node newData) {
        if (root == null) {
            root = newData;
            return root;
        }

        if (newData.data < root.data) {
            root.left = NewNode(root.left, newData);
        } else {
            root.right = NewNode(root.right, newData);
        }
        return root;
    }

    public void inOrder(Node node) {
        if (node != null) {
            inOrder(node.left);
            System.out.println(node.data + " ");
            inOrder(node.right);
        }
    }
```

```
    public void inOrder(Node node) {
        if (node != null) {
            inOrder(node.left);
            System.out.println(node.data + " ");
            inOrder(node.right);
        }
    }

    public void preOrder(Node node) {
        if (node != null) {
            System.out.println(node.data + " ");
            preOrder(node.left);
            preOrder(node.right);
        }
    }

    public void postOrder(Node node) {
        if (node != null) {
            postOrder(node.left);
            postOrder(node.right);
            System.out.println(node.data + " ");
        }
    }

    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();

        tree.NewNode(20);
        tree.NewNode(2);
        tree.NewNode(25);
        tree.NewNode(37);
        tree.NewNode(16);

        System.out.println("\nPre Order: ");
        tree.preOrder(tree.root);
        System.out.println("\nIn Order: ");
        tree.inOrder(tree.root);
        System.out.println("\nPost Order: ");
        tree.postOrder(tree.root);
    }
}
```

c. Perhatikan code diatas, apakah ada perbedaan antara logika dari code Binary Tree dengan Binary Search Tree? Jika ada tunjukkan bagian mana yang berbeda dan jelaskan kepada asisten masing-masing.

TUGAS

TUGAS 1

Buatlah sebuah source code (program kreasi sendiri) yang dapat menerima inputan dari user berupa kombinasi (bisa memanfaatkan huruf dan angka) dan gambarkan **Binary Tree**-nya. Program yang dibuat harus bisa menerapkan konsep Tree Traversal (PreOrder, InOrder, dan PostOrder).

Dari gambar Binary Tree yang sudah kalian buat, berikan jawaban dengan jelas dari pertanyaan yang diberikan oleh asisten (pertanyaan akan diberikan langsung pada saat praktikum, pertanyaan berhubungan dengan istilah pada BinaryTree. Berikut adalah contoh pertanyaan yang akan ditanyakan oleh asisten:

a. Predessor
b. Sucessor
c. Ancestor

*Pastikan untuk mempelajari dan memahami semua istilah binary Tree agar dapat menjawab pertanyaan dengan benar. Asisten diperbolehkan memberikan maksimal 8 poin pertanyaan

NB:

a. Arti dari kreasi sendiri adalah topik program yang diambil berasal dari pemikiran praktikan masing-masing dan pastinya setiap orang akan memiliki pemikiran yang berbeda.
b. Inputan kombinasi dapat bersifat opsional jika praktikan dapat memberikan alasan yang jelas dan bisa digantikan dengan pemahaman materinya.

TUGAS 2

Buatlah sebuah source code program sistetm manajemen inventaris toko buku. Dalam sistem ini, kita akan menggunakan **Binary Search Tree (BST)** untuk menyimpan buku berdasarkan ISBN (International Standard Book Number) sebagai **kunci**. Program harus bisa melakukan beberapa fitur seperti:

1. Menambahkan buku ke inventaris
2. Mencari buku berdasarkan ISBN
3. Menampilkan semua buku dalam inventaris secara terurut berdasarkan ISBN menggunakan konsep Tree Traversal (PreOrder, InOrder, dan PostOrder)

Berikut adalah contoh output program:

```
Inventaris Buku (terurut berdasarkan ISBN - InOrder):
21 Phyton Programming
123 Java Programming
143 Statistics
456 Data Structures and Algorithms
789 Computer Networks

Inventaris Buku (PreOrder):
123 Java Programming
21 Phyton Programming
456 Data Structures and Algorithms
143 Statistics
789 Computer Networks

Inventaris Buku (PostOrder):
21 Phyton Programming
143 Statistics
789 Computer Networks
456 Data Structures and Algorithms
123 Java Programming

Pencarian Buku:
Buku ditemukan: ISBN = 456, Judul = Data Structures and Algorithms
```

Cacatan

Aturan umum penulisan JAVA agar mudah dikoreksi oleh asisten:

1. Untuk nama class, enum, dan yang lainnya biasakan menggunakan CamelCase(diawali dengan huruf besar pada tiap kata untuk mengganti spasi) seperti: JalanRaya, ParkiranGedung, dll.
2. Untuk penulisan nama method dan attribute diawali dengan huruf kecil di awal kata dan menggunakana huruf besar untuk kata setelahnya, seperti: getNamaJalan, namaJalan, dll.
3. Jika menggunakan IDE Inteliij jangan lupa untuk memformat kode agar terlihat rapi menggunakan menu code -> show reformat file dialog -> centang semua field dan klik ok.

KRITERIA & DETAIL PENILAIAN

Kriteria	Nilai
Codelab 1	10
Codelab 2	10
Tugas 1	30
Tugas 2	30

Pemahaman	20
Total	**100**