FTI

TEKNIK ELEKTRO

EIE5023

# ALGORITMA DAN STRUKTUR DATA + PRAK

6


ENDANG SRI RAHAYU

# Outlines:

## Tree:

- Pengertian, istilah-istilah,
- Bentuk-bentuk Tree,
- Completed Binary Tree,
- Full Binary Tree.
- Penelusuran binary tree (preorder, inorder, postorder, level order)

Endang Sri Rahayu

# ALGORITMA DAN STRUKTUR DATA

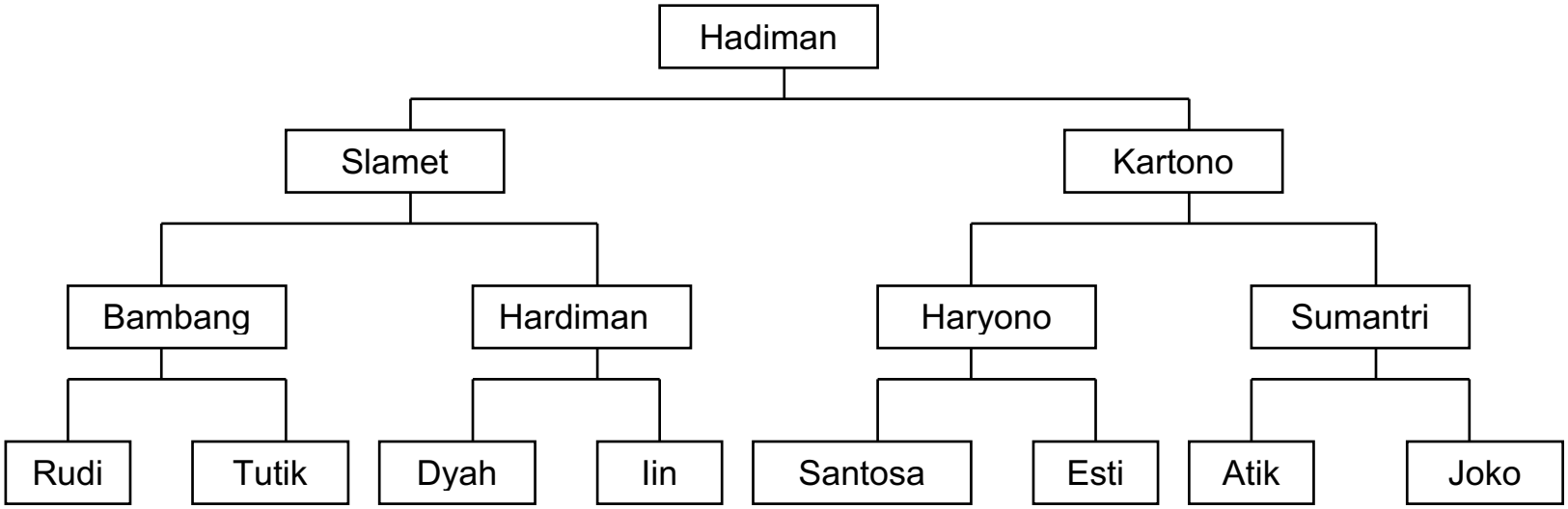
# *Tree*

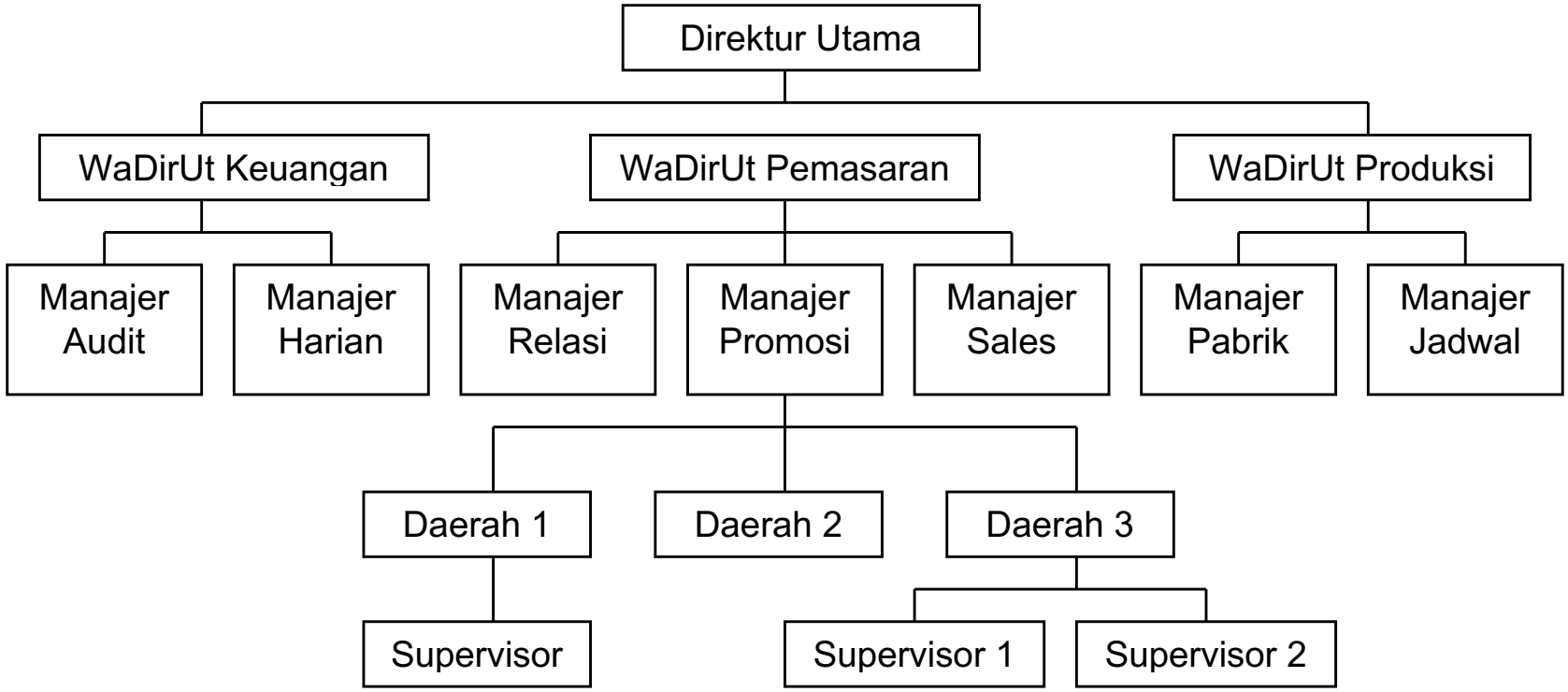
- salah satu struktur data terpenting pada *computer science*.
- dapat ditampilkan dalam banyak bentuk dan menyediakan representasi natural untuk banyak macam data yang muncul dalam aplikasi sehingga sangat berguna untuk menyelesaikan bermacan-macam masalah algoritma

⇒ Salah satu bentuk struktur data tak linear. (Contoh struktur data linear : *linked list*, *stack, queue)*

- Biasanya digunakan untuk menggambarkan susunan hirarkis antara elemen-elemen yang ada.

# Contoh Tree


Hadiman
Slamet
Kartono
Bambang
Hardiman
Haryono
Sumantri
Rudi
Tutik
Dyah
Iin
Santosa
Esti
Atik
Joko


Direktur Utama
WaDirUt Keuangan
WaDirUt Pemasaran
WaDirUt Produksi
Manajer Audit
Manajer Harian
Manajer Relasi
Manajer Promosi
Manajer Sales
Manajer Pabrik
Manajer Jadwal
Daerah 1
Daerah 2
Daerah 3
Supervisor
Supervisor 1
Supervisor 2

# Konsep Dasar dan Terminologi

- ***Tree*** adalah kumpulan elemen yang salah satu elemennya disebut ***root*** dan sisa elemen yang lain disebut *node (vertex)*, terpecah menjadi sejumlah himpunan yang saling tidak berhubungan yang disebut *subtree*.
- Diagram Tree membentuk ***node*** dan segmen garis yang disebut ***edge*** (sisi) atau ***branch*** (cabang).
- Penggambaran tree berdasarkan 3 sumber terminologi :

1. Family relationship (contoh : *parent* atau *children*)
2. Geometric relationship (contoh : *left* atau *right*)
3. Nama-nama biologi (contoh : *root* atau *leave*)

***Subtree*** : Bagian dari tree yang berupa suatu node beserta descendantnya dan memiliki karakteristik dari tree tersebut.

***Size*** : Banyaknya node dalam suatu tree

Size = 15

***Node*** adalah elemen yang berisi informasi/data dan penunjuk percabangan.

***Root*** : Satu-satunya node khusus dalam tree yang tak punyak predecessor

Contoh : Root = A

***Leaf*** : Node-node dalam tree yang tak memiliki successor

***Degree*** dari suatu *node* dinyatakan sebagai banyaknya *subtree* atau turunan (*child*) dari node tsb.

Contoh : node A memiliki *degree* 2, node B memiliki *degree* 2, node C memiliki *degree* 3, *node* F, G, I, J, K, L, N, O memiliki *degree* 0 dan disebut *leaf* atau *external node*

***Height*** atau ***Depth*** dari suatu *Tree* adalah banyaknya tingkatan/level dalam suatu tree. Jadi tree diatas memiliki *Height* atau *Depth* 5.

***Predecessor*** : Node yang berada di atas node tertentu

***Succesor*** : Node yang berada di bawah node tertentu

***Ancestor*** : adalah semua *node* yang terletak dalam satu jalur dengan *node* tsb dari root sampai node yang ditinjau. Contoh : *Ancestor* dari L adalah A, C, H

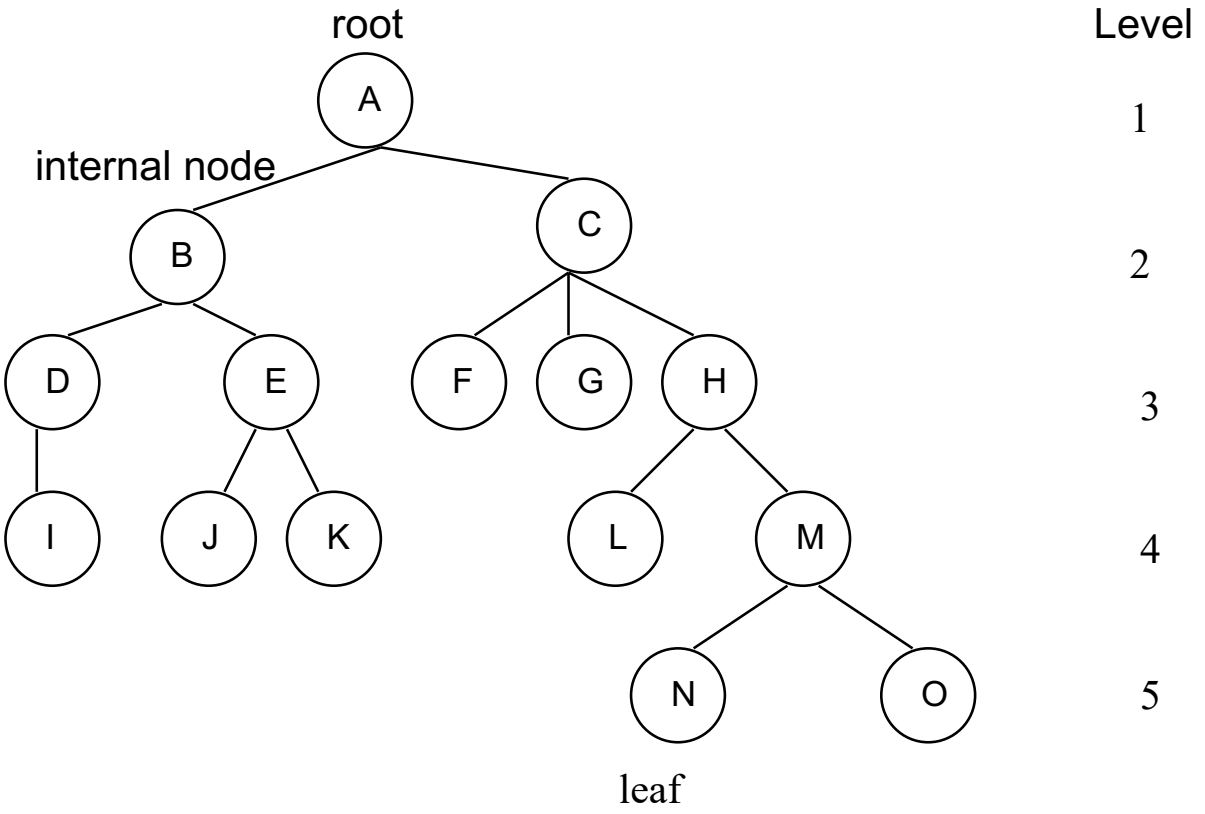
***Descendant*** : Seluruh node yang terletak sesudah node tertentu pada jalur yang sama. Contoh : Descendent(M) = N, O

***Forest*** adalah kumpulan sejumlah *Tree* yang tidak saling berhubungan.

***Parent*** : predecessor satu level di atas suatu node. Contoh : Parent(F) = C

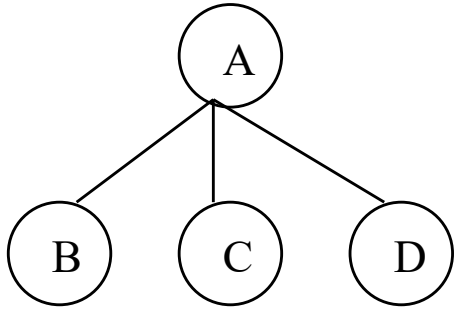
***Child*** : Successor satu level di bawah suatu node. Contoh : Child(A) = B, C

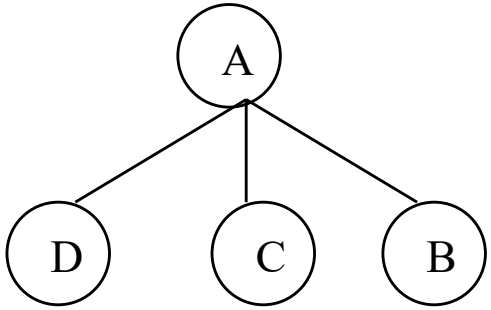
***Sibling*** : node-node yang memiliki parent yang sama. Contoh : Sibling (B) = C


root
Level
A
1
internal node
B
C
2
D
E
F
G
H
3
I
J
K
L
M
4
N
O
5
leaf

***Ordered Tree :*** adalah jenis Tree yang sangat memperhatikan urutan hubungan antara node satu dengan yang lain

Contoh 2 buah Ordered Tree



# Representasi Tree

## Cara-cara menggambarkan bentuk Tree

1. Seperti yang sudah dicontohkan (paling banyak digunakan)
2. Menggunakan notasi kurung
3. Menggunakan Diagram Venn
4. Menggunakan notasi tingkat
5. Menggunakan notasi garis

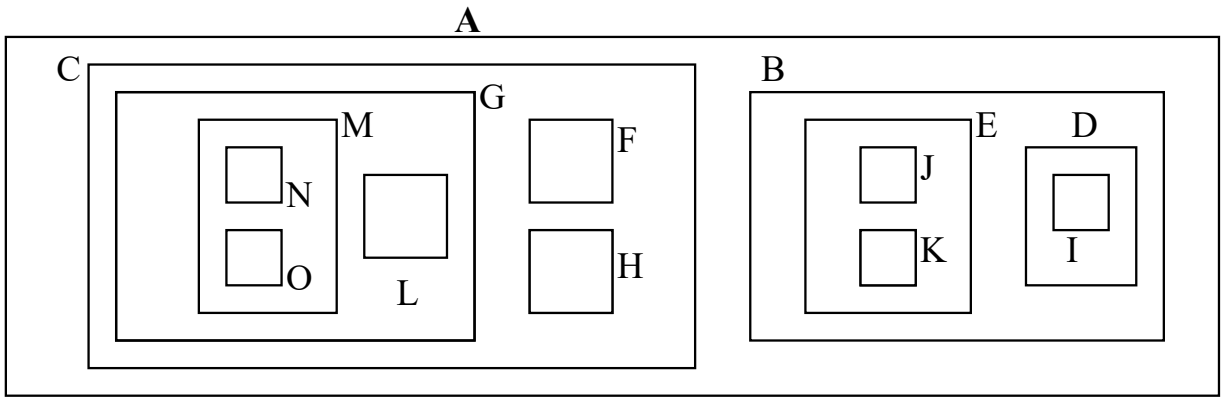
Contoh dibawah ini menggambarkan bagaimana macam-macam bentuk tree yang digunakan untuk menyelesaikan 1 persoalan yang sama

## Menggunakan notasi kurung

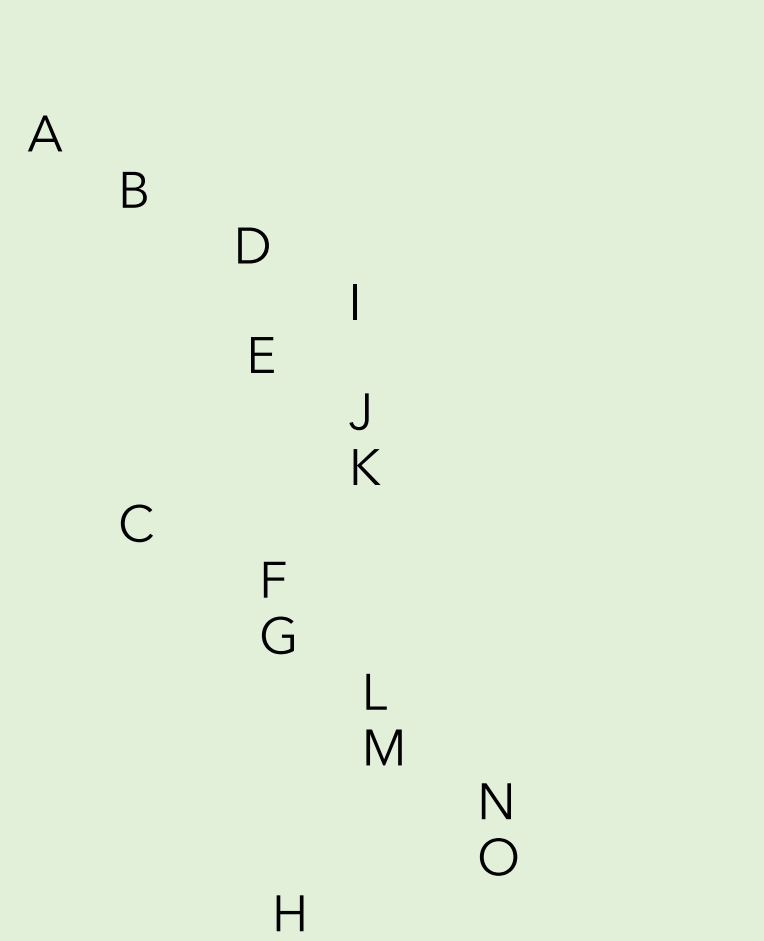
Contoh : (A(B(D(I),E(J,K)),C(F,G(L,M(N,O)),H)))

## Menggunakan Diagram Venn

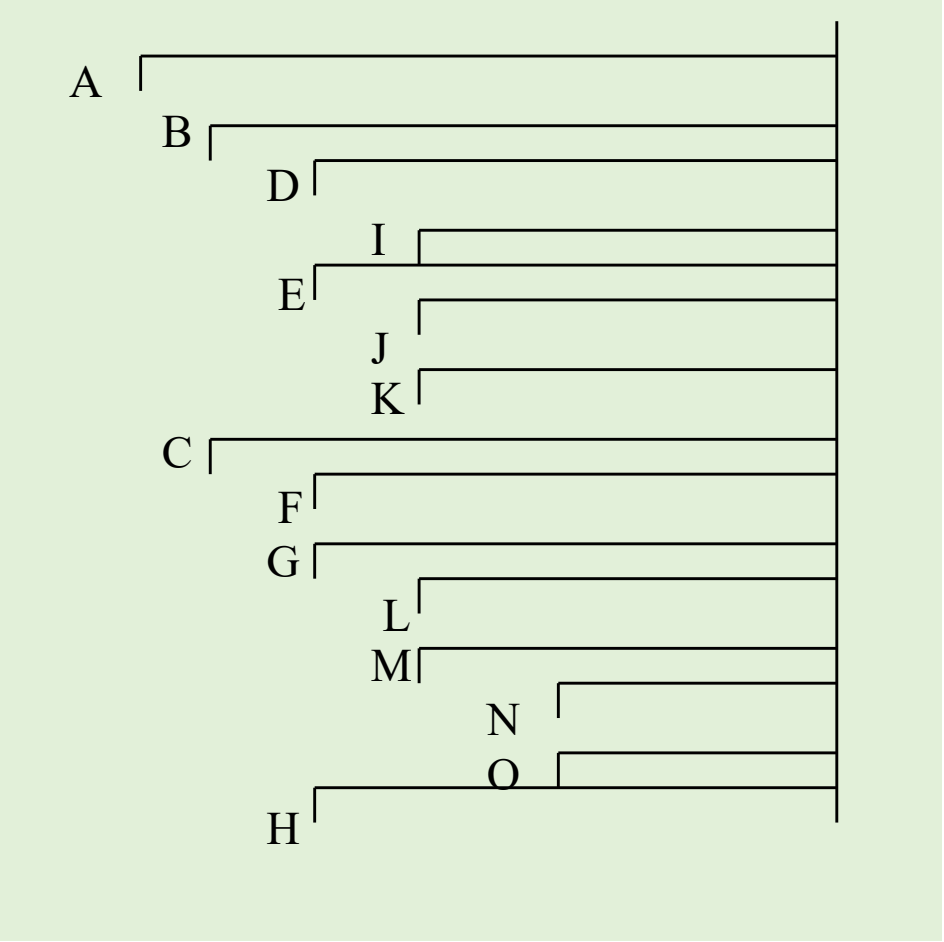
Contoh :



Menggunakan notasi tingkat




Menggunakan notasi garis



# Binary Tree

Adalah kumpulan *node* yang mungkin kosong atau mempunyai *root* dan 2 subtree yang saling terpisah (*Left Subtree* dan *Right Subtree*)

**Karakteristik :**

Setiap *node* (simpul) paling banyak memiliki 2 anak

Termasuk *ordered tree* (akan memperhatikan urutan *Left* dan *Right*)

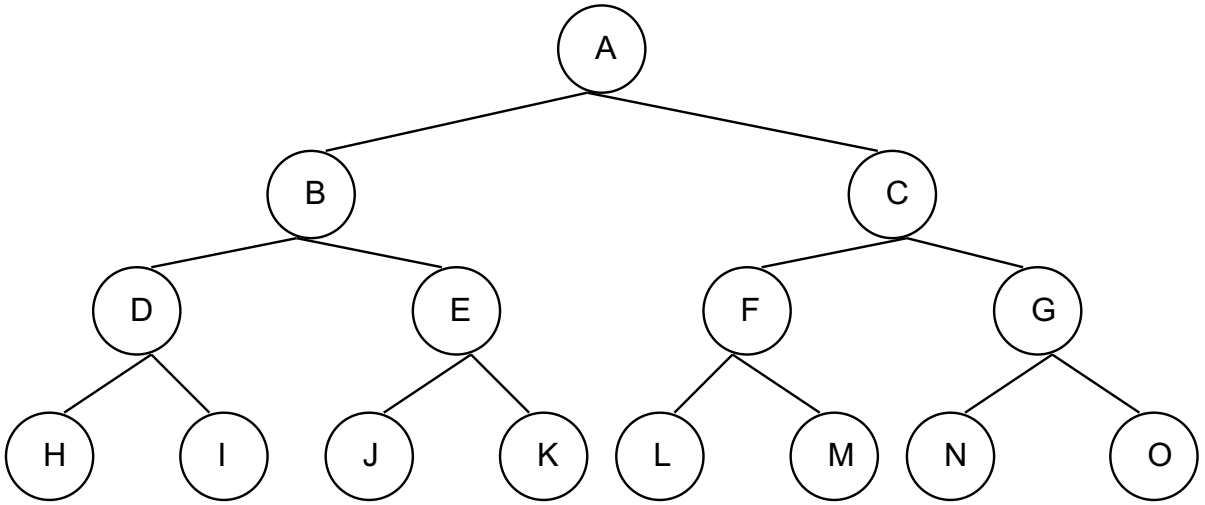
Dimungkinkan tidak memiliki *node*

**Jenis-jenis binary tree :**

- Full Binary Tree
- Complete Binary Tree
- Skewed Binary Tree

**Full Binary Tree**

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

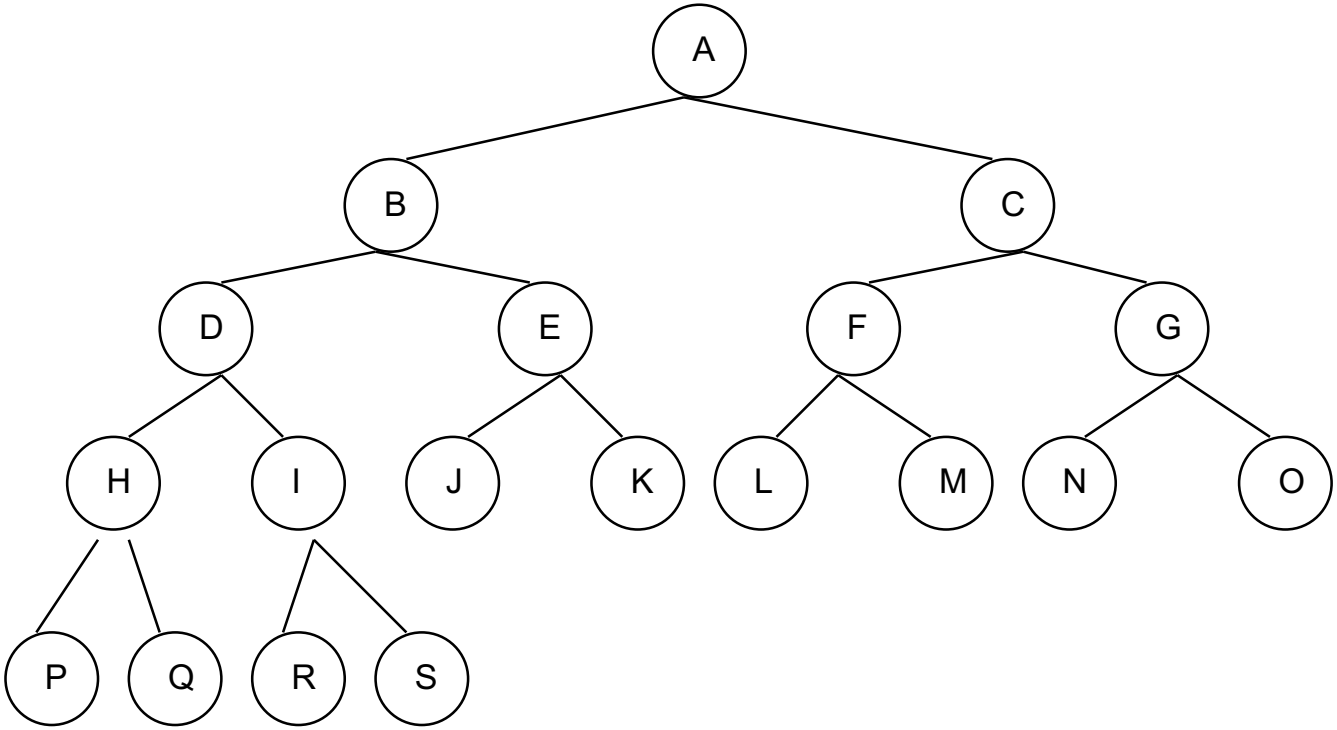



Banyaknya node maksimum pada tingkat N = $2^{(N-1)}$

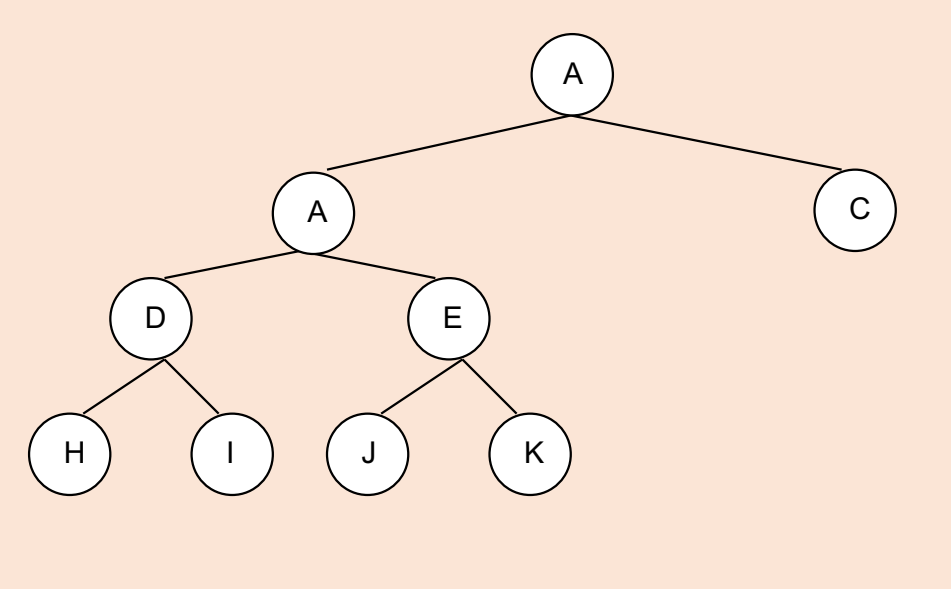
Banyaknya node maksimum sampai tingkat N = $\sum_{I=1}^{N} 2^{(i-1)}$

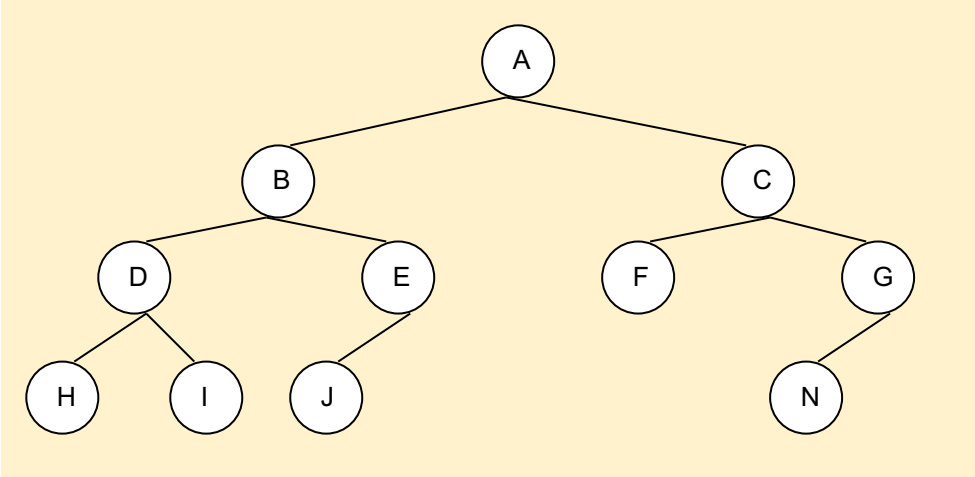
## Complete Binary Tree

Adalah binary tree yang memiliki leaves hanya pada satu level atau dua adjacent level (tingkat yang bertetangga), dimana leaves pada level yang terbawah selalu diletakkan mulai dari posisi kiri.

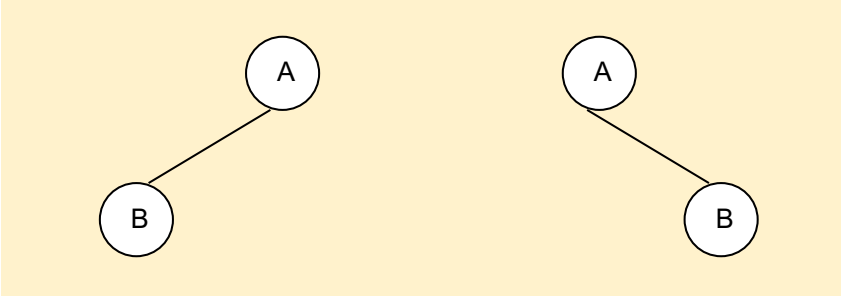


# Contoh Incomplete Binary Tree

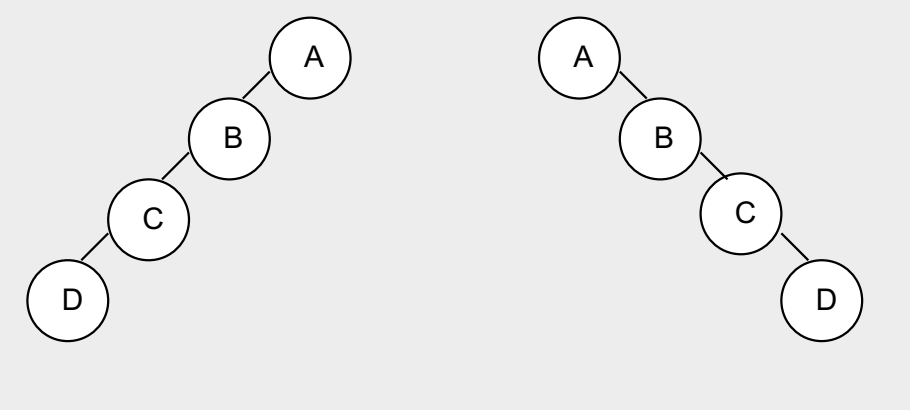


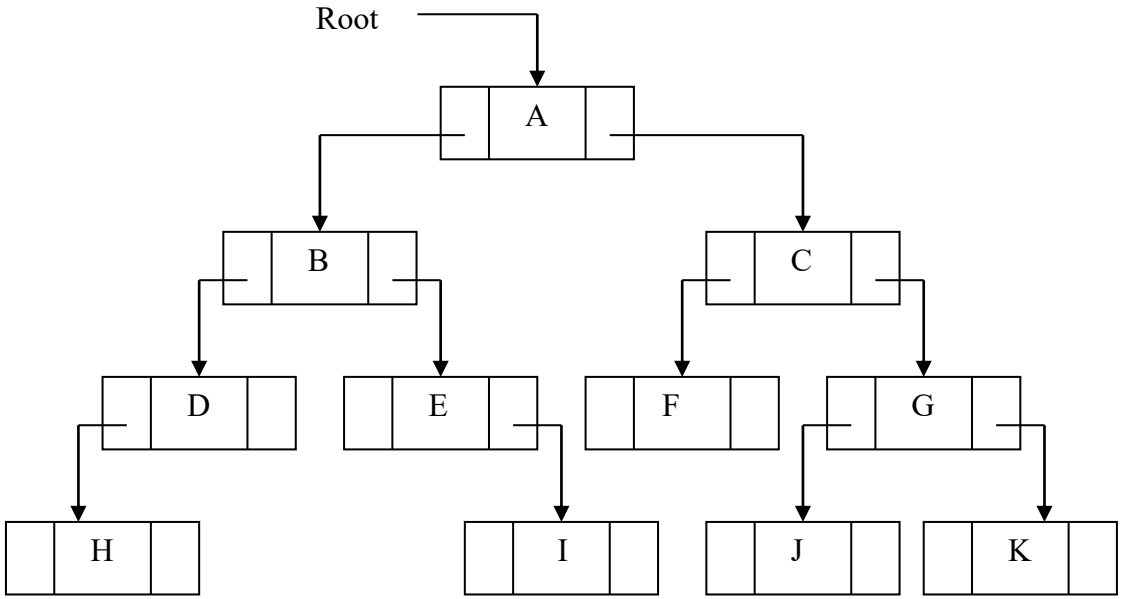
Contoh Dua Binary Tree yang berbeda




Skewed Binary Tree : binary tree yang semua nodenya (kecuali leaf) hanya memiliki satu child



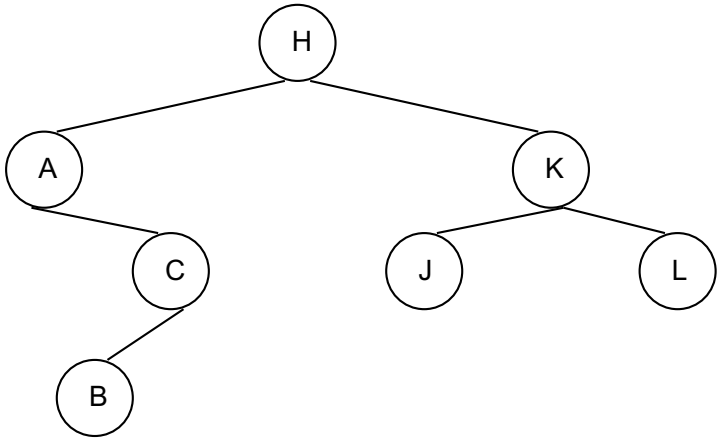
# Binary Tree dengan Linked List



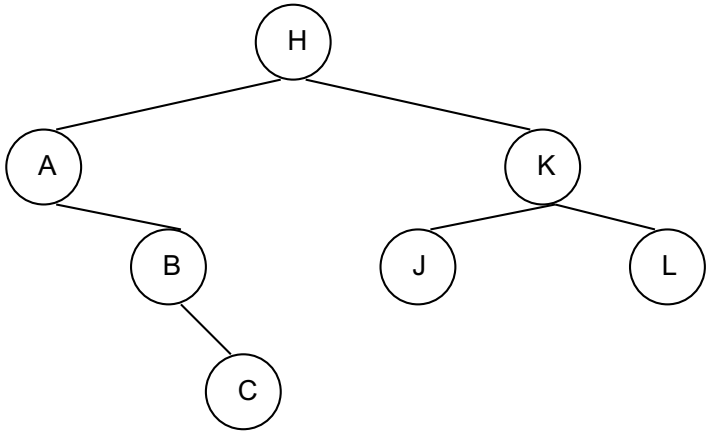
Contoh

Suatu list berisi HKACBLJ, buatlah bentuk Binary Treenya !

Algoritma yang digunakan :
Node yang berisi informasi yang nilainya lebih besar akan ditempatkan di cabang RIGHT, jika lebih kecil akan ditempatkan di LEFT




Jika Listnya berubah menjadi HAKBCLJ, maka bentuk binary treenya :



# Binary Tree Traversal

Tujuan : untuk memperoleh urutan informasi secara linear yang tersimpan dalam binary tree.

Traversal berdasarkan letak node yang dikunjungi :

- **Preorder (Depth First Order)**

  Cetak isi node → Kunjungi LEFT → Kunjungi RIGHT

- **Inorder (Symetric Order)**

  Kunjungi LEFT → Cetak isi node. → Kunjungi RIGHT

- **Postorder**

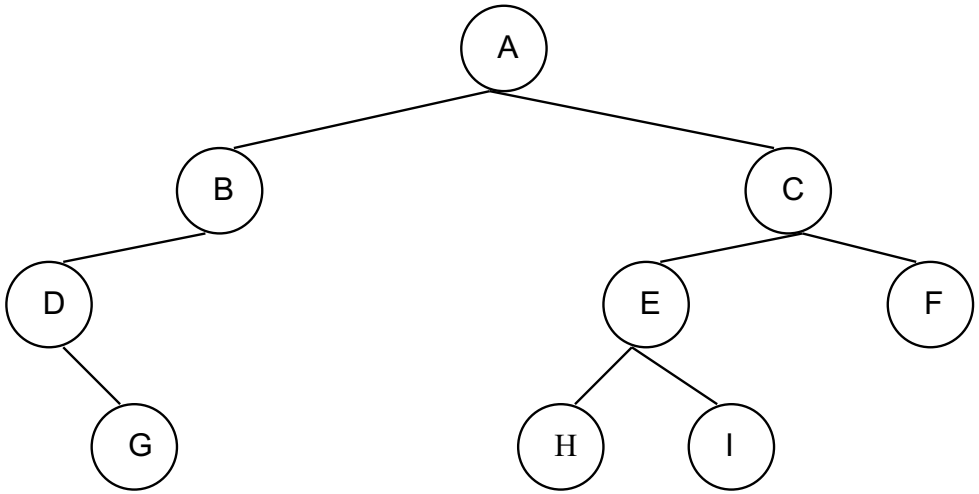
  Kunjungi LEFT. → Kunjungi RIGHT. → Cetak isi node

- **Levelorder** (berdasarkan kedudukan setiap node dalam tree)
  - Kunjungan simpul pada level yang sama dimulai dari root sampai node yang bertindak sebagai leave

- Prinsip arah yang digunakan dalam tree ada dua :
  - LRO (Left to Right Oriented)
  - RLO (Right to Left Oriented)

# Contoh

a)



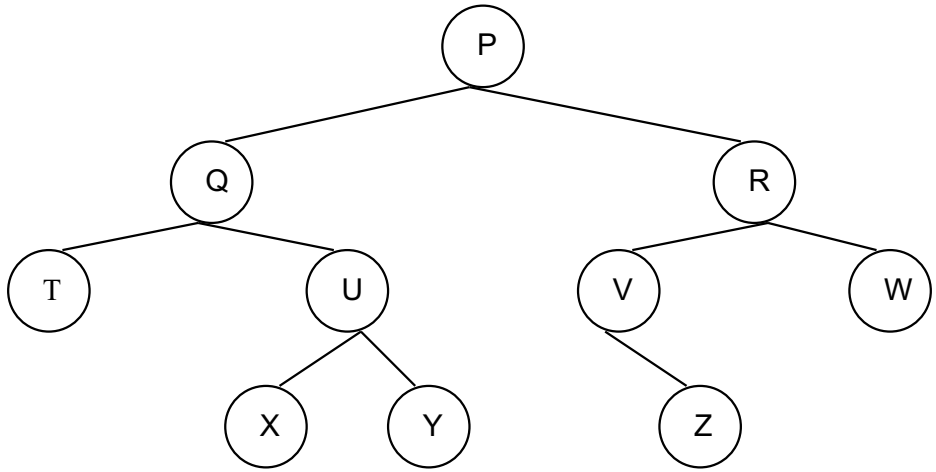

Preorder : A B D G C E H I F

Inorder : D G B A H E I C F

Postorder : G D B H I E F C A

Levelorder : A B C D E F G H I

b)




Preorder : P Q T U X Y R V Z W

Inorder : T Q X U Y P V Z R W

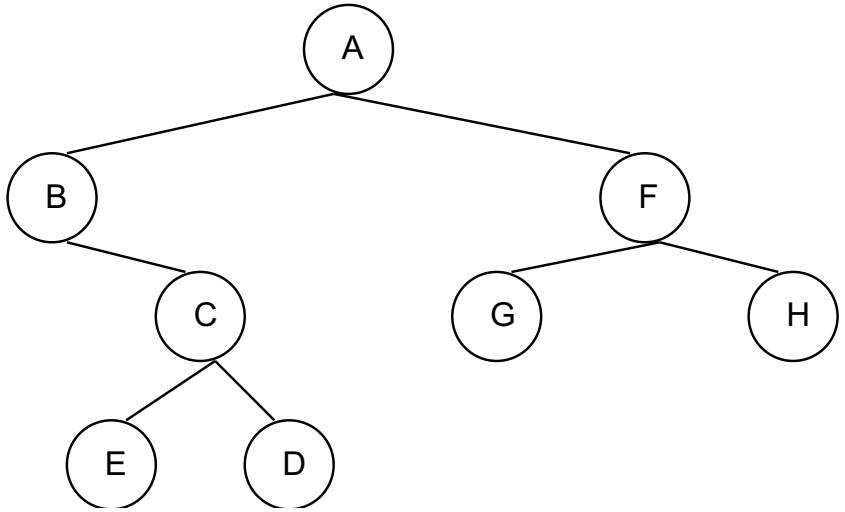
Postorder : T X Y U Q Z V W R P

Levelorder : P Q R T U V W X Y Z

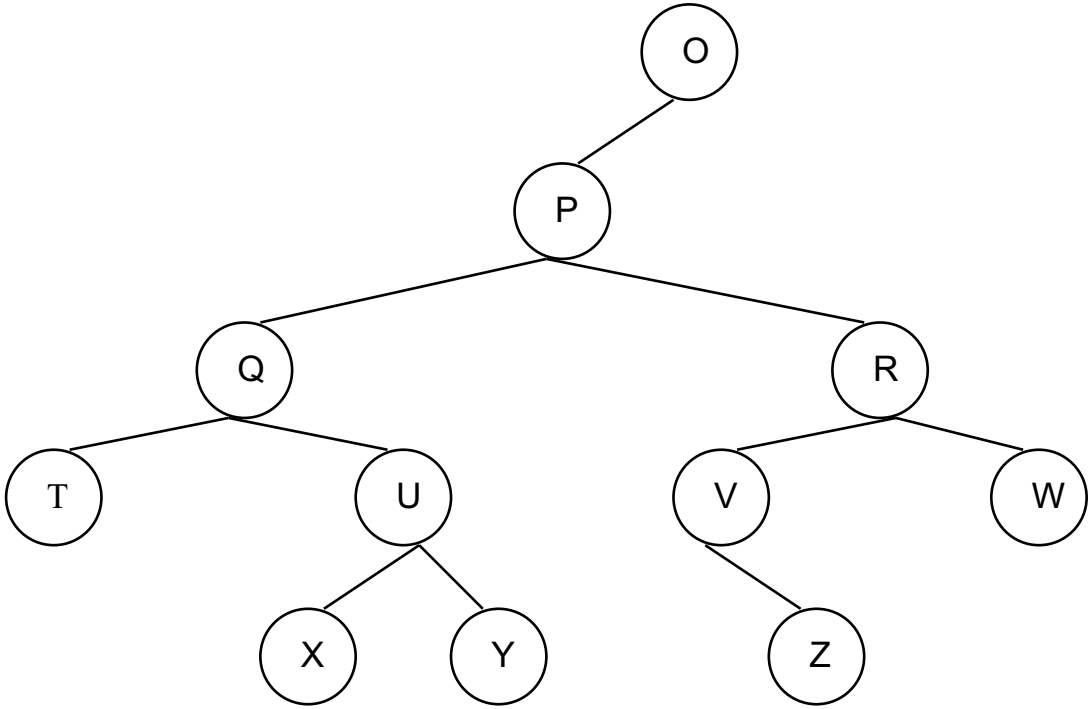
# Soal Latihan

- Buat list yang dihasilkan dari binary tree berikut menggunakan 4 metode traversal (preorder, inorder, postorder, levelorder)

a)



b)



# TERIMA KASIH

Next ---- PERTEMUAN ke-7

TEKNIK ELEKTRO
FTI UJ

ENDANG SRI RAHAYU