VERSI 2.2

DESEMBER, 2020


# [STRUKTUR DATA]

MODUL 5, Tree

TIM PENYUSUN: - DOSEN
- DICKY PRABOWO OCTIANTO

PRESENTED BY: LAB. TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH MALANG

## CAPAIAN PEMBELAJARAN MATA KULIAH

Mahasiswa mampu menguasai & menjelaskan konsep struktur data dari tree

## SUB CAPAIAN PEMBELAJARAN MATA KULIAH

1. Mahasiswa mampu memahami dan menerapkan tree beserta contohnya

## KEBUTUHAN HARDWARE & SOFTWARE

- Java Development Kit
- Java Runtime Environment
- IDE (Intellij IDEA, Eclipse, Netbeans, dll.)

## PERSYARATAN PEMAHAMAN

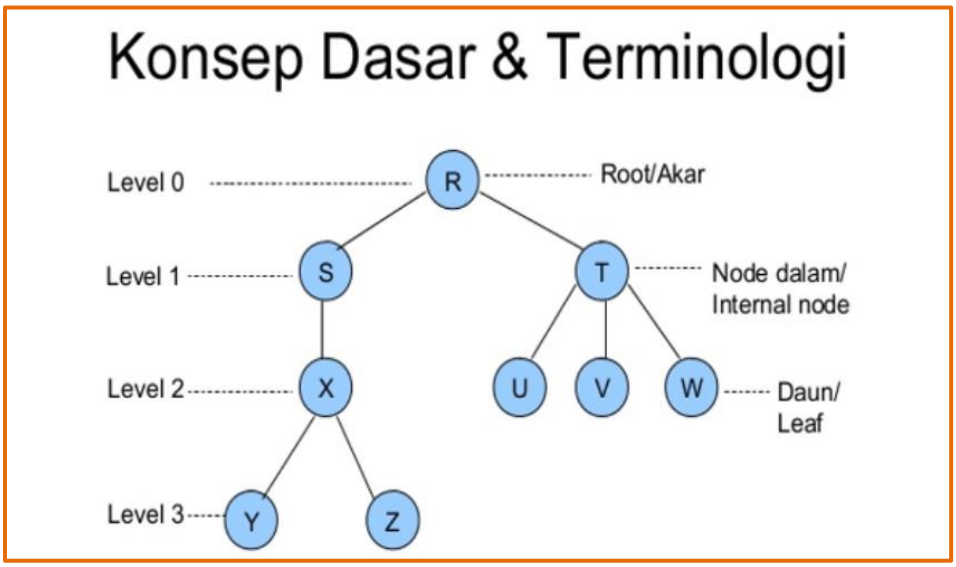
1. Tree
2. Binary Tree

## MATERI POKOK

### 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 vektor. 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 komputer, tree bertumbuh ke bawah, tidak seperti pohon di dunia nyata yang tumbuh ke atas. Dengan demikian node anak akan digambarkan berada di bawah node induknya.




**1.1 Gambaran Tree**

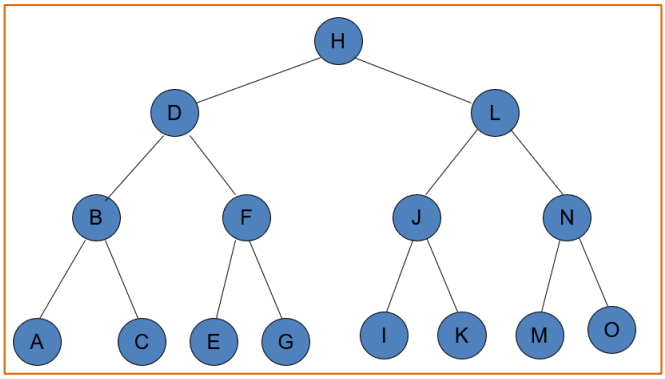
| | |
|---|---|
| Predecesor | Node yang berada di atas node tertentu |
| Successor | Node yang berada di bawah node tertentu |
| Ancestor | Seluruh node yang terletak sebelum node tertentu dan terletak pada jalur yang sama |
| Descendant | Seluruh node yang terletak setelah node tertentu dan terletak 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 descendantnya |
| 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 node |

**1.2 Istilah – istilah pada tree**

## 2. Binary Tree

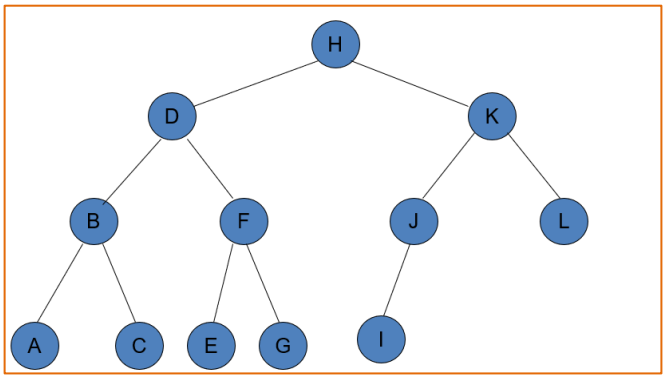
Binary Tree adalah tree dengan syarat bahwa tiap node hanya boleh memiliki maksimal 2 subtree dan kedua subtree tersebut harus terpisah. Sesuai dengan definisi tersebut, maka tiap node dalam binary tree hanya boleh memiliki paling banyak dua child. Jenis-jenis Binary Tree

a) **Full Binary Tree,** yaitu Binary Tree yang tiap nodenya ( kecuali leaf) memiliki dua child dan tiap subtree harus mempunyai panjang path yang sama.



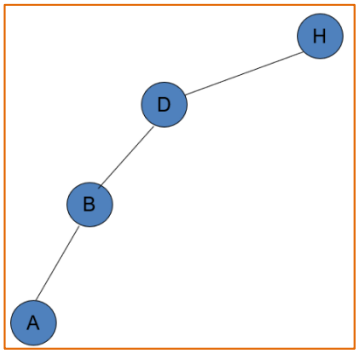

2.a Full Binary Tree

b) **Complete Binary Tree,** Mirip dengan Full Binary Tree, namun tiap subtree boleh memiliki panjang path yang berbeda. Node kecuali leaf memiliki 0 atau 2 child.

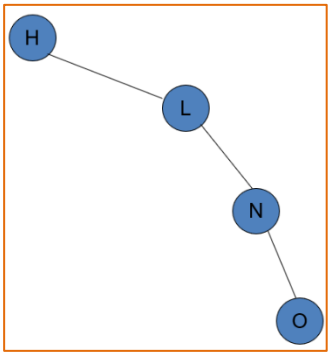



2.b Complete Binary Tree

c) **Skewed Binar Tree,** Yakni Binary Tree yang semua nodenya(kecuali leaf) hanya memiliki satu child.




2.c Left Skewed

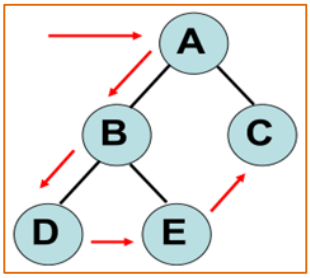



2.c Right Skewed

## 3. Binary Tree Traversal

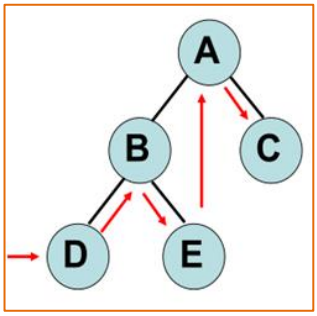
Binary tree traversal adalah proses mengunjungi node tepat satu kali dan tiap node hanya boleh memiliki maksimal 2 subtree yang disebut sebagai sub pohon kiri (left subtree) dan sub pohon kanan (right subtree).Dengan melakukan kunjungan secara lengkap, maka akan didapatkan urutan informasi secara linier yang tersimpan dalam sebuah binary tree. Terdapat 3 metode dalam Binary tree Traversal:

i) PreOrder



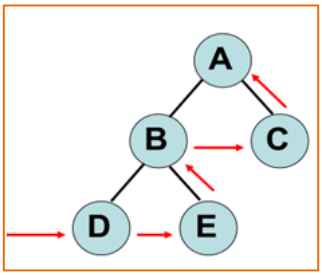

Urutan pada PreOrder:
a) Cetak isi simpul yang dikunjungi ( Root)
b) Kunjungi cabang kiri
c) Kunjungi Cabang Kanan

ii) InOrder




Urutan pada InOrder:
a) Kunjungi cabang kiri
b) Cetak isi simpul yang dikunjungi (Root)
c) Kunjungi cabang kanan

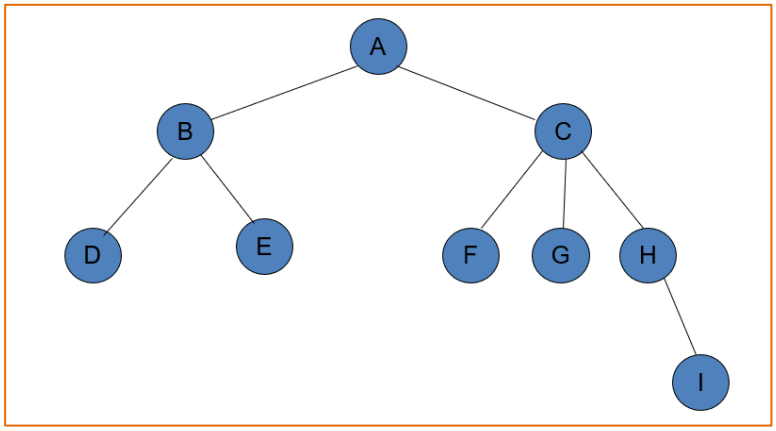
iii) PostOrder




Urutan pada PostOrder:
a) Kunjungi cabang kiri
b) Kunjungi cabang kanan
c) Cetak isi simpul yang dikunjungi (Root)

## MATERI PRAKTIKUM

### 1. Istilah – istilah dalam Binary Tree




Dari gambar tree di atas tentukanlah:

a. Predesesor (F) ?
b. Successor (B) ?
c. Ancestor (F) ?
d. Descendant (B) ?
e. Parent (I) ?
f. Child (C) ?
G. Sibling (G) ?
H. Degree (C) ?
I. Height ?
J. Root ?
K. Leaf ?
L. Size ?

**Jawab:**

a. Predesesor (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**

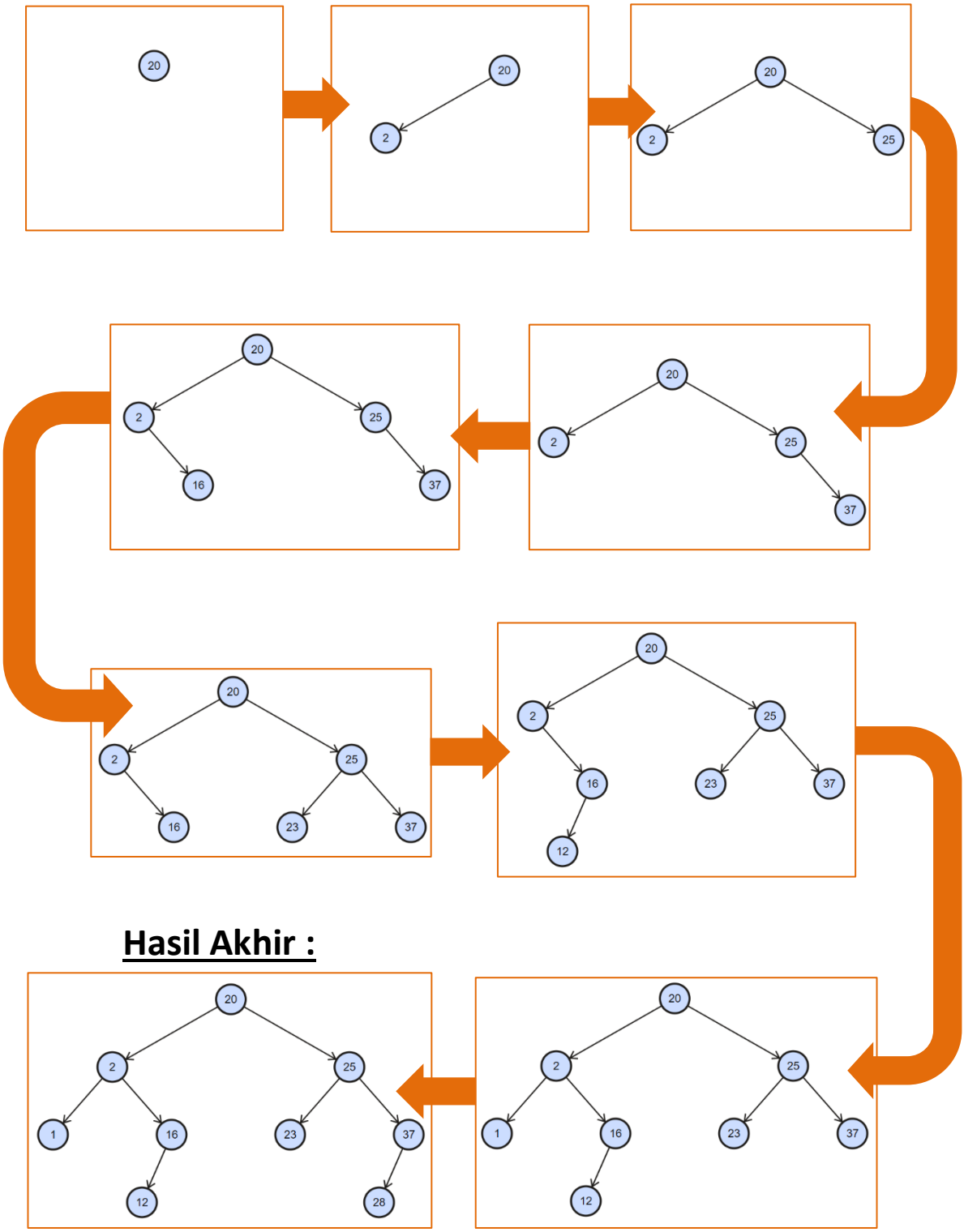
### 2. Merubah Urutan data menjadi Binary Tree

Aturan umum:

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

Urutan Data : 20 2 25 37 16 23 12 1 28

Urutan data tersebut dapat dibentuk menjadi sebagai berikut:




**Hasil Akhir :**




**NB: Jika data berupa selain angka maka perbandigan besar kecilnya beracuan pada nilai ASCII**

### 3. Tree Travelsal Source Code

Dari urutan data pada nomer 2, maka dapat dibentuk program untuk menentukan Tree Travelsal sebagai berikut:

a) Membuat Kelas Node.java untuk mendeklarasikan node yang ada

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

    public Node(char data) {
        this.data = data;
    }

}
```

b) Membuat kelas BinaryTree.java untuk memproses node menjadi urutan binary tree sesuai dengan konsep traversal (pre order, in order, dan post order)

```
public class BinaryTree {

    private 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 preOrder() {...3 lines }

    public void inOrder() {...3 lines }

    public void postOrder() {...3 lines }

    public void preOrder(Node root) {...7 lines }

    public void inOrder(Node root) {...7 lines }

    public void postOrder(Node root) {...7 lines }

}
```

c) Membuat Driver Class

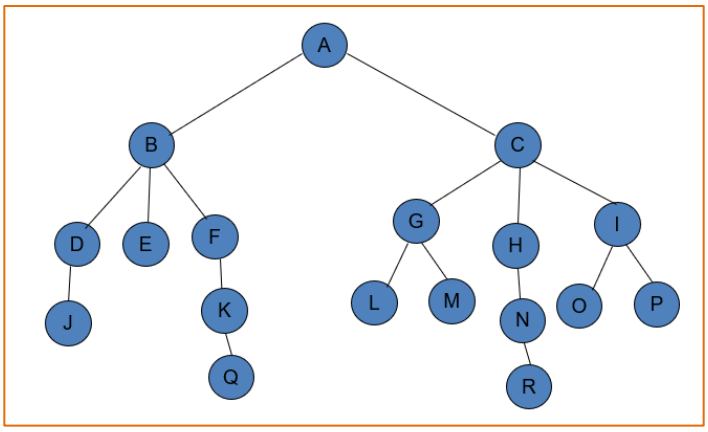
```
public static void main(String[] args) {
    BinaryTree binarytree = new BinaryTree();

    binarytree.NewNode(20);
    binarytree.NewNode(2);
    binarytree.NewNode(25);
    binarytree.NewNode(37);
    binarytree.NewNode(16);
    binarytree.NewNode(23);
    binarytree.NewNode(12);
    binarytree.NewNode(1);
    binarytree.NewNode(28);


    System.out.println("preorder :");
    binarytree.preOrder();
    System.out.println("inorder :");
    binarytree.inOrder();
    System.out.println("postorder :");
    binarytree.postOrder();
    System.out.println();
}
```

## LEMBAR KERJA

### KEGIATAN 1




Dari gambar Tree di atas, Sebutkan dan Jelaskan kepada Asisten:

a. Predesesor (J).
b. Successor (K).
c. Ancestor (R).
d. Descendant (G).
e. Parent (Q).
f. Child (H).
G. Sibling (F).
H. Degree (C).

### KEGIATAN 2

Buatlah sebuah program yang dapat menerima **inputan berupa String dari user**. Kemudian uraiankan inputan user tersebut sehingga dapat menerapkan Tree Traversal **(Preorder, Inorder, dan Postorder).** Setelah program Berhasil dijalankan, demo kan **Source Code dan gambarkan Binary tree-nya** dari inputan yang user inputkan (gambar dapat ditampilkan pada word / paint / etc).

```
Output - Modul5Strukdatt (run)
run:
Masukan inputan :INFORMATIKA
preorder :I F A A N M I K O R T
inorder :A A F I I K M N O R T
postorder :A A F K I M T R O N I
```

## CATATAN

Silahkan dikerjakan **tanpa copy – paste**, jika ada indikasi program yang kembar maka akan dilakukan **pengurangan poin**.

aturan umum penulisan bahasa JAVA agar mudah di koreksi oleh asisten:

1. Untuk nama kelas,interface,enum, dan yang lainnya biasakan menggunakan gaya CamelCase (diawali dengan huruf besar pada tiap kata untuk mengganti spasi) seperti: **K**ursi , **J**alan**R**aya, **P**arkiran**G**edung, dan lain seterusnya.
2. Untuk penulisan nama method, dan attribute diawali dengan huruf kecil di awal katadan menggunakan huruf besar untuk kata setelahnya, seperti: **g**et**N**ama**J**alan, **n**ama**J**alan, **h**arga, **s**et**N**ama**J**alan, dan lain seterusnya.
3. Jika menggunakan IDE Intellij jangan lupa untuk memformat penulisan kode agar terlihat rapi menggunakan menu code -> show reformat file dialog -> centang semua field dan klik ok.
4. Tidak boleh import library apapun selain Scanner dan ArrayList.

## RUBRIK PENILAIAN

| Soal | Nilai |
|---|---|
| Kegiatan 1: | 30% |
| Kegiatan 2: | 70% |