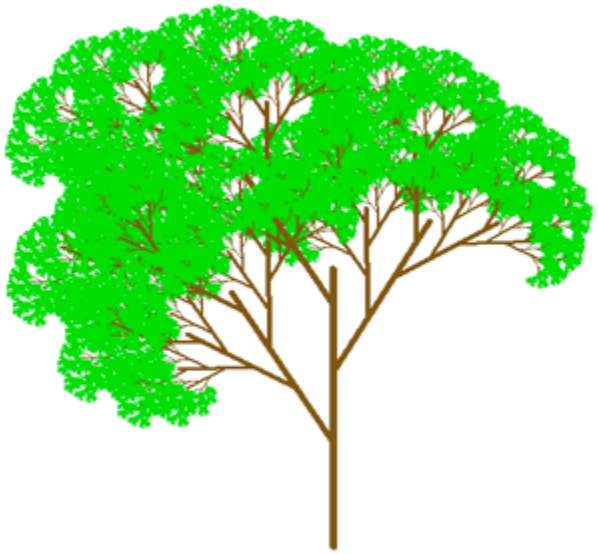
Algoritma dan Struktur Data

Tree



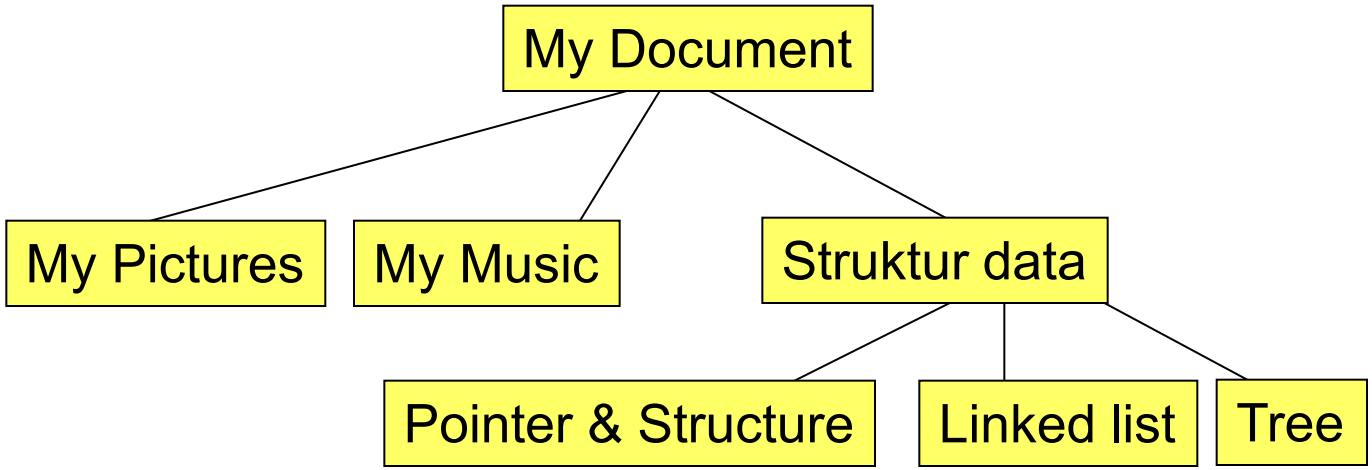
Tree

Outline

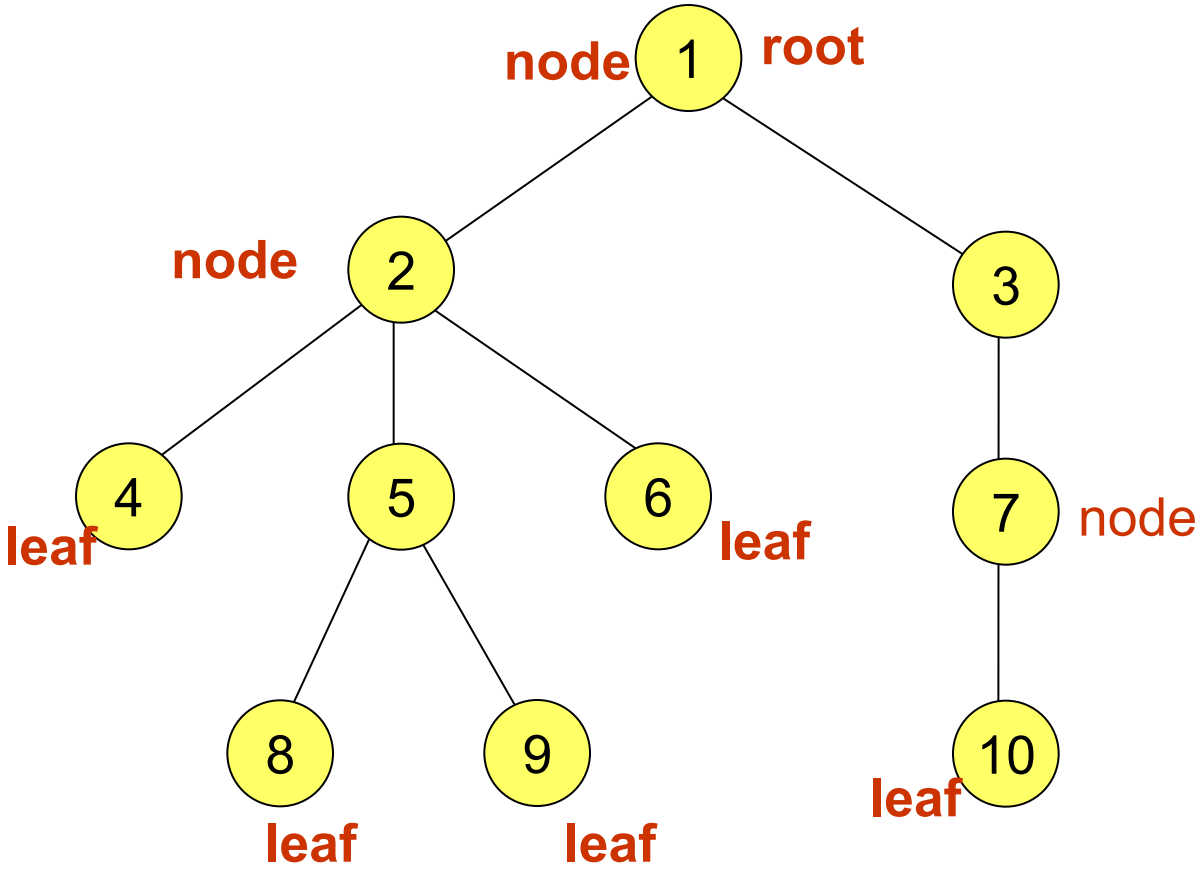
1. Apakah Tree Structure itu ?
2. Binary Tree & implementasinya
3. Tree Traversal
4. Implementasi tree (selain binary tree)

Apakah Tree Structure itu ?

- Struktur data yang menunjukkan hubungan bertingkat (memiliki hierarkhi)
- Contoh: direktori/folder pada windows atau linux

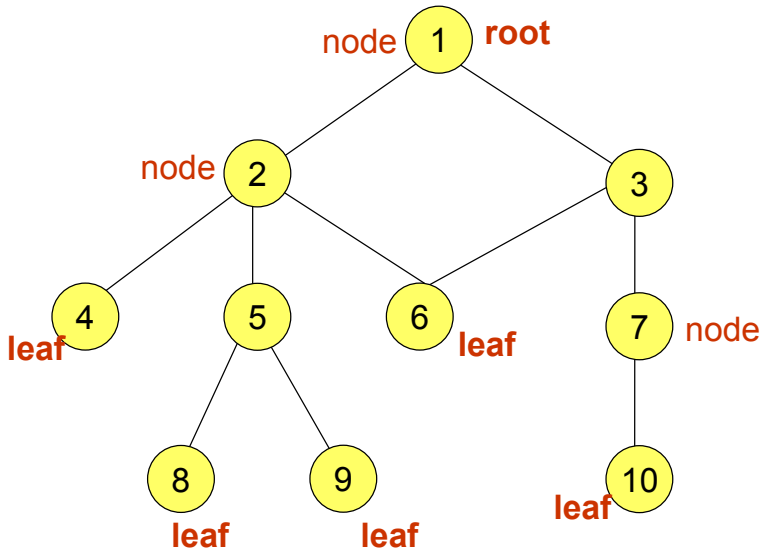


Nama komponen pada Tree



Hubungan antar komponen

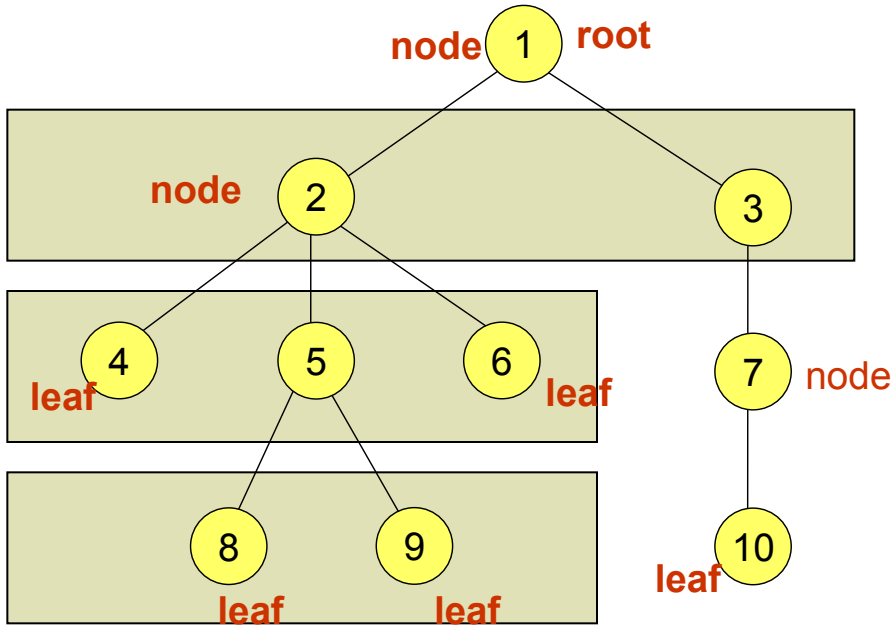
- Hubungan antar elemen: parent-child, father-son, mother-daughter
- Nama node: nama(angka) yang dipakai untuk membedakan sebuah node dengan node yang lain. Dalam kuliah ini adalah angka yang tertulis dalam lingkaran.
- Label: nilai yang diingat oleh sebuah node
- Tree vs Graph
  - Tree: setiap node kecuali root hanya memiliki sebuah parent
  - Graph: dapat memiliki lebih dari sebuah parent



Contoh graph

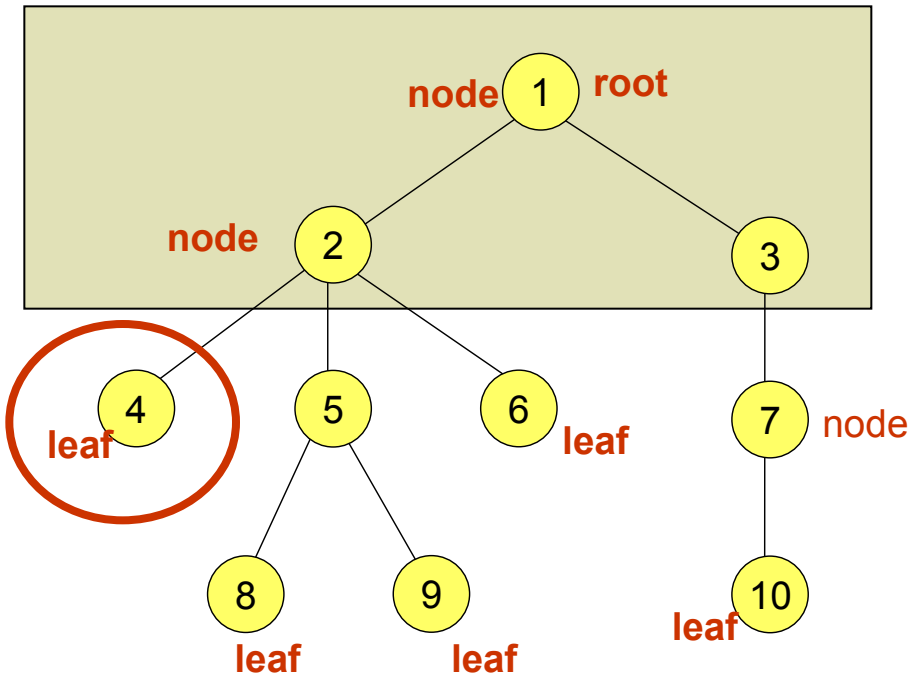
Hubungan antar komponen

- sibling : node-node yang memiliki parent yang sama
- Ancestor dari node x : node yang ditemukan, ketika menyusuri tree ke atas dari node x
- Descendant dari node x : node yang ditemukan ketika menyusuri tree ke bawah dari node x



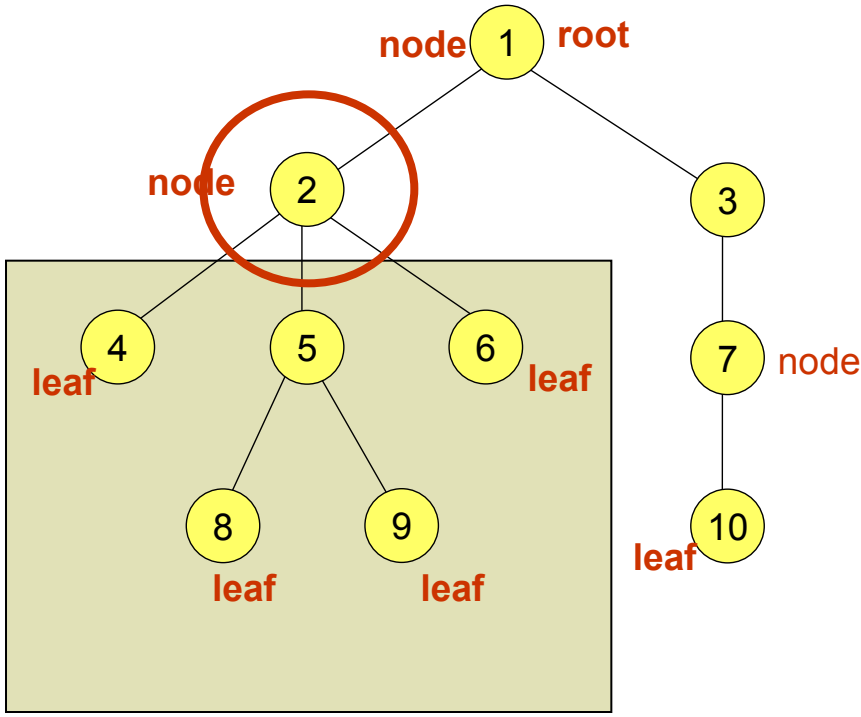
Hubungan antar komponen

- sibling : node-node yang memiliki parent yang sama
- Ancestor dari node x : node yang ditemukan, ketika menyusuri tree ke atas dari node x
- Descendant dari node x : node yang ditemukan ketika menyusuri tree ke bawah dari node x

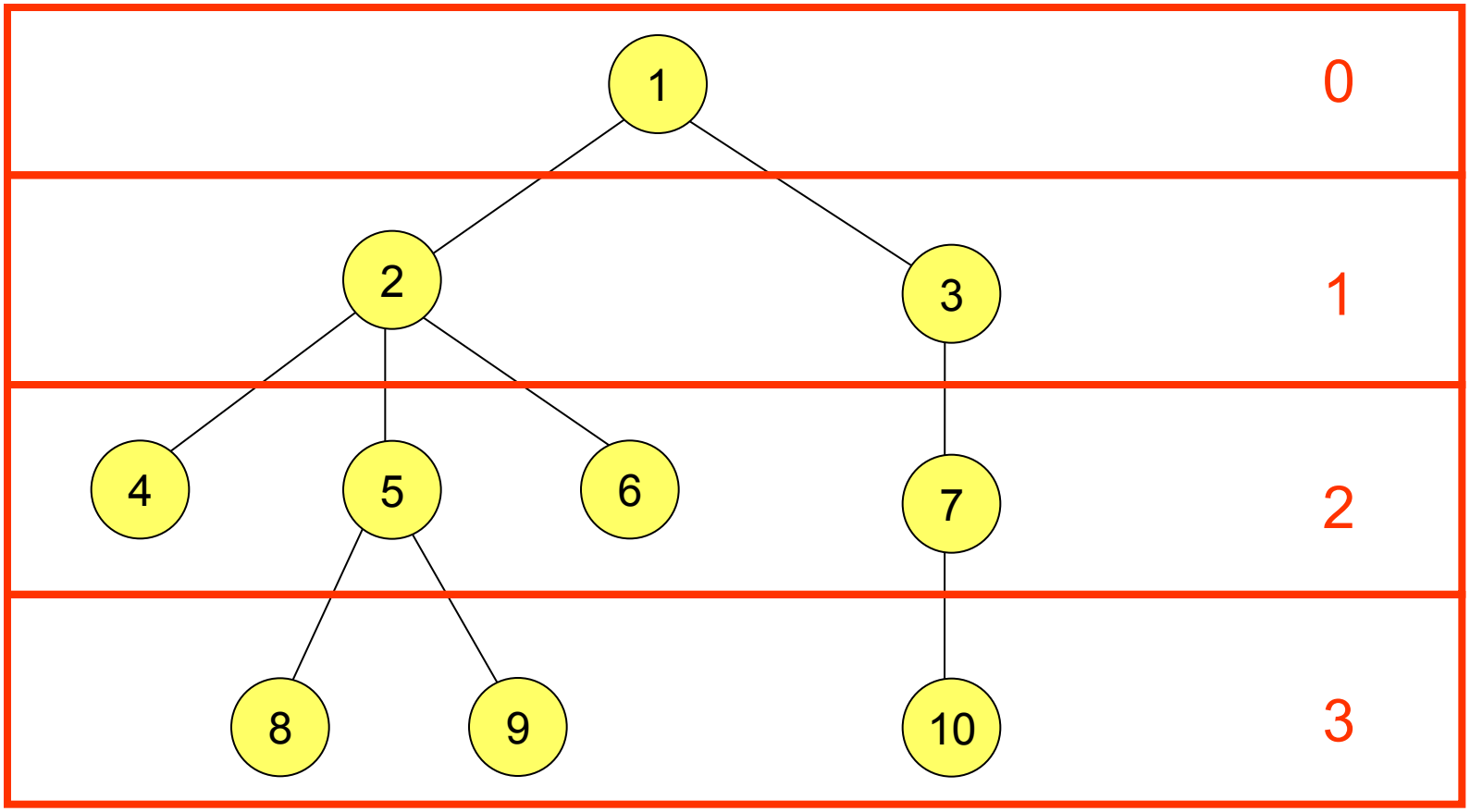


Hubungan antar komponen

- sibling : node-node yang memiliki parent yang sama
- Ancestor dari node x : node yang ditemukan, ketika menyusuri tree ke atas dari node x
- Descendant dari node x : node yang ditemukan ketika menyusuri tree ke bawah dari node x



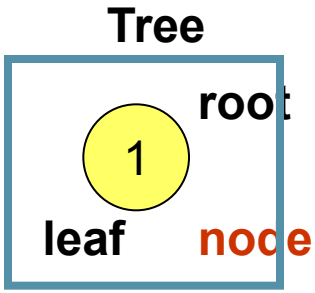
Level



Definisi TREE

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

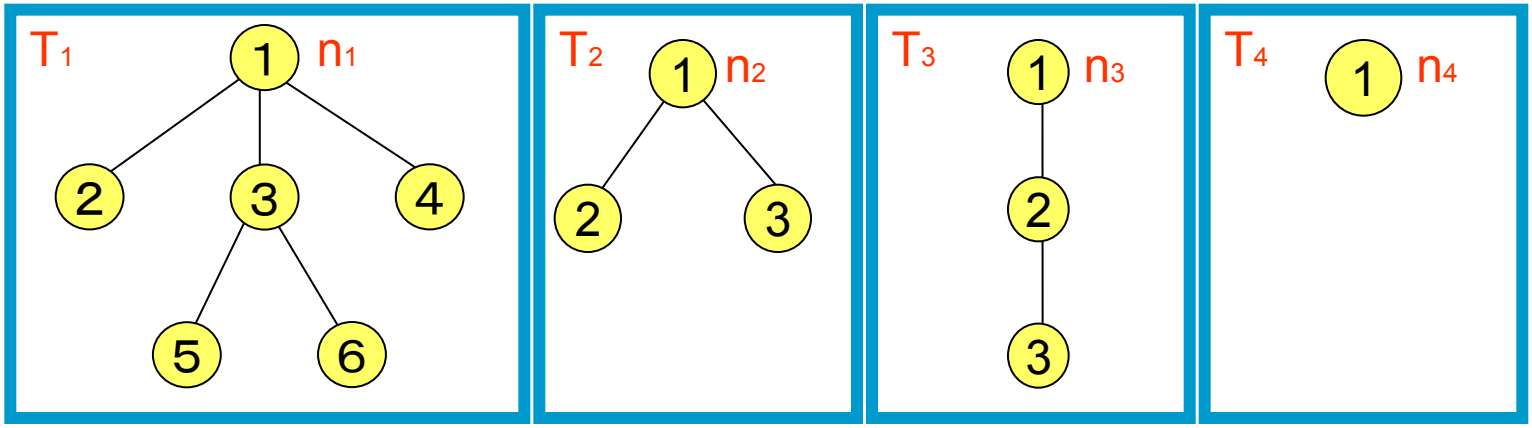
1. Sebuah node adalah sebuah tree. Node satu-satunya pada tree ini berfungsi sebagai root maupun leaf.
2. Dari k buah tree $T_1 \sim T_k$, dan masing-masing memiliki root $n_1 \sim n_k$.Jika node n adalah parent dari node$n_1 \sim n_k$, akan diperoleh sebuah tree baru T yang memiliki root n. Dalam kondisi 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 .



Definisi TREE

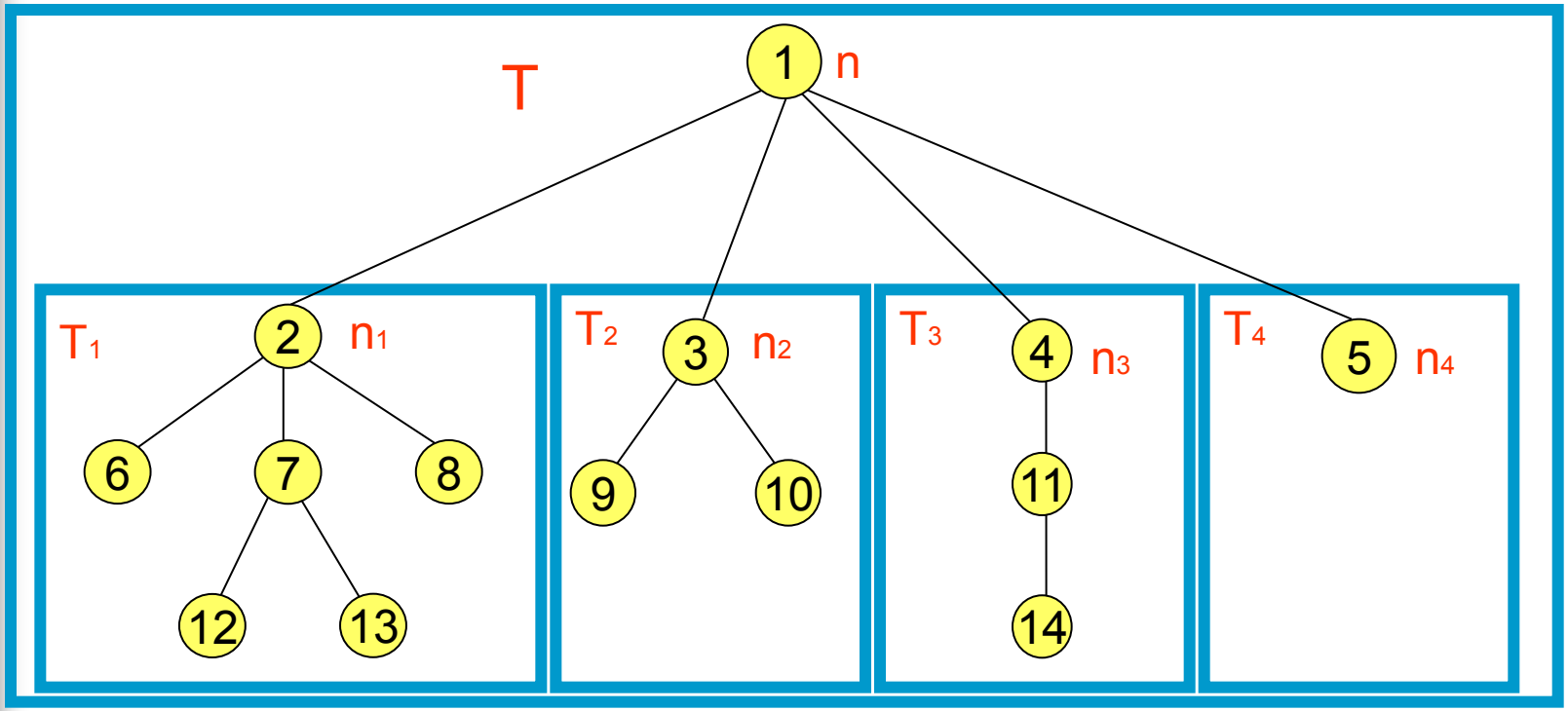
Sebuah tree didefinisikan sebagai struktur y ang dibentuk secara recursive oleh kedua rule berikut:

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



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

Outline

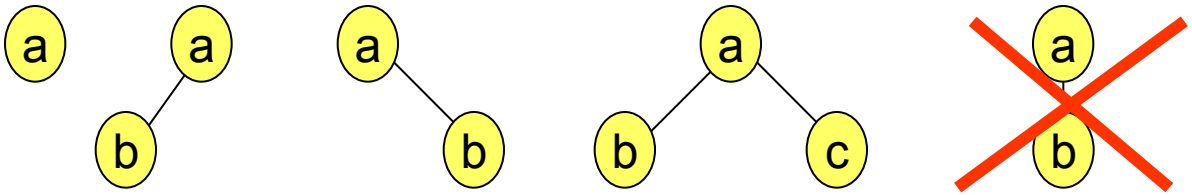
1. Apakah Tree Structure itu ?
2. Binary Tree & implementasinya
3. Tree Traversal
4. Implementasi tree (selain binary tree)

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

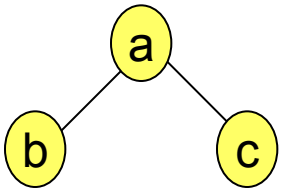
HATI-HATI DALAM MENGGAMBAR BINARY TREE

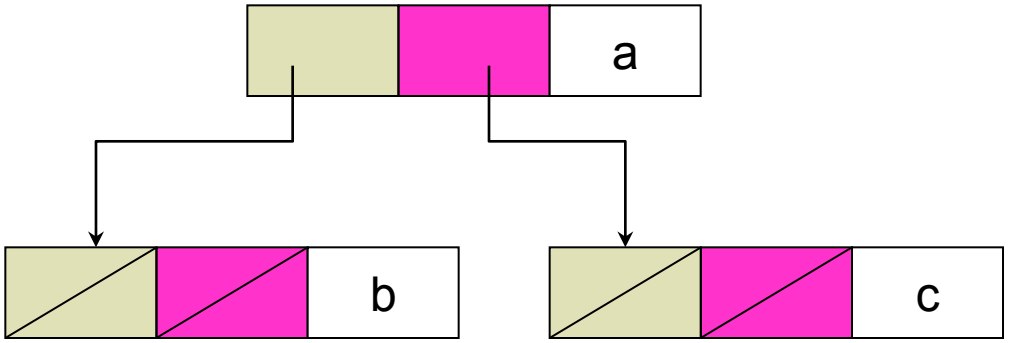


Subtree (child) yang dimiliki bukan kiri maupun kanan

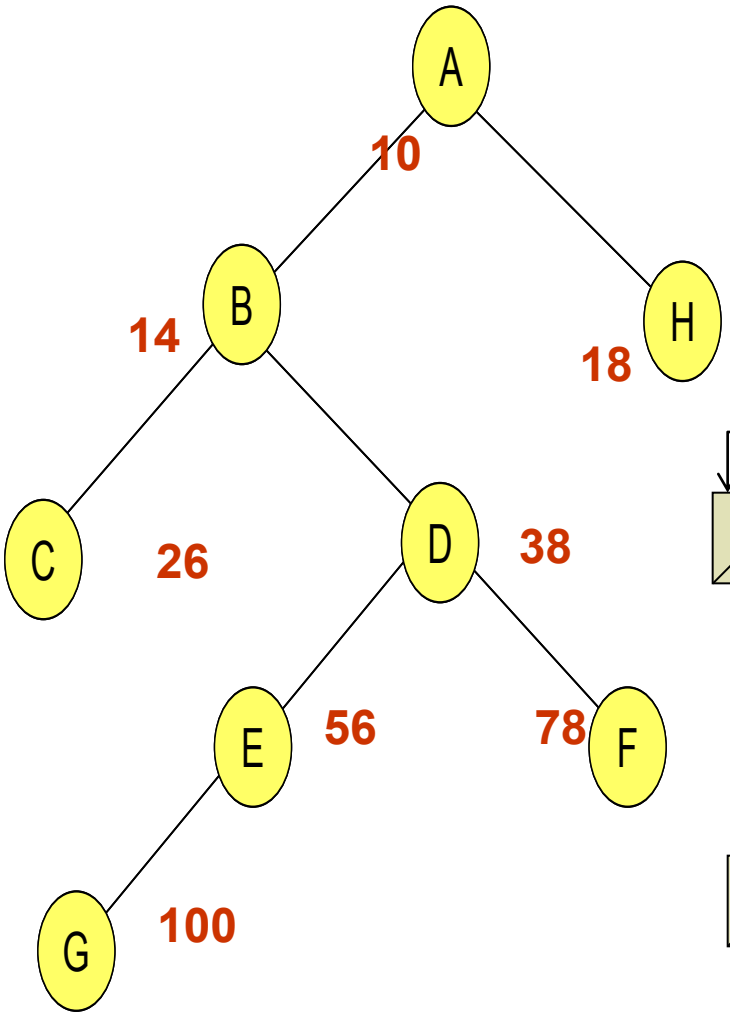
Implementasi Binary Tree

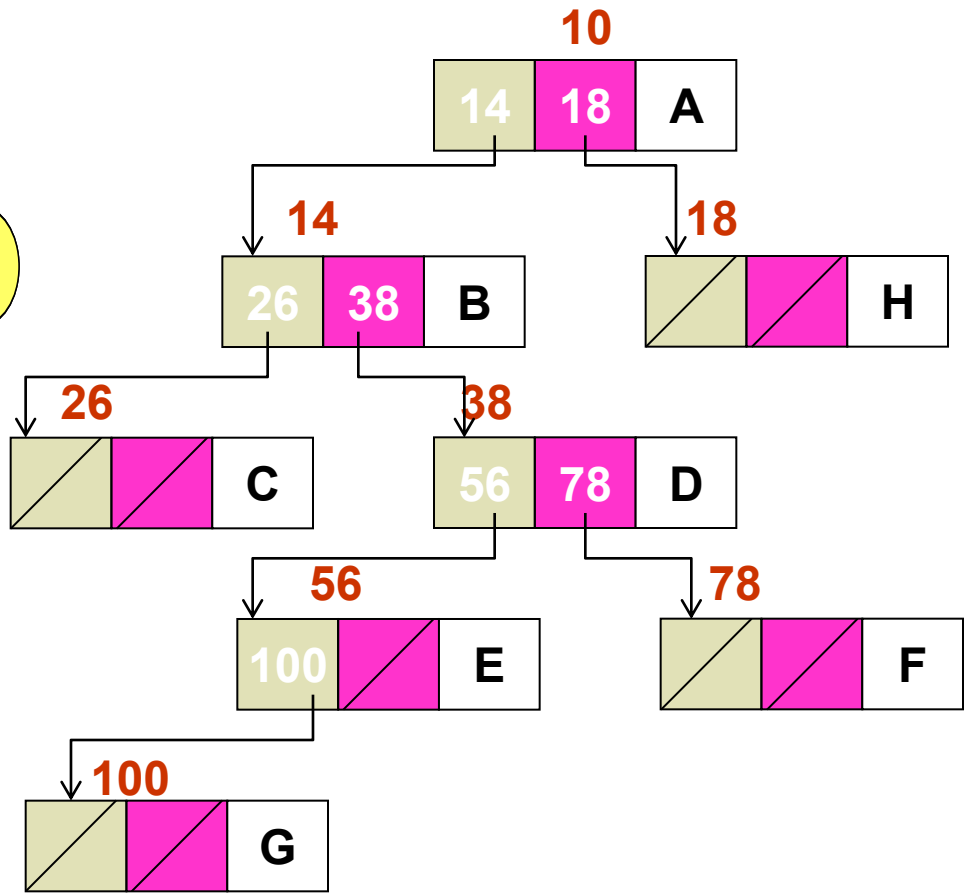
```
struct node  {
        struct node  *left;
        struct node  *right;
        mydata       label;
}
```





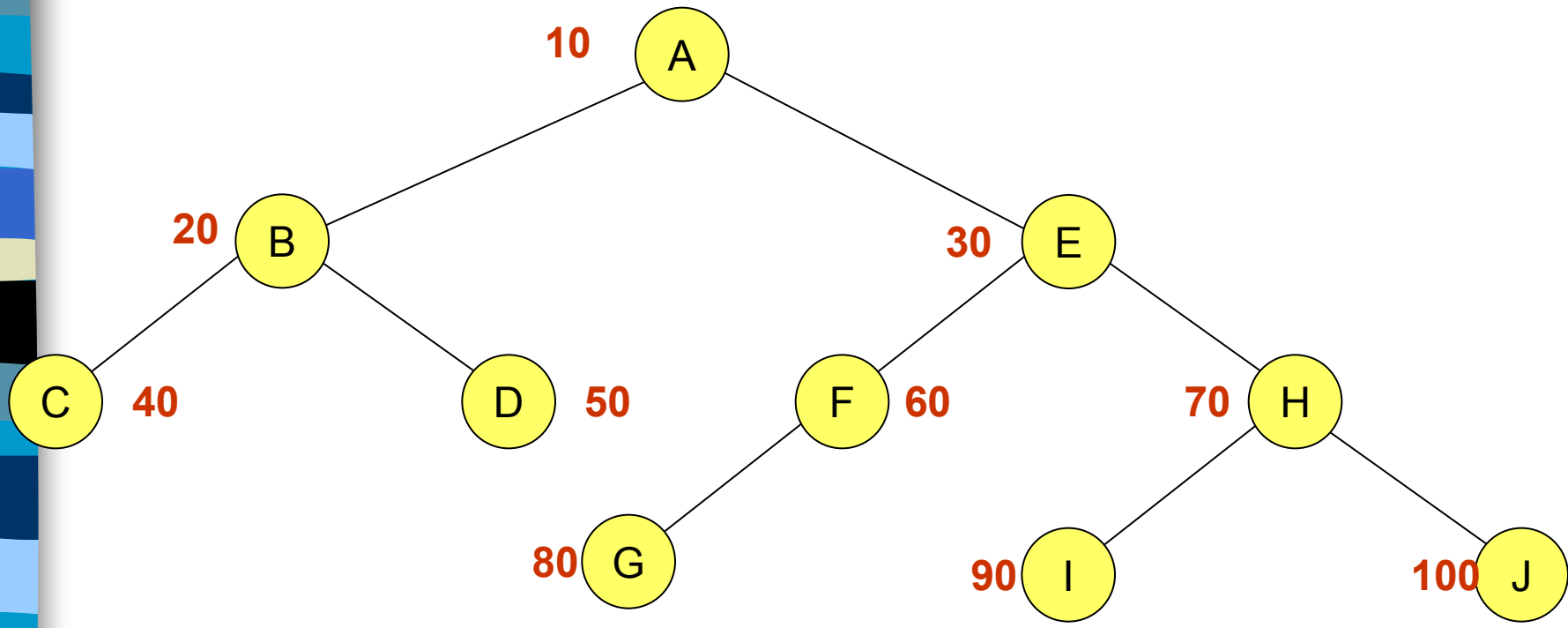
Contoh


A
10
B
14
H
18
C
26
D
38
E
56
78
F
G
100


10
14
18
A
14
18
26
38
B
H
26
38
C
56
78
D
56
78
100
E
F
100
G

Latihan 1

- Gambarkan implementasi dari binary tree berikut



Outline

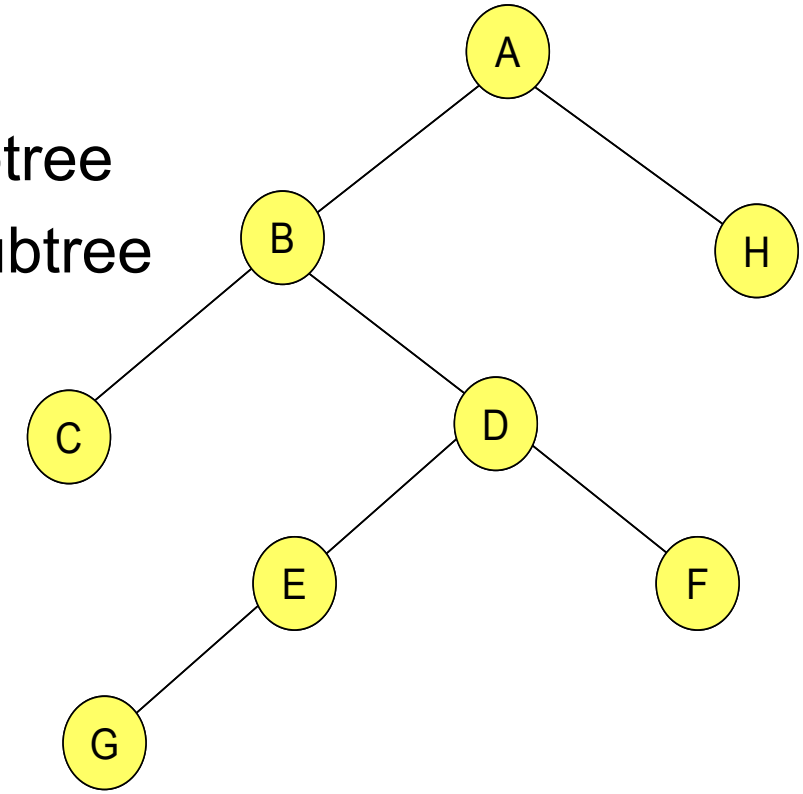
1. Apakah Tree Structure itu ?
2. Binary Tree & implementasinya
3. Tree Traversal
4. Implementasi tree (selain binary tree)

Definisi Tree Traversal

- Teknik menyusuri tiap node dalam sebuah tree secara sistematis, sehingga semua node dapat dan hanya satu kali saja dikunjungi
- Ada tiga cara traversal
  - preorder
  - inorder
  - 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

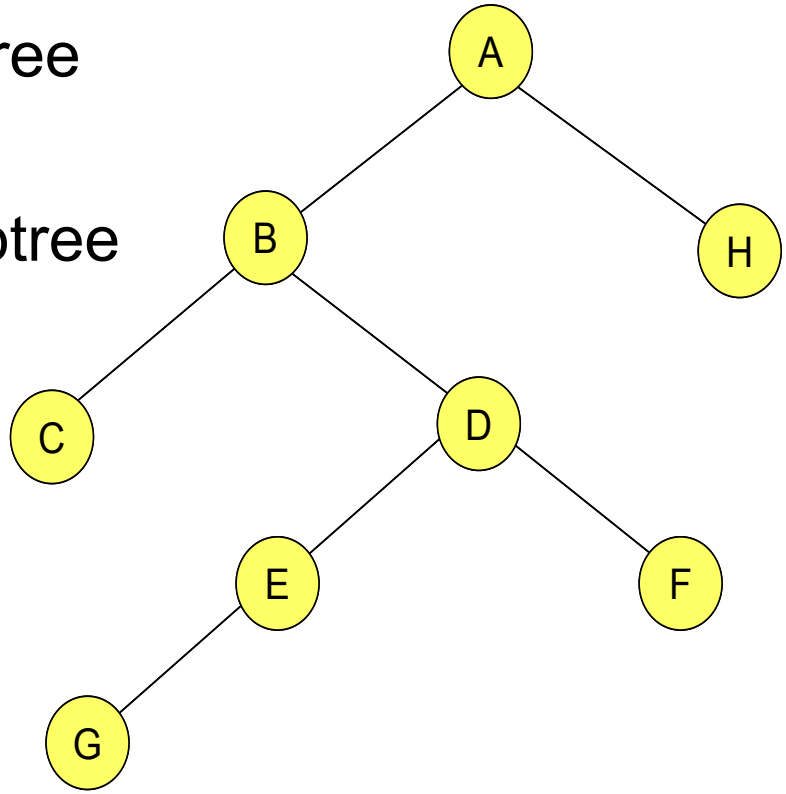
Implementasi dalam bahasa C

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

void preorder(struct node *p)
{
    if (p==NULL)  return;       jika empty-tree, tidak perlu lakukan apa-apa
    printf("visit %c ", 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

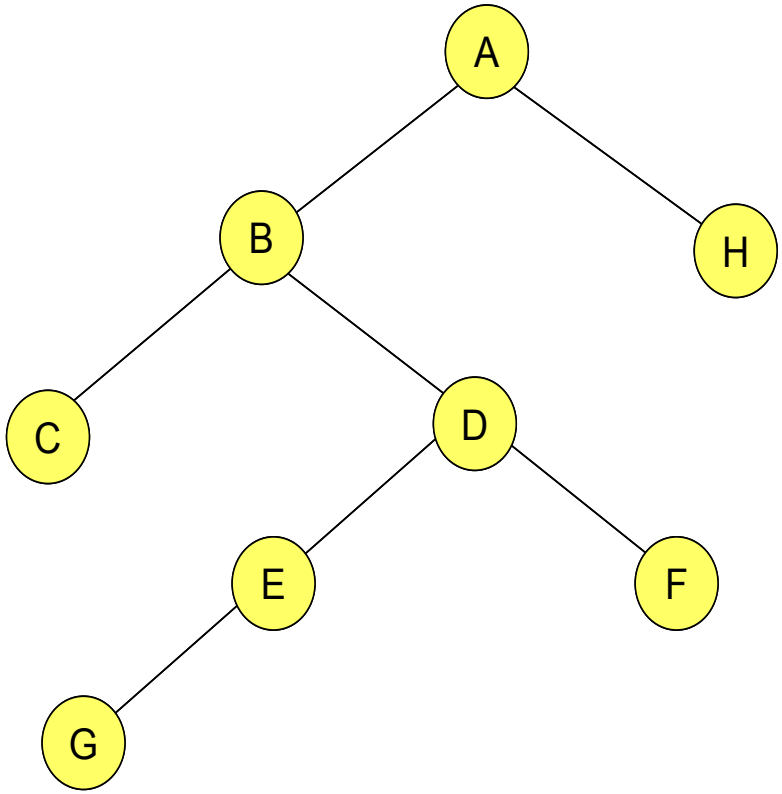
Implementasi dalam bahasa C

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

void inorder(struct node *p)
{
    if (p==NULL)  return;       jika empty-tree, tidak perlu lakukan apa-apa
    inorder(p->left);           traverse the left subtree
    printf("visit %c ", 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

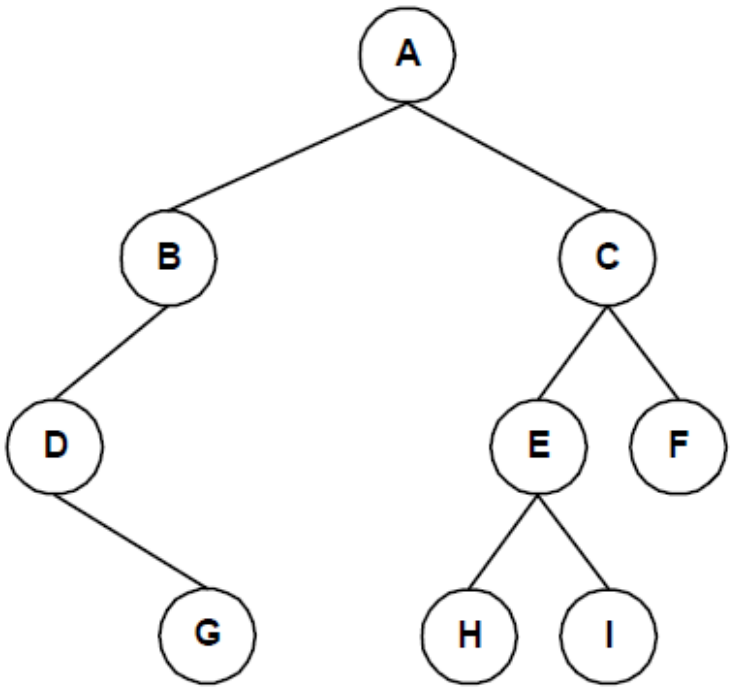


C→G→E→F→D→B→H→A

Implementasi dalam bahasa C

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

void postorder(struct node *p)
{
    if (p==NULL)  return;       jika empty-tree, tidak perlu lakukan apa-apa
    postorder(p->left);           traverse the left subtree
    postorder(p->right);          traverse the right subtree
    printf("visit %c ", p->label);  tampilkan label node yang dikunjungi
}
```



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

Secara preorder :

ABDGCEHIF

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

Secara inorder :

DGBAHEICF

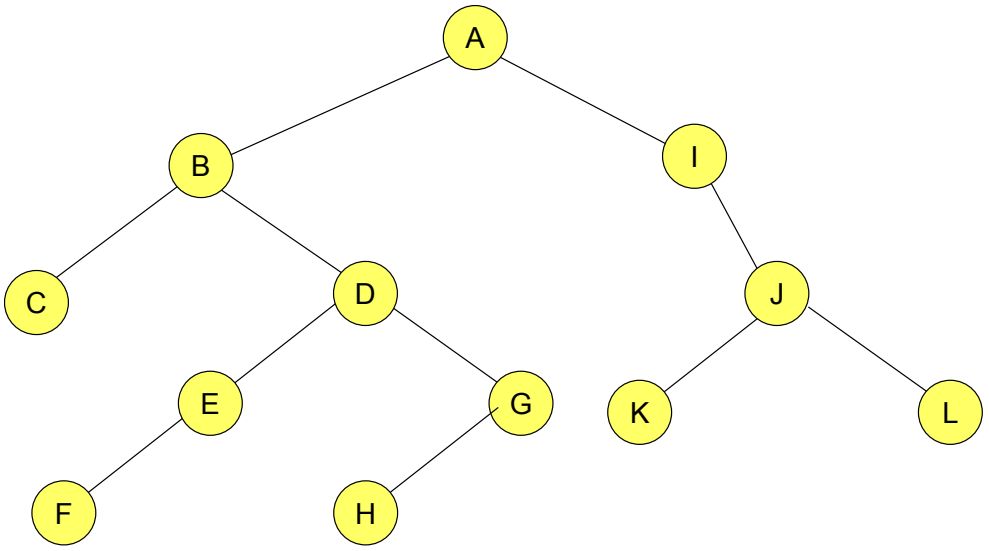
1. Traverse the left subtree
2. Traverse the right subtree
3. Print

Secara postorder :

GDBHIEFCA

Latihan-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

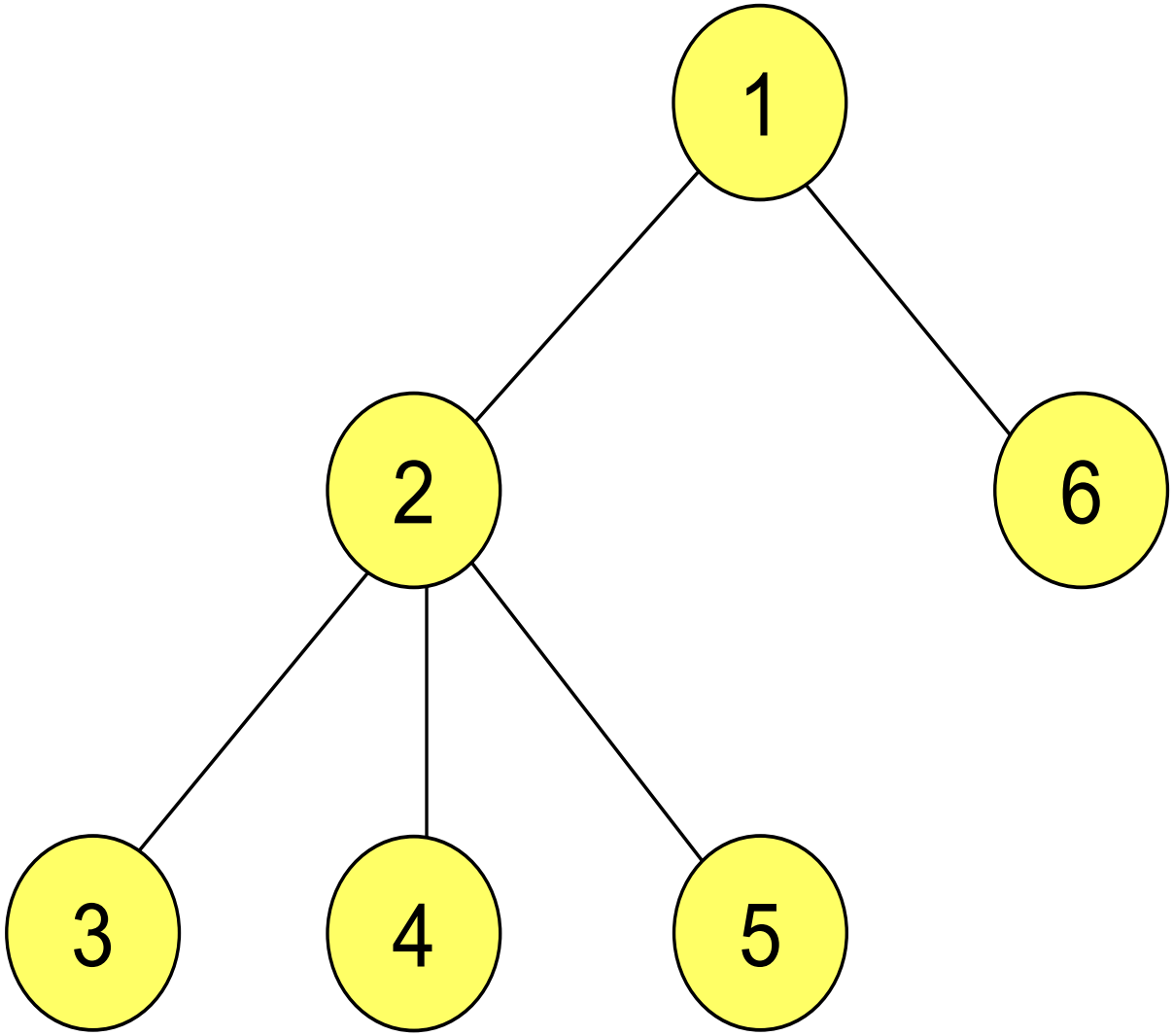
Outline

1. Apakah Tree Structure itu ?
2. Binary Tree & implementasinya
3. Tree Traversal
4. Implementasi tree (selain binary tree)

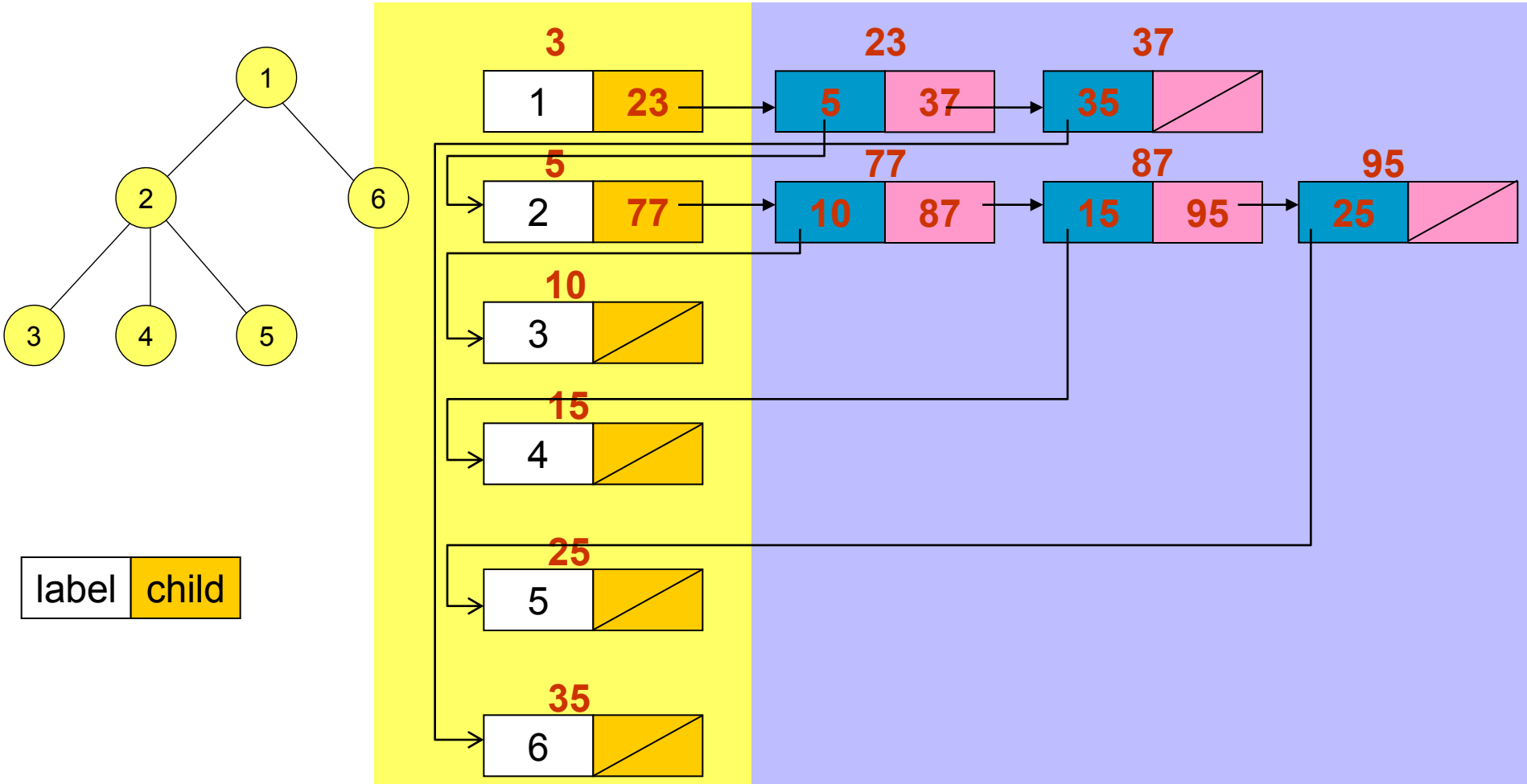
Teknik implementasi tree

- Binary tree hanya memiliki dua anak: kiri dan kanan. Karena itu implementasinya hanya memerlukan dua buah pointer untuk masing-masing subtree.
- Untuk implementasi tree yang memiliki sebarang anak, dapat dilakukan dengan dua cara
  - memakai linked-list
  - memakai binary tree

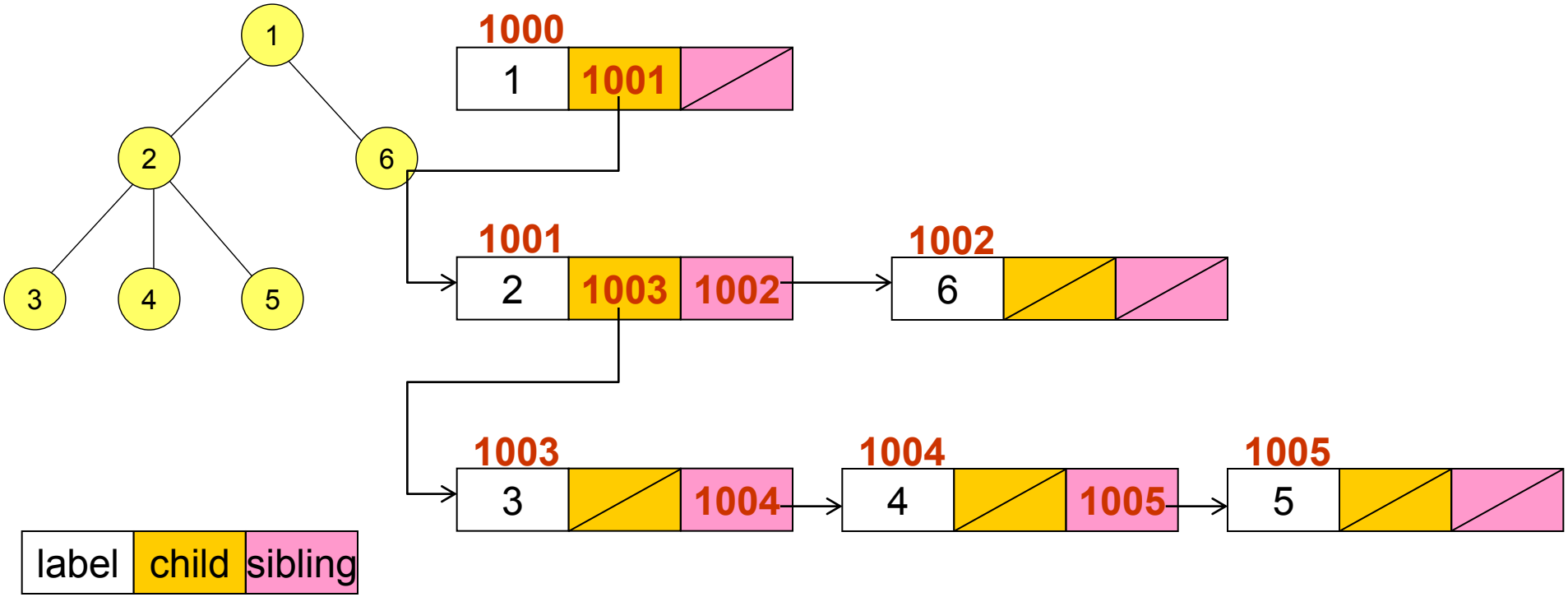
Contoh



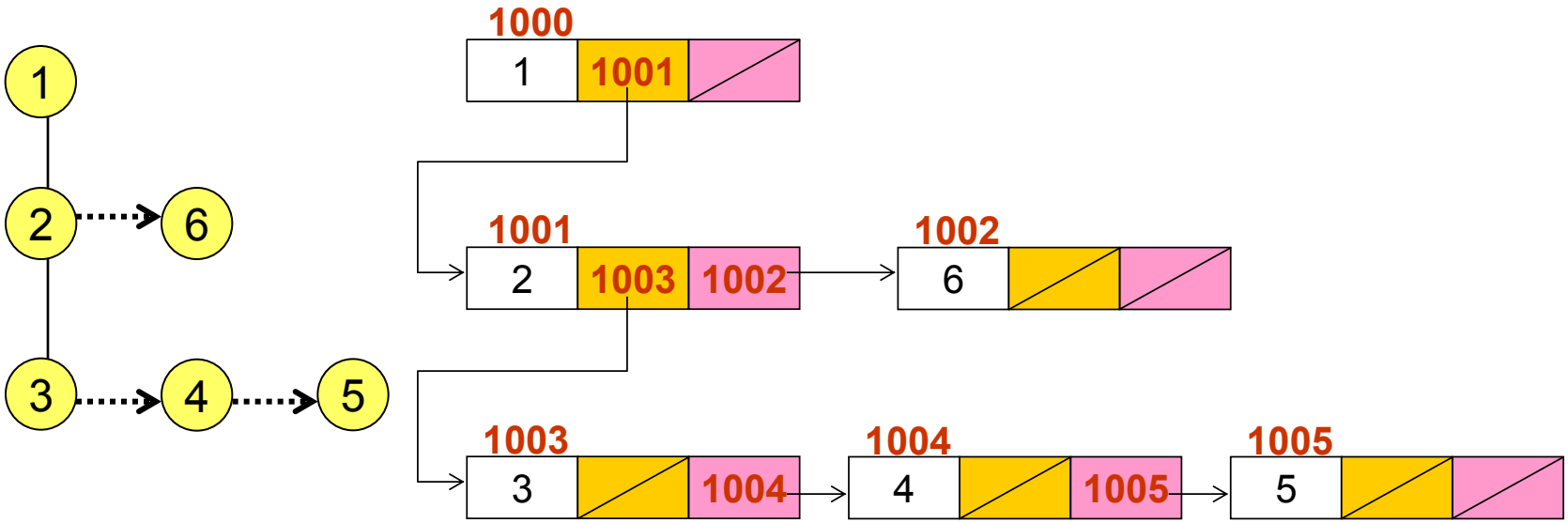
Implementasi memakai linked-list



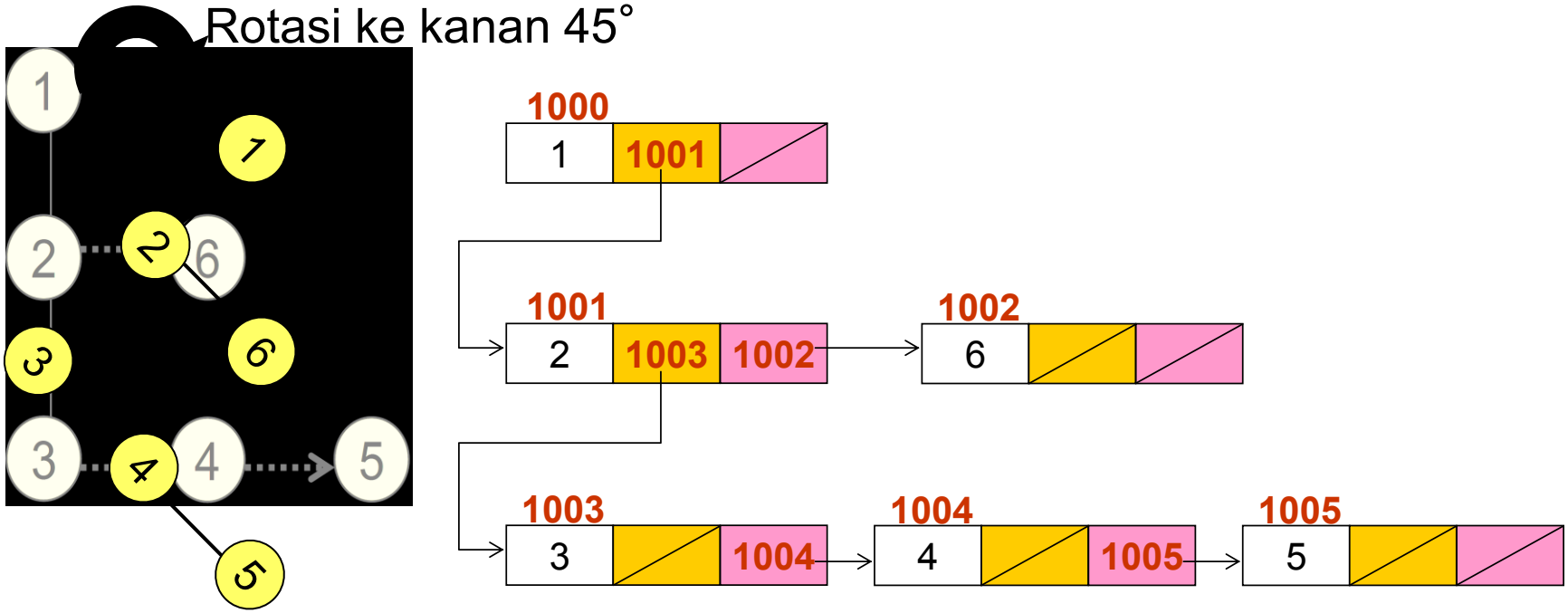
Implementasi memakai binary-tree



Implementasi memakai binary-tree



Implementasi memakai binary-tree



Latihan-3

Jelaskan implementasi tree berikut

a) memakai linked-list

b) memakai binary tree