STRUKTUR DATA

Tree

PENGENALAN TREE

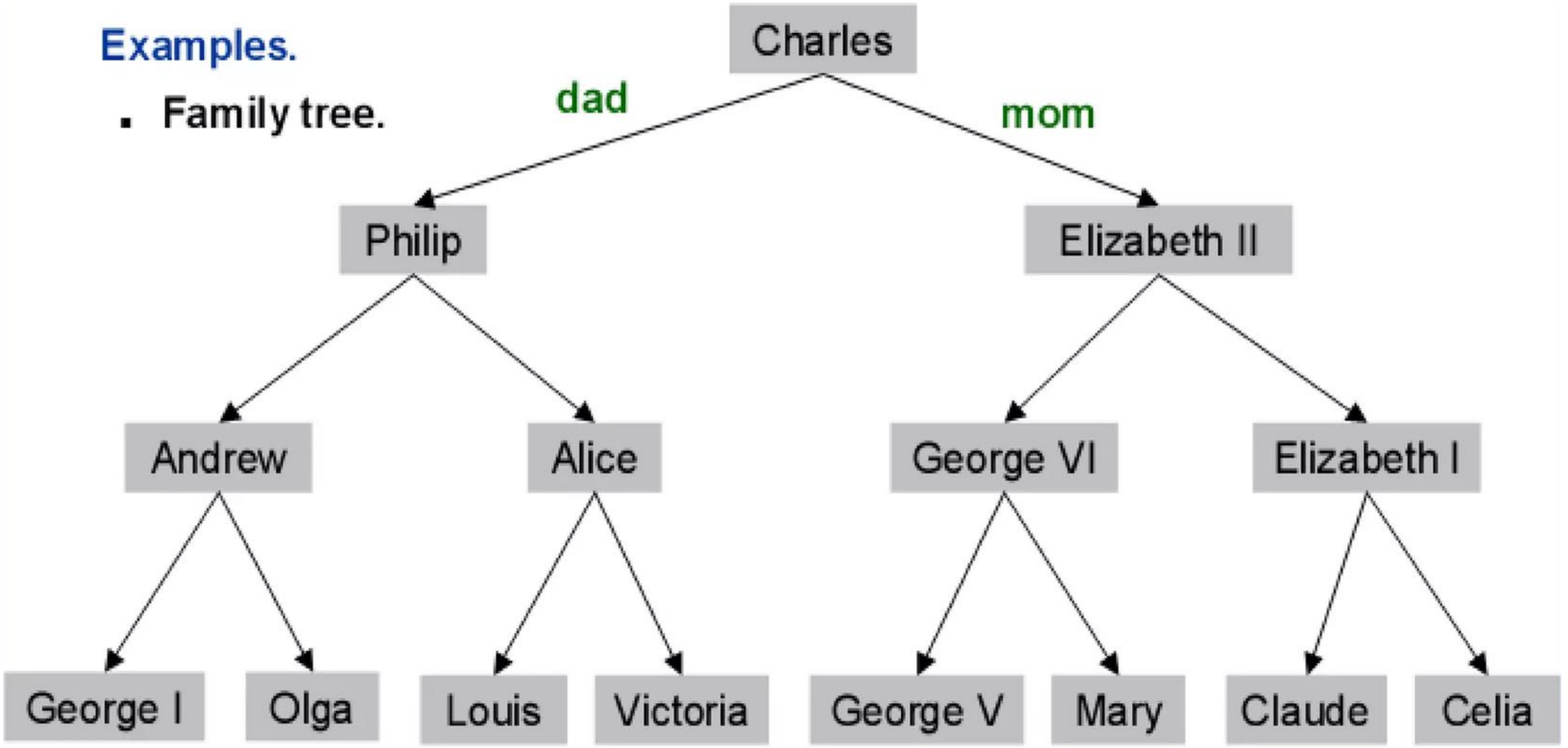
- Struktur pohon (tree) biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis atau hubungan bertingkat antara elemen-elemen yang ada.
- Terdapat sejumlah node (titik) yang terhubung atas susunan hirarkis parent (orangtua) dan child (anak).
- Sebuah child node pasti memiliki satu parent node
- Sebuah parent node bisa memiliki beberapa node lain yang berada dibawahnya (child node)

PENGENALAN TREE

Contoh penggunaan struktur pohon:

1. Silsilah keluarga
2. Alur proses
3. Klasifikasi
4. Struktur organisasi dari sebuah perusahaan
5. Hasil pertandingan yang berbentuk turnamen

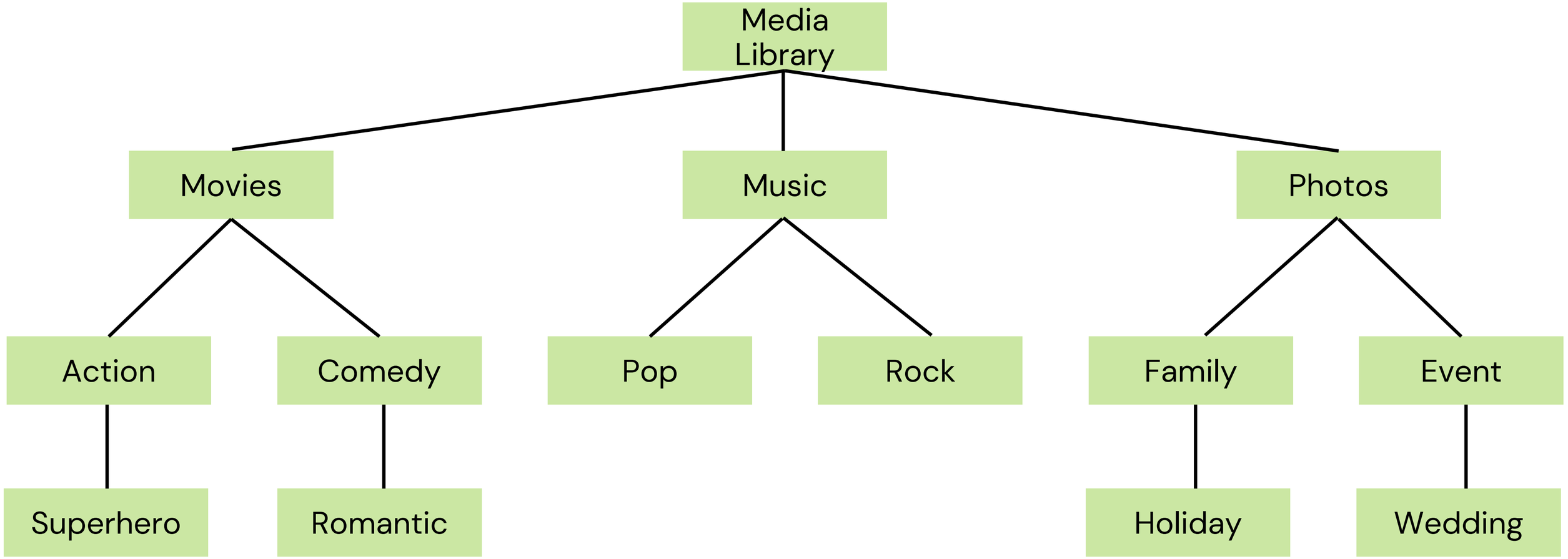
CONTOH TREE



Dalam struktur data tree, tidak bisa ada dua root. Tree didefinisikan sebagai struktur yang memiliki satu node paling atas yang disebut root, dan dari situ, node-node lainnya terhubung sebagai child. Jika ada dua node yang berfungsi sebagai root, maka itu bukan lagi sebuah tree,

CONTOH TREE

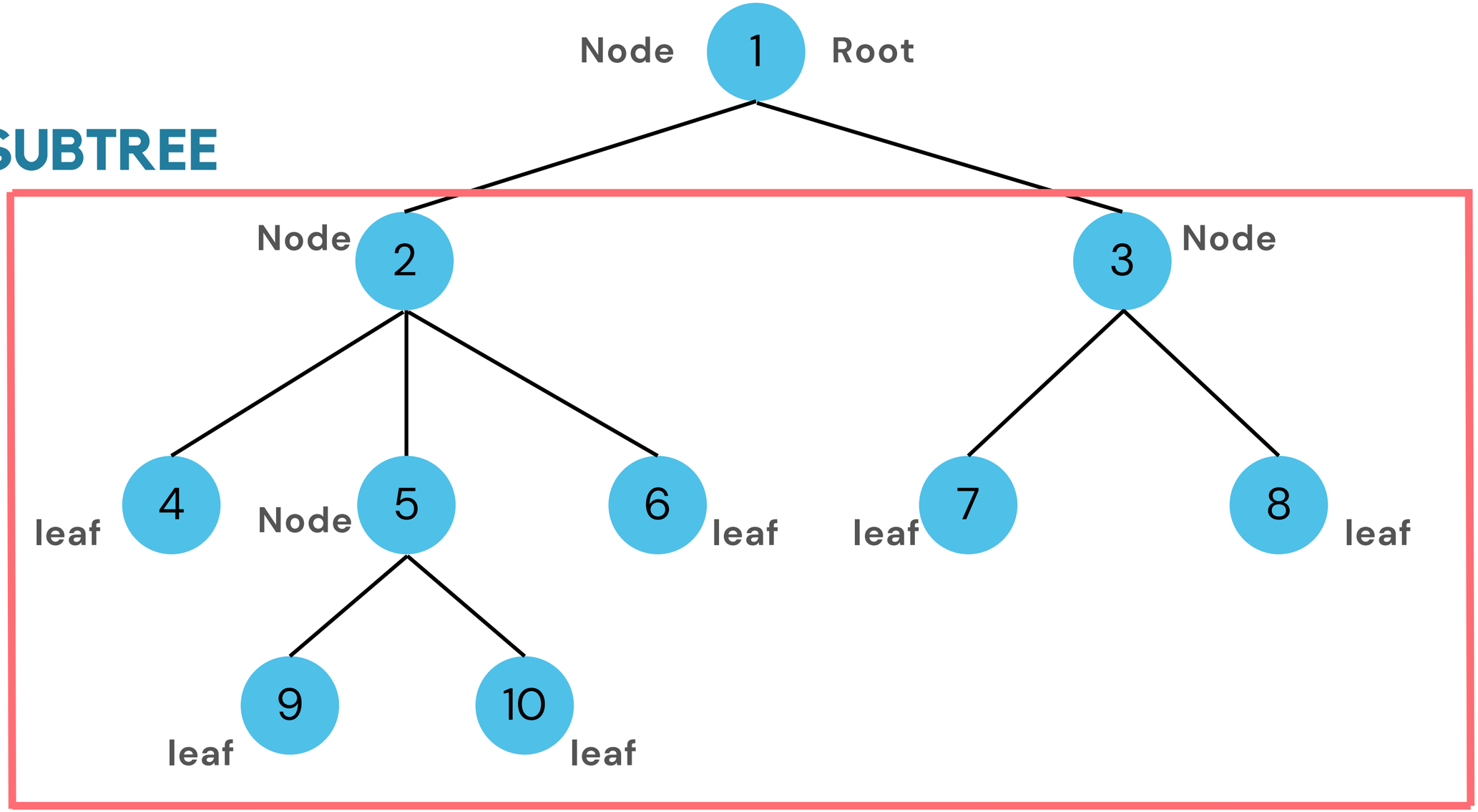
Contoh: direktori/folder pada windows atau linux



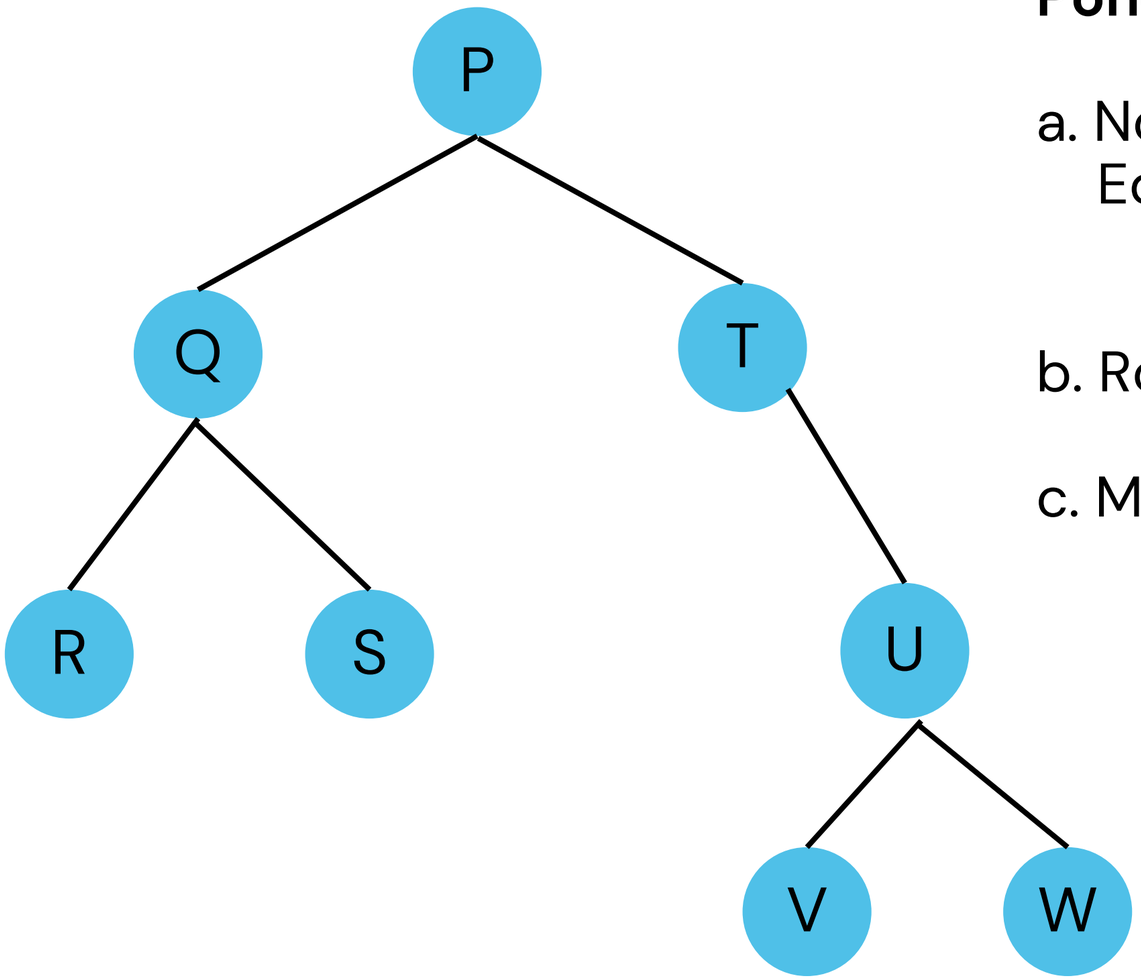
KOMPONEN PADA TREE

SUBTREE

Subtree adalah cabang dari pohon yaitu bagian dari tree yang dimulai dari suatu node tertentu hingga semua node di bawahnya.



CONTOH TREE



Pohon disamping memiliki:

a. Node sebanyak = 8 dan
Edge = n - I = 8 — I = 7 (P-Q, P-T, Q-R, Q-S, T-U, U-V, dan U-W)

b. Root pada Pohon T diatas adalah Simpul P

c. Mempunyai daun (Leaf) = 4, yaitu = R, S, V dan W

ISTILAH DALAM TREE

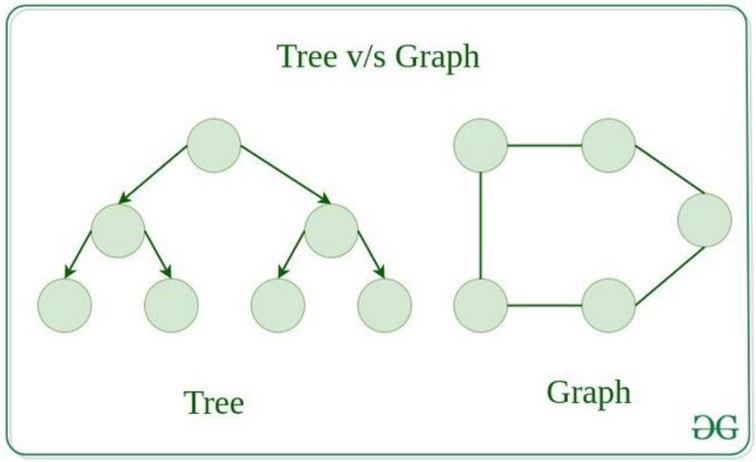
Term	Definition
Node	Sebuah elemen dalam sebuah tree; berisi sebuah informasi
Parent	Node yang berada di atas node lain secara langsung; Q adalah parent dari R dan S
Child	Cabang langsung dari sebuah node; D dan E merupakan children dari B
Root	Node teratas yang tidak punya parent (P)
Sibling	Sebuah node lain yang memiliki parent yang sama; Sibling dari Q adalah T karena memiliki parent yang sama yaitu P
Leaf	Sebuah node yang tidak memiliki children. R, S, V, W adalah leaf (*daun*). Leaf biasa disebut sebagai *external node*, sedangkan node selainnya disebut sebagai *internal node*. Q, B, T, U adalah *internal node*
Level	Semua node yang memiliki jarak yang sama dari root. P→level 1; Q,T→level 2; R, S, U→level 3; V, W →level 4
Depth	Jumlah level yang ada dalam tree
Complete	Semua parent memiliki children yang penuh
Balanced	Semua subtree memiliki depth yang sama

HUBUNGAN ANTAR KOMPONEN

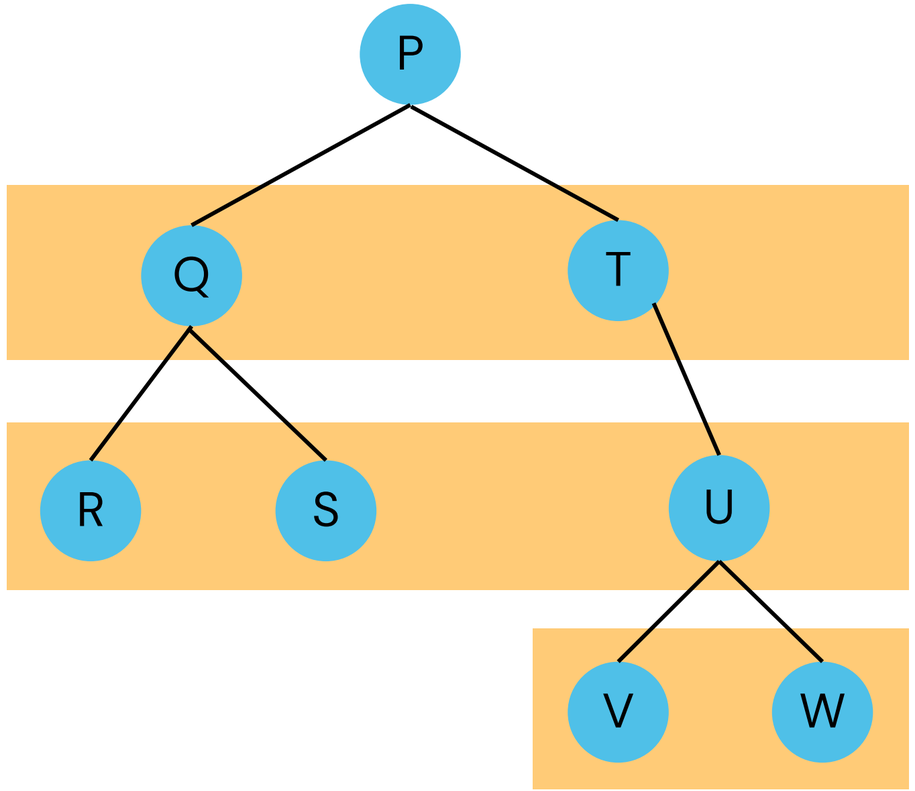
- Hubungan antar elemen: parent-child, father-son, mother-daughter
- Nama node: nama (angka) yang dipakai untuk membedakan sebuah node dengan node yang lain. Dalam kuliah ini adalah angka yang tertulis dalam lingkaran.
- Label: nilai yang diingat oleh sebuah node

Tree vs Graph

- Tree: setiap node kecuali root hanya memiliki sebuah parent
- Graph: dapat memiliki lebih dari sebuah parent



HUBUNGAN ANTAR KOMPONEN

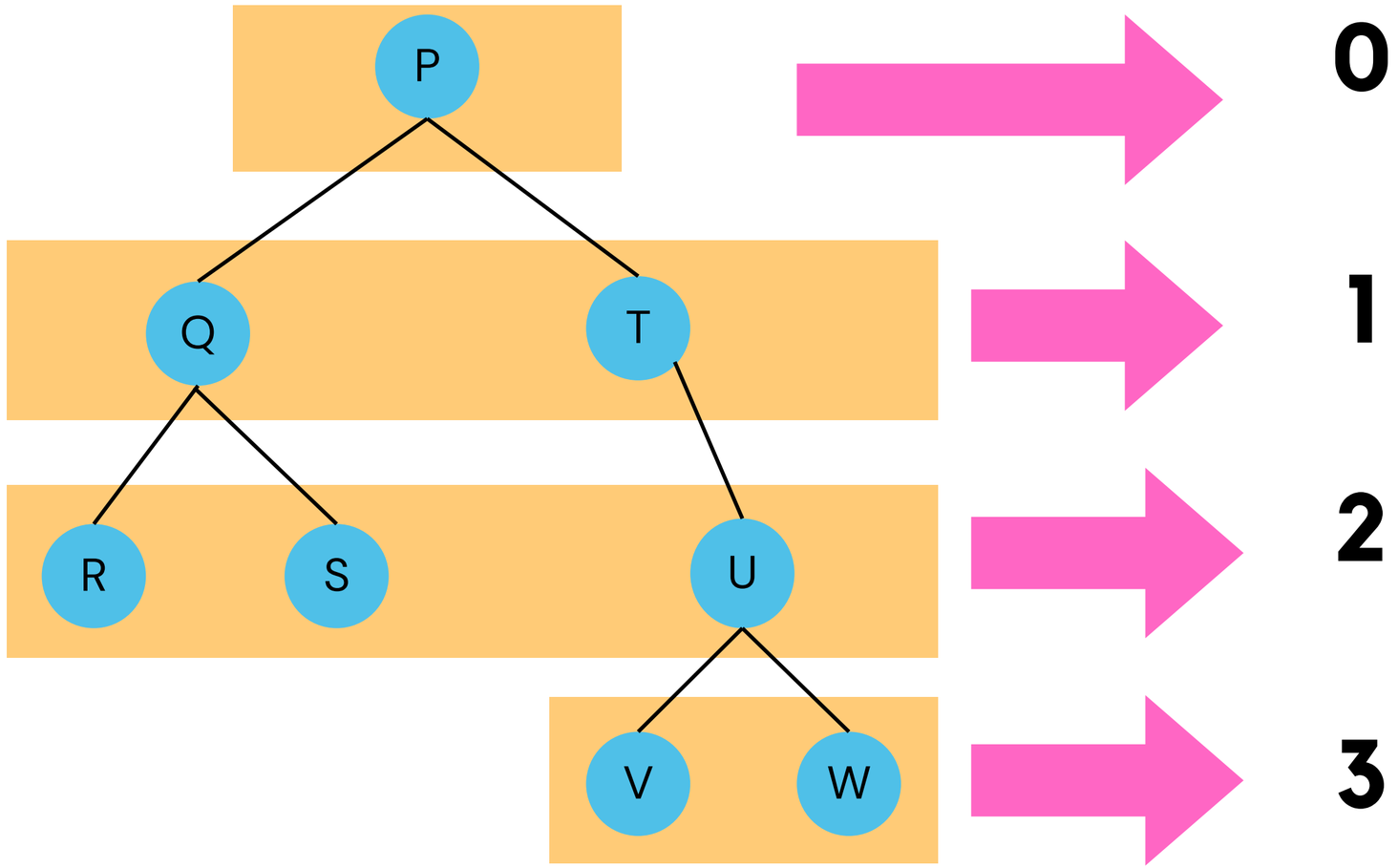


- **Sibling** : node-node yang memiliki parent yang sama

- **Ancestor (nenek moyang)** : semua node yang berada di jalur dari root hingga node tersebut. Termasuk node itu sendiri, parent, grandparent, dan seterusnya.
- Ancestor dari node R adalah: P, Q (dan R sendiri jika dihitung).
- Ancestor dari node T adalah: P (dan T sendiri).

- **Descendant (Keturunan):** semua node yang dapat dicapai dari node tersebut melalui child-nya. Termasuk child, grandchild, dan seterusnya.
- Descendant dari node Q adalah: R, S (dan Q sendiri jika dihitung).
- Descendant dari node P adalah: Q, T, R, S, U, V, W (dan P sendiri jika dihitung).

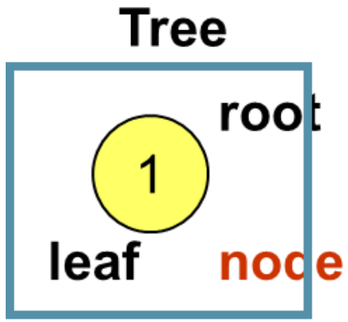
LEVEL



DEFINISI TREE

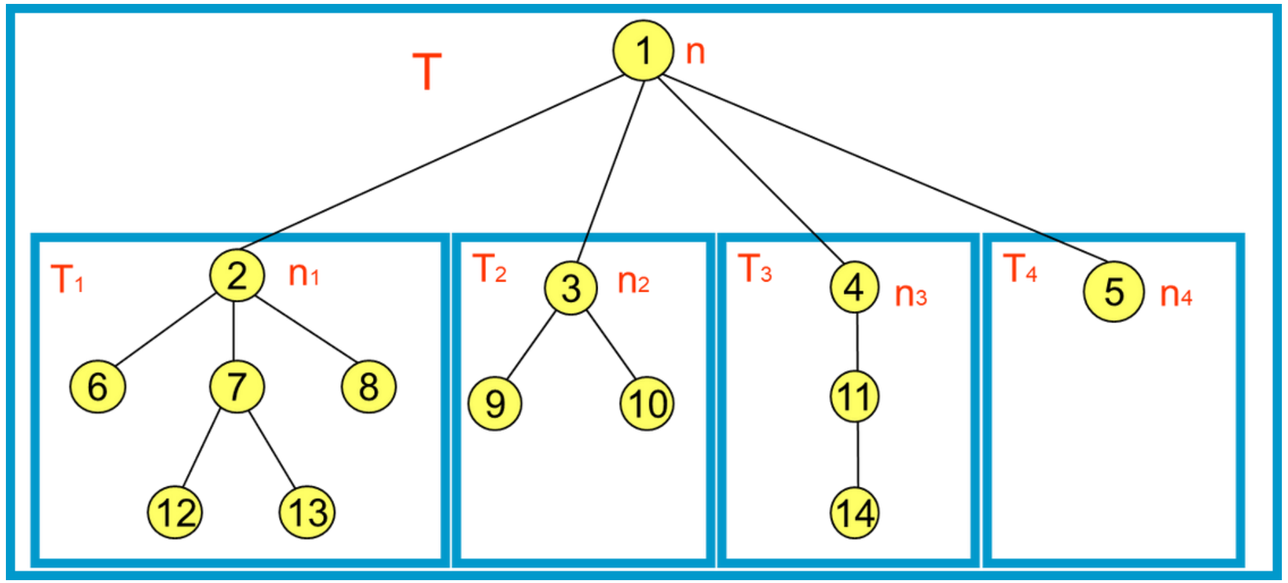
Sebuah tree didefinisikan sebagai struktur y ang dibentuk secara recursive oleh kedua rule berikut:

1. Sebuah node adalah sebuah tree. Node satu-satunya pada tree ini berfungsi sebagai root maupun leaf.



DEFINISI TREE

2. Dari k buah tree $T_1 \sim T_k$, dan masing-masing memiliki root $n_1 \sim n_k$.Jika node n adalah parent dari node$n_1 \sim n_k$, akan diperoleh sebuah tree baru T yang memiliki root n. Dalam kondisi ini, tree $T_1 \sim T_k$menjadi sub-tree dari tree T. Root dari sub-tree $n_1 \sim n_k$adalah child dari node n .



ORDERED VS UNORDERED TREE

Ordered tree

- Antar sibling terdapat urutan “usia”
- Node yang paling kiri berusia paling tua (sulung), sedangkan node yang paling kanan berusia paling muda (bungsu)
- Posisi node diatur atas urutan tertentu

Unordered tree

- Antar sibling tidak terdapat urutan tertentu

BINARY TREE

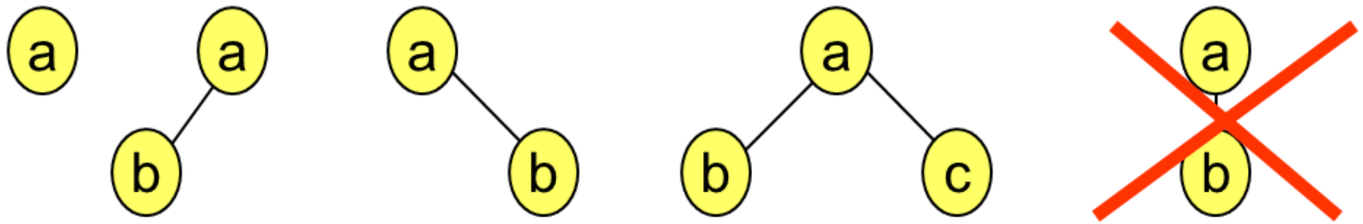
DEFINISI

Definisi Binary Tree

1.Sebuah tree yang kosong juga merupakan sebuah binary tree

2.Binary tree harus memenuhi salah satu syarat berikut:

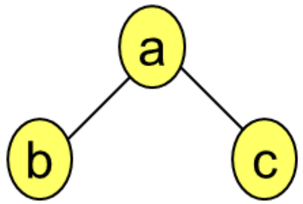
- Tidak memiliki anak
- Memiliki subtree di sebelah kiri (left subtree)
- Memiliki subtree di sebelah kanan (right subtree)
- Memiliki baik left subtree maupun right subtree

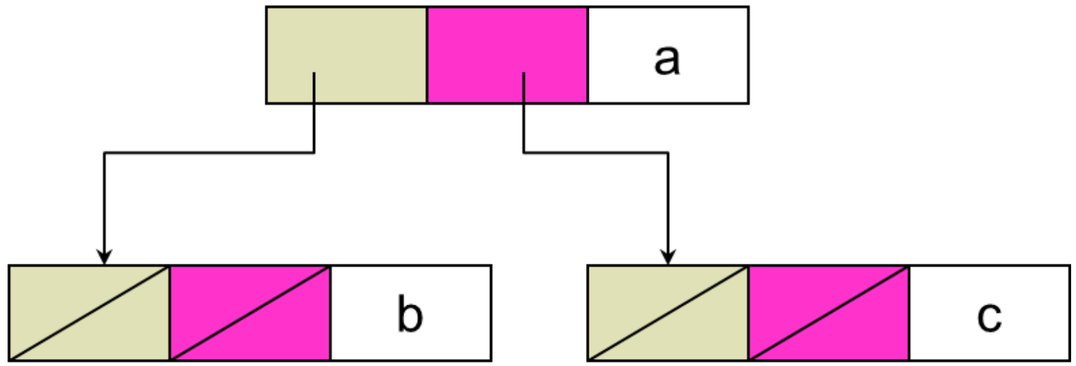


Subtree (child) yang dimiliki bukan kiri maupun kanan

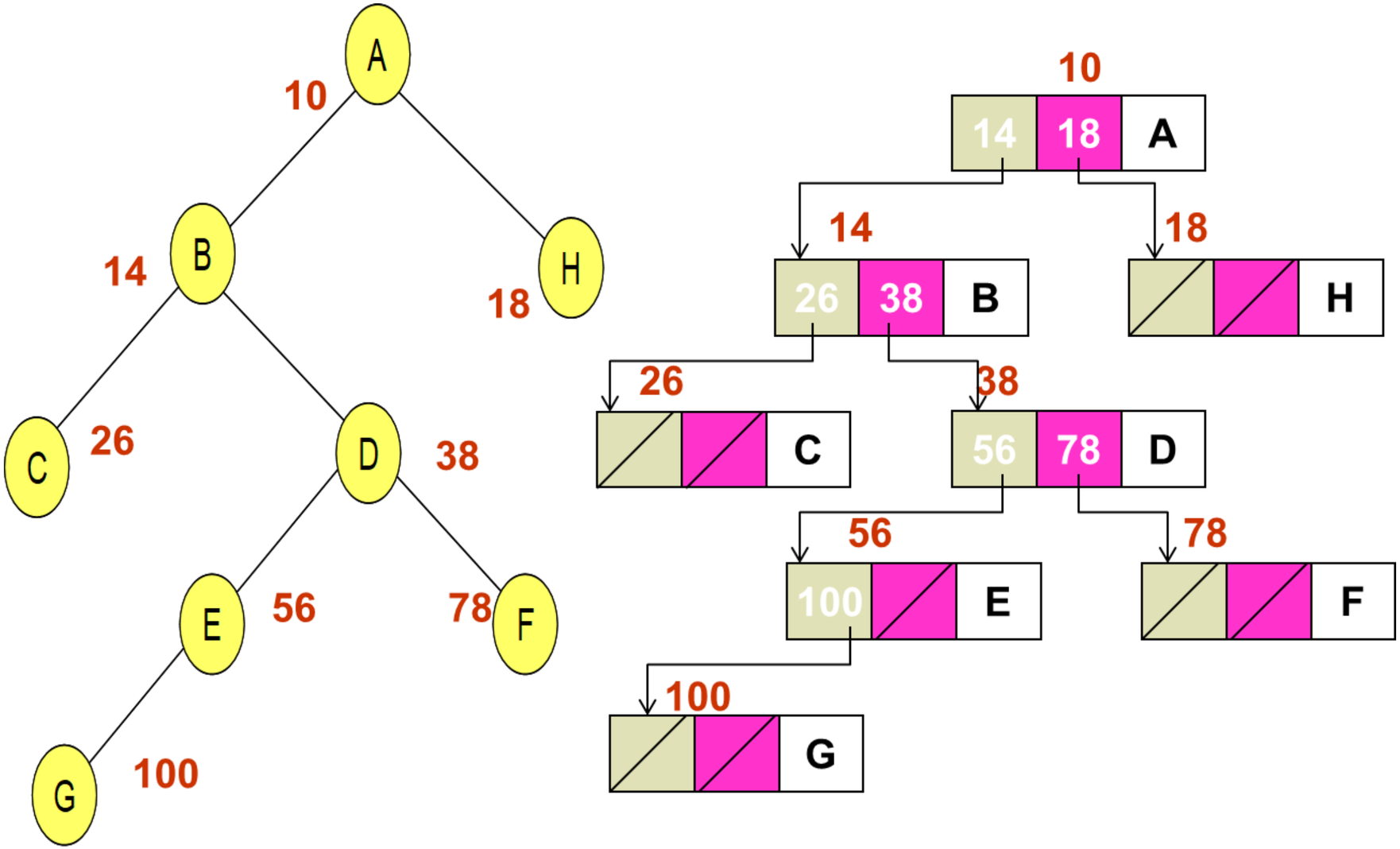
IMPLEMENTASI BINARY TREE

```
struct node {
        struct node*left;
        struct node*right;
        mydata label;
}
```





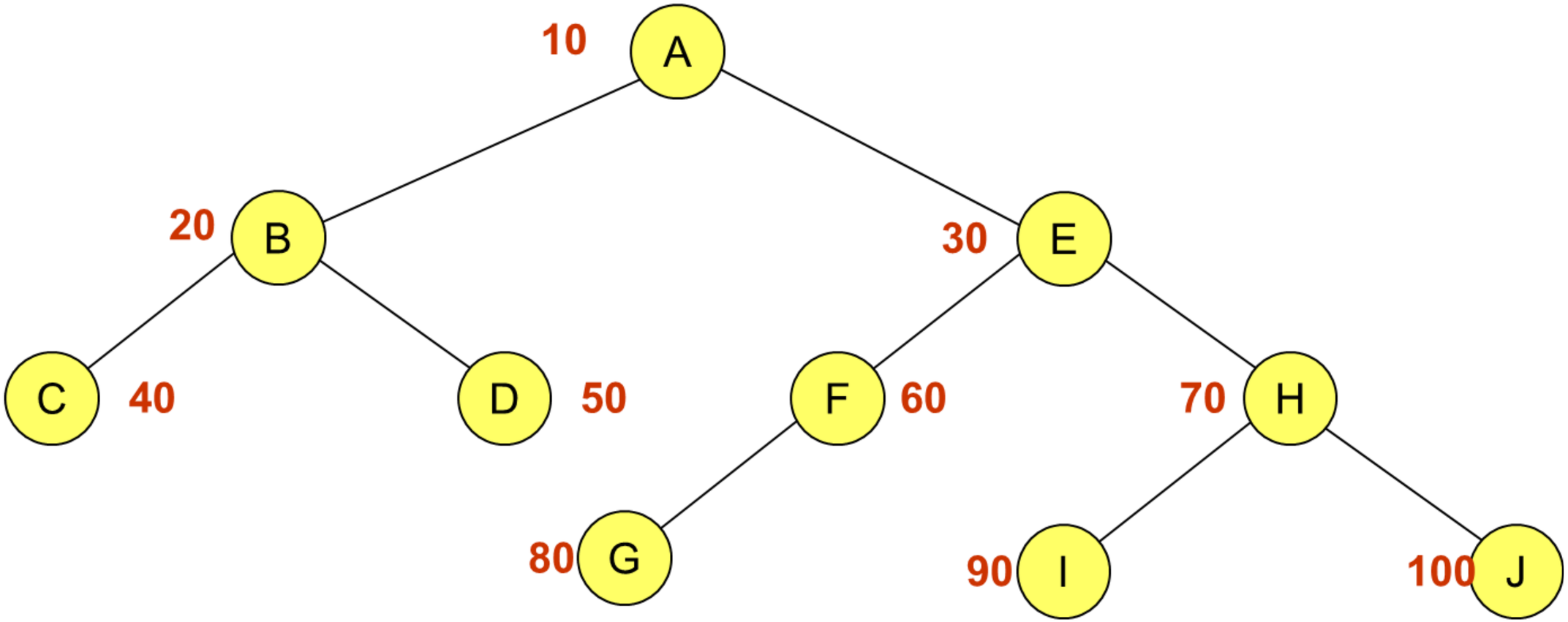
CONTOH



A, B, C, D, E, F, G adalah Label
10, 14, 18, 26, 38,56, 78, 100 adalah alamat memori

TUGAS 1

Gambarkan implementasi dari binary tree berikut



TREE TRAVERSAL

TUGAS

- Teknik menyusuri tiap node dalam sebuah tree secara sistematis, sehingga semua node dapat dan hanya satu kali saja dikunjungi

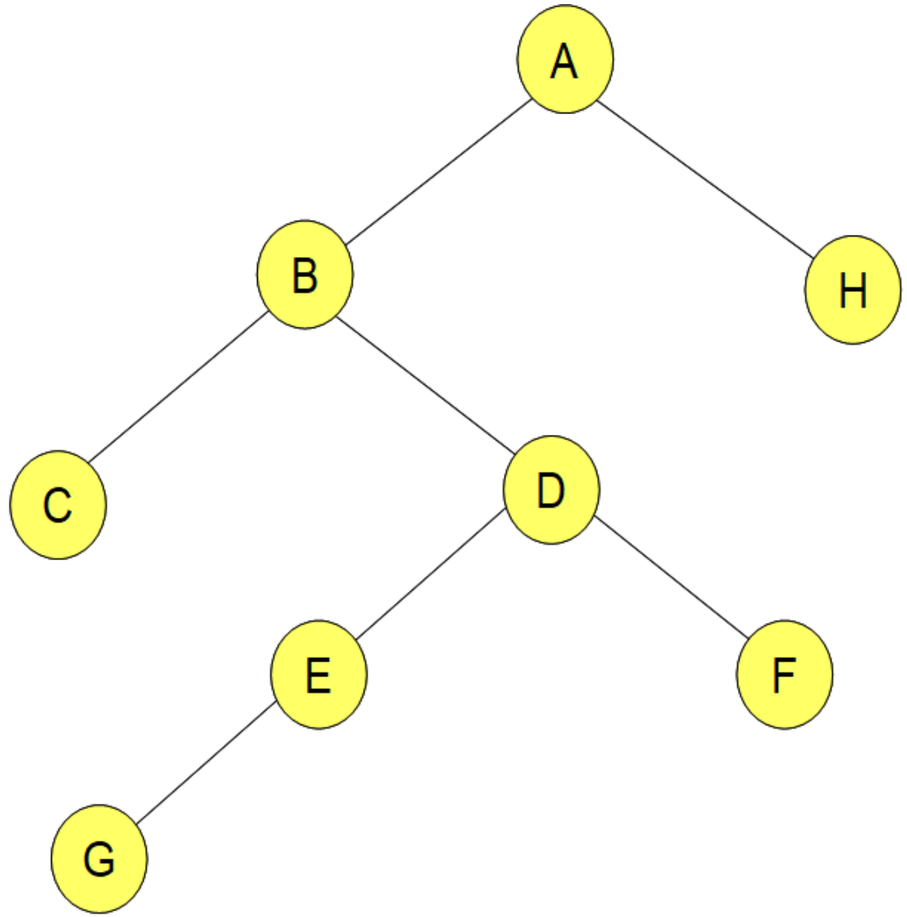
- Ada tigacara traversal

1. preorder
2. inorder
3. postorder

- Untuk tree yang kosong, traversal tidak perlu dilakukan

PREORDER

1. Visit the root
2. Traverse the left subtree
3. Traverse the right subtree



A→B→C→D→E→G→F→H

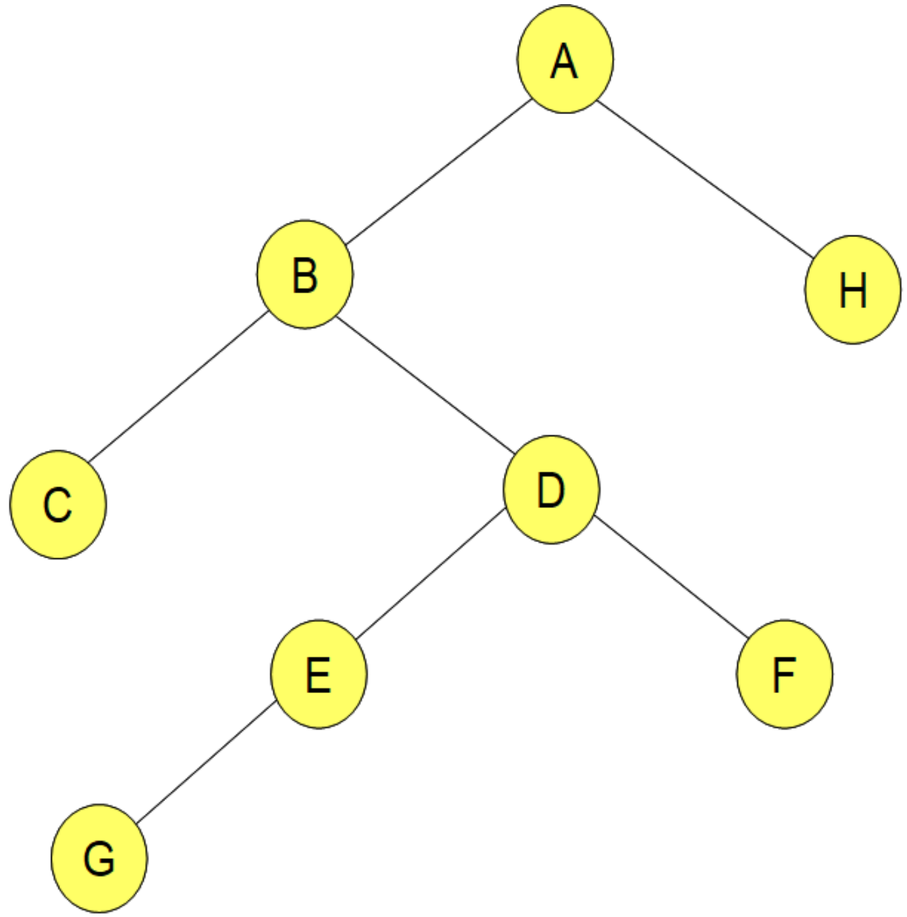
IMPLEMENTASI DALAM BAHASA C++

```
struct Node {
        Node* left;
        Node* right;
        char label;
};

void preorder(Node* p)
{
        if (p == NULL) return; // jika tree kosong, tidak perlu lakukan apa-apa
        std::cout << "visit " << p->label << " "; // tampilkan label node yang dikunjungi
        preorder(p->left); // traverse the left subtree
        preorder(p->right); // traverse the right subtree
}
```

INORDER

1. Traverse the left subtree
2. Visit the root
3. Traverse the right subtree



C→B→G→E→D→F→A→H

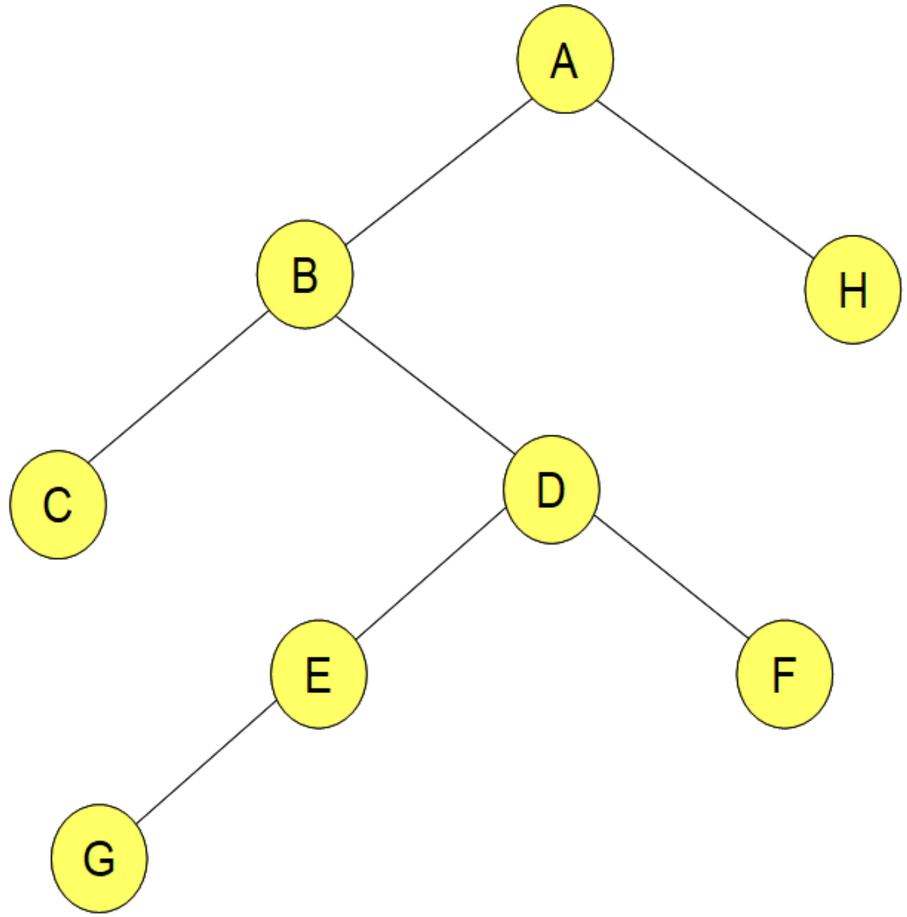
IMPLEMENTASI DALAM BAHASA C++

```
struct Node {
        Node* left;
        Node* right;
        char label;
};

void inorder(Node* p)
{
        if (p == NULL) return; // jika tree kosong, tidak perlu lakukan apa-apa
        inorder(p->left); // traverse the left subtree
        std::cout << "visit " << p->label << " "; // tampilkan label node yang dikunjungi
        inorder(p->right); // traverse the right subtree
}
```

POSTORDER

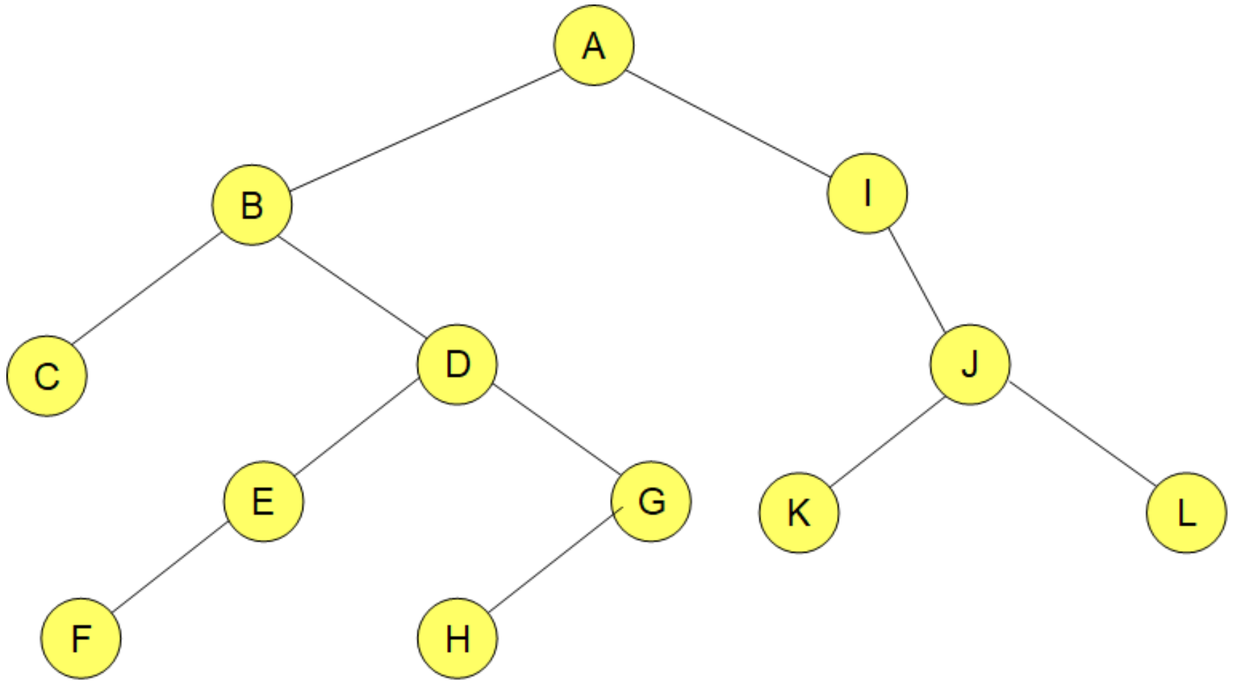
1. Traverse the left subtree
2. Traverse the right subtree
3. Visit the root



C→G→E→F→D→B→H→A

TUGAS 2

Tuliskan hasil preorder, inorder dan postorder traversal untuk binary tree berikut.



1. Print
2. Traverse the left subtree
3. Traverse the right subtree

1. Traverse the left subtree
2. Print
3. Traverse the right subtree

1. Traverse the left subtree
2. Traverse the right subtree
3. Print

THANK YOU