

# Pohon (Bag. 2)

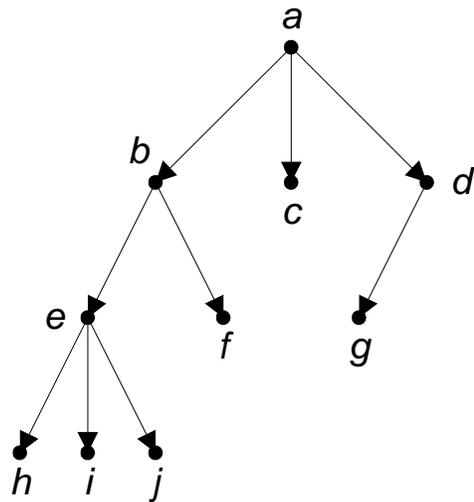
Bahan Kuliah *IF2120 Matematika Diskrit*

**Program Studi Teknik Informatika  
STEI- ITB**

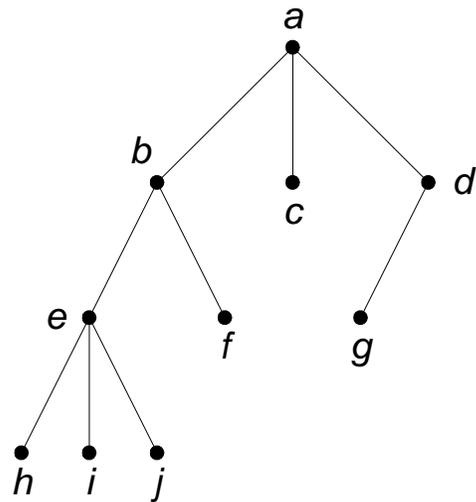


# Pohon berakar (*rooted tree*)

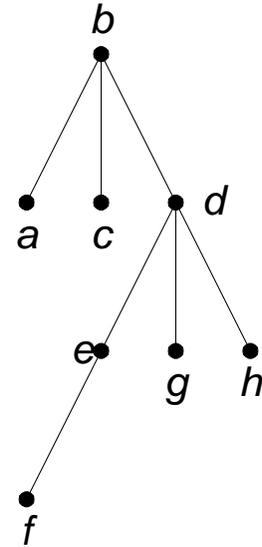
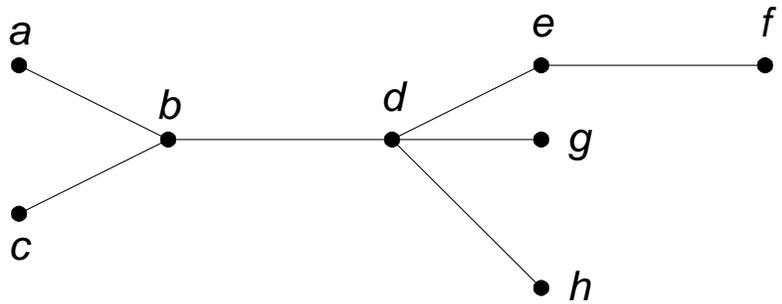
- Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan **pohon berakar** (*rooted tree*).



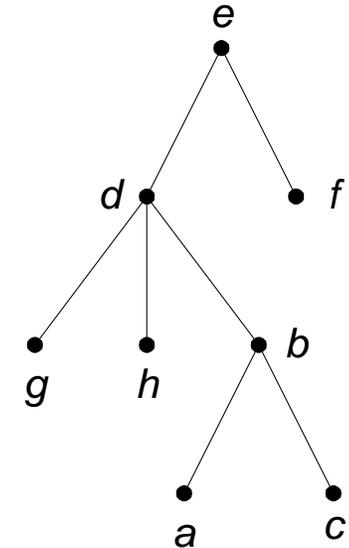
(a) Pohon berakar



(b) sebagai perjanjian, tanda panah pada sisi dapat dibuang



*b* sebagai akar



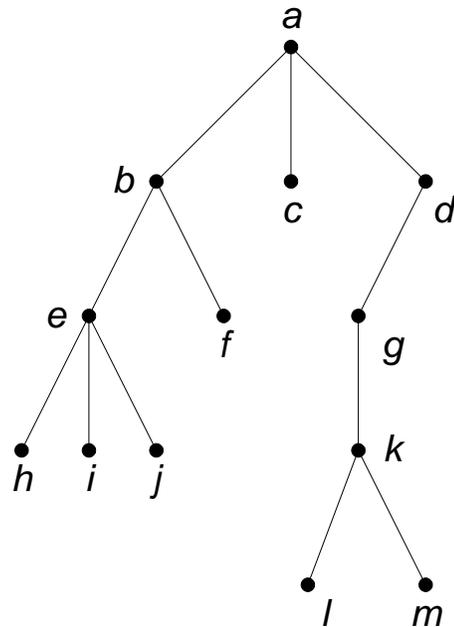
*e* sebagai akar

Pohon dan dua buah pohon berakar yang dihasilkan dari pemilihan dua simpul berbeda sebagai akar

# Terminologi pada Pohon Berakar

## Anak (*child* atau *children*) dan Orangtua (*parent*)

$b$ ,  $c$ , dan  $d$  adalah anak-anak simpul  $a$ ,  
 $a$  adalah orangtua dari anak-anak itu



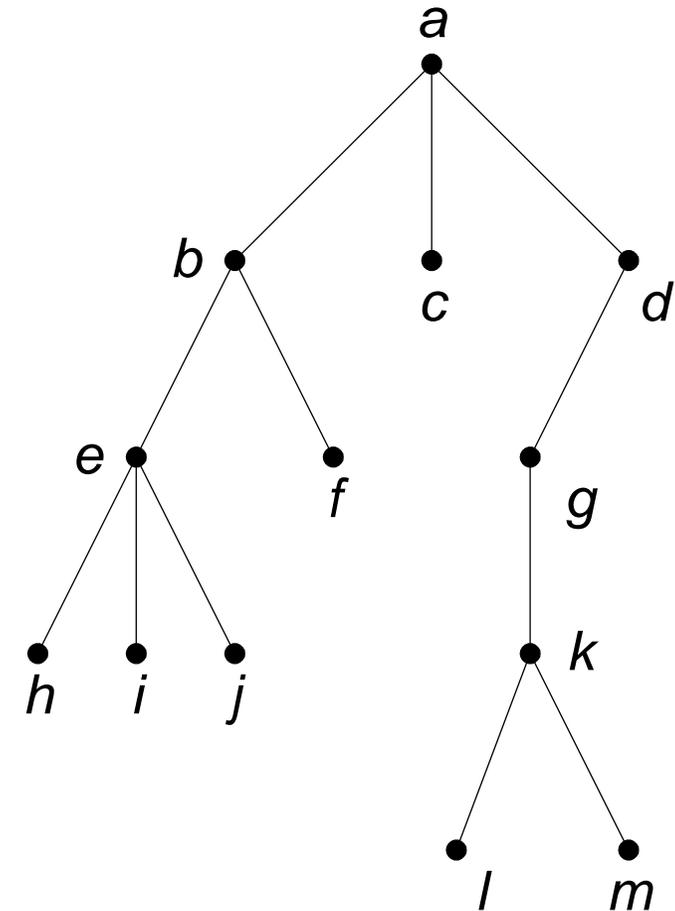
## 2. Lintasan (*path*)

Lintasan dari  $a$  ke  $j$  adalah  $a, b, e, j$ .

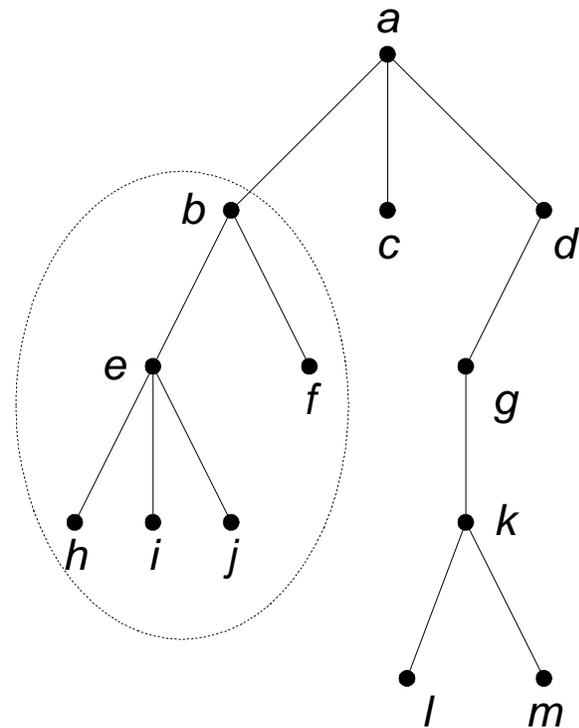
Panjang lintasan dari  $a$  ke  $j$  adalah 3.

## 3. Saudara kandung (*sibling*)

$f$  adalah saudara kandung  $e$ , tetapi  $g$  bukan saudara kandung  $e$ , karena orangtua mereka berbeda.



## 4. Upapohon (*subtree*)



## 5. Derajat (*degree*)

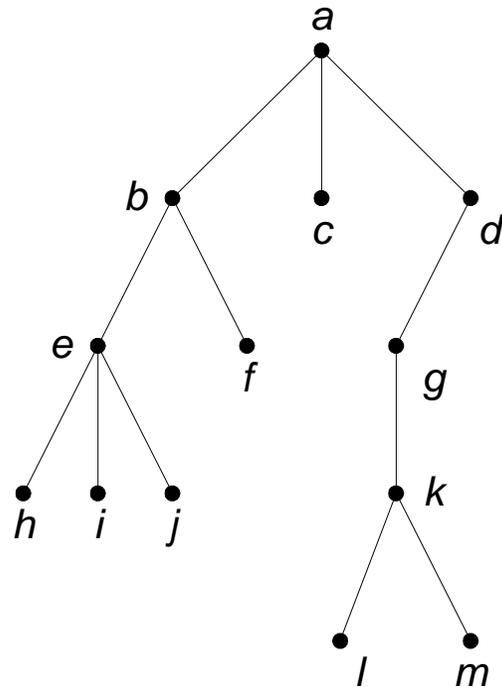
**Derajat** sebuah simpul adalah jumlah upapohon (atau jumlah anak) pada simpul tersebut.

Derajat  $a$  adalah 3, derajat  $b$  adalah 2,

Derajat  $d$  adalah satu dan derajat  $c$  adalah 0.

Jadi, derajat yang dimaksudkan di sini adalah derajat-keluar.

Derajat maksimum dari semua simpul merupakan derajat pohon itu sendiri. Pohon di bawah ini berderajat 3

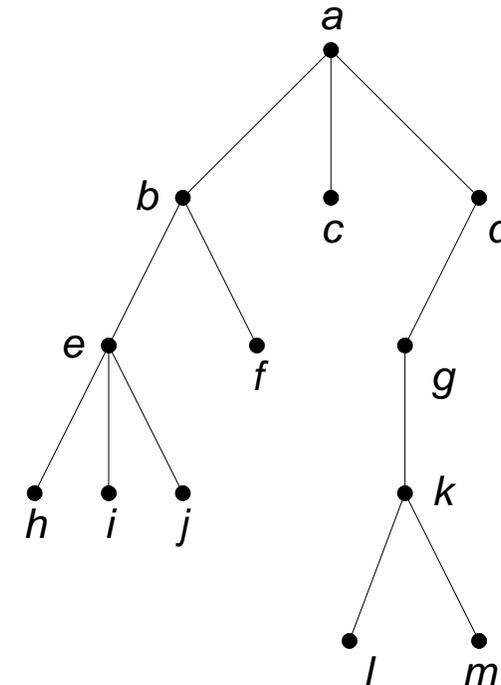


## 6. Daun (*leaf*)

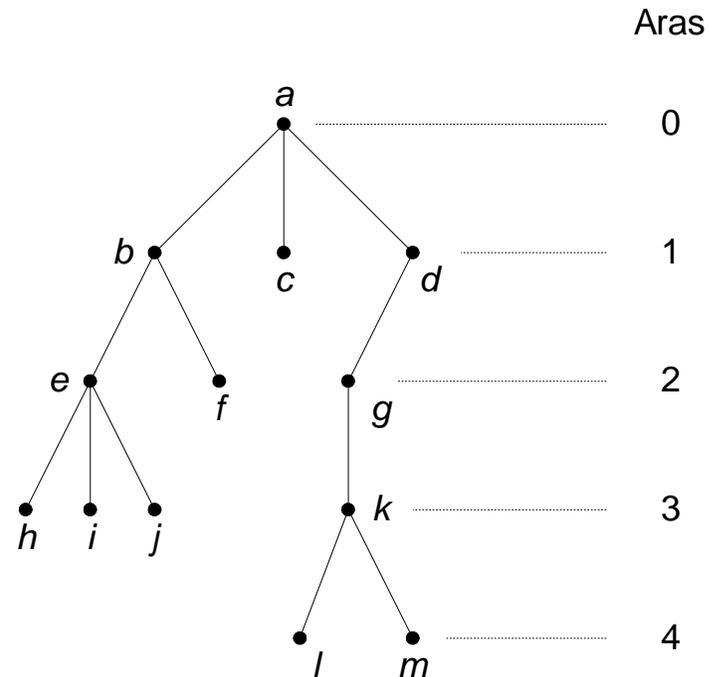
Simpul yang berderajat nol (atau tidak mempunyai anak) disebut **daun**. Simpul  $h, i, j, f, c, l,$  dan  $m$  adalah daun.

## 7. Simpul Dalam (*internal nodes*)

Simpul yang mempunyai anak disebut **simpul dalam**. Simpul  $b, d, e, g,$  dan  $k$  adalah simpul dalam.



## 8. Aras (*level*) atau Tingkat

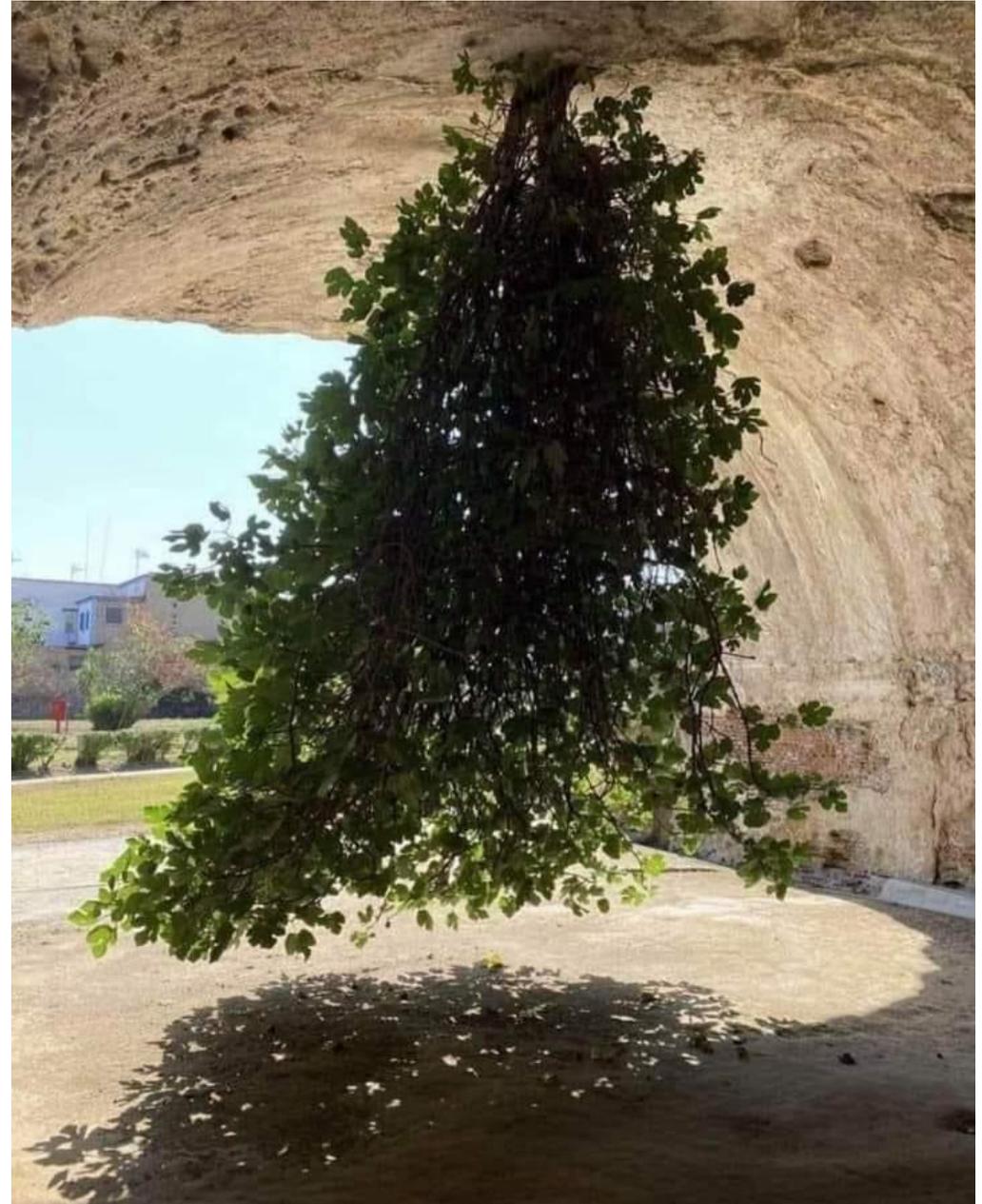


## 9. Tinggi (*height*) atau Kedalaman (*depth*)

Aras maksimum dari suatu pohon disebut **tinggi** atau **kedalaman** pohon tersebut. Pohon di atas mempunyai tinggi 4.

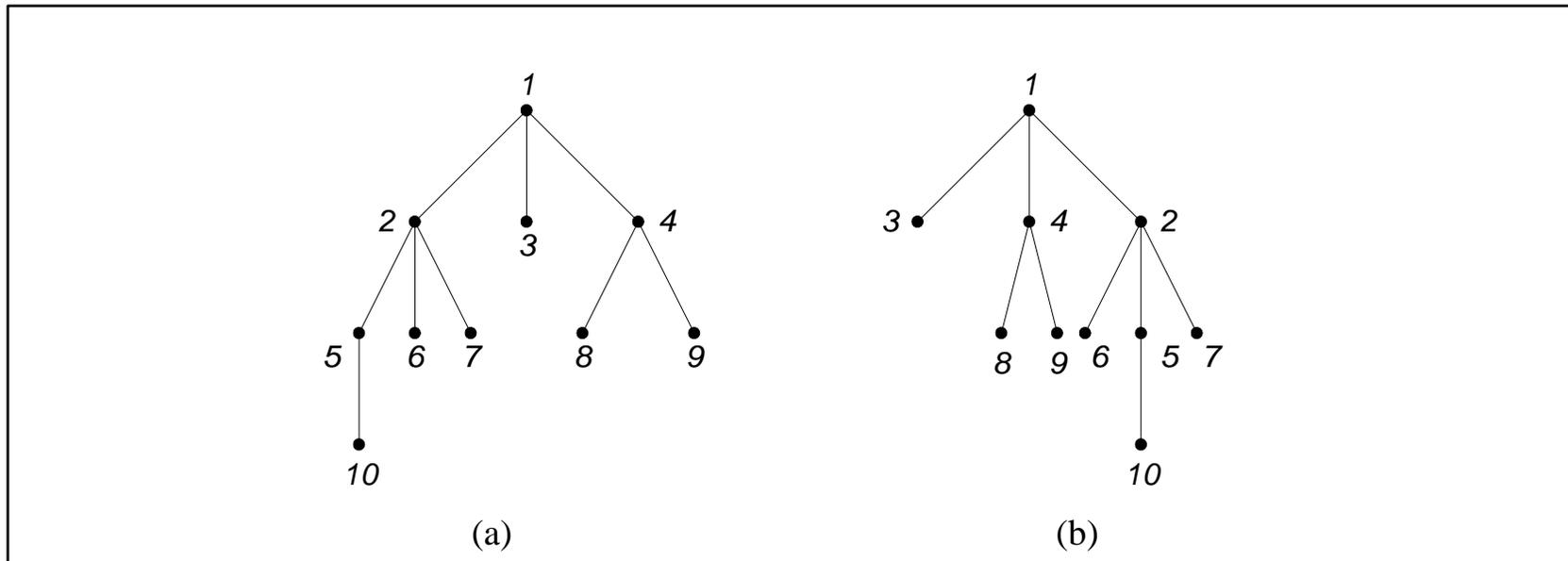
Ketemu juga pohon (*tree*) yang sesuai dengan teori graf. \*) 😊

\*) Akarnya di atas



# Pohon Terurut (*ordered tree*)

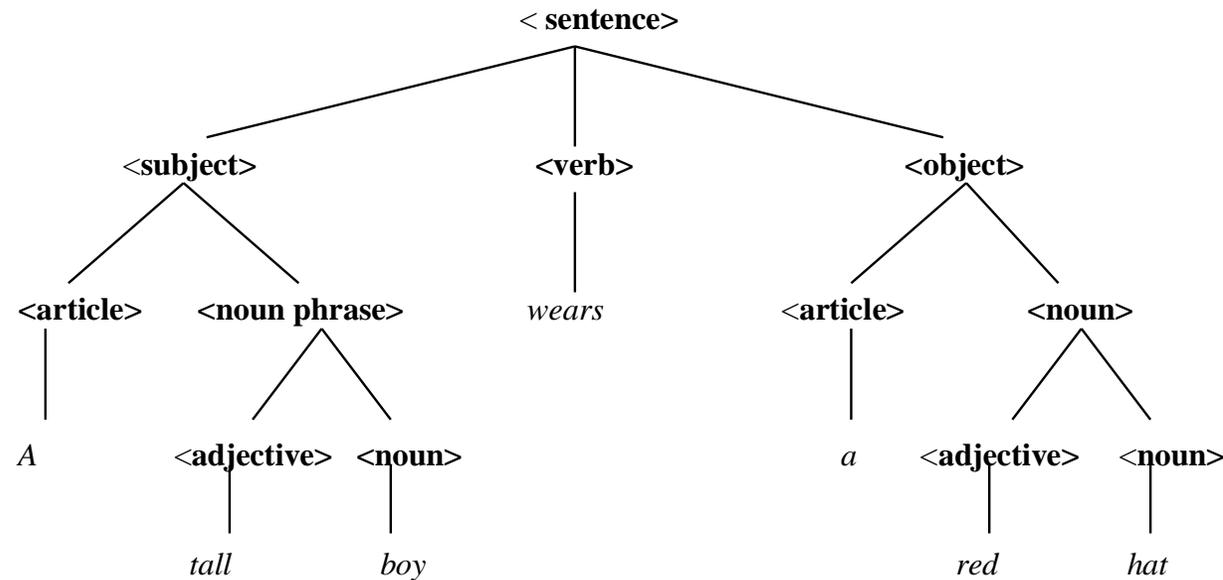
Pohon berakar yang urutan anak-anaknya penting disebut **pohon terurut** (*ordered tree*).



(a) dan (b) adalah dua pohon terurut yang berbeda

# Pohon $n$ -ary

- Pohon berakar yang setiap simpul cabangnya mempunyai paling banyak  $n$  buah anak disebut **pohon  $n$ -ary**.

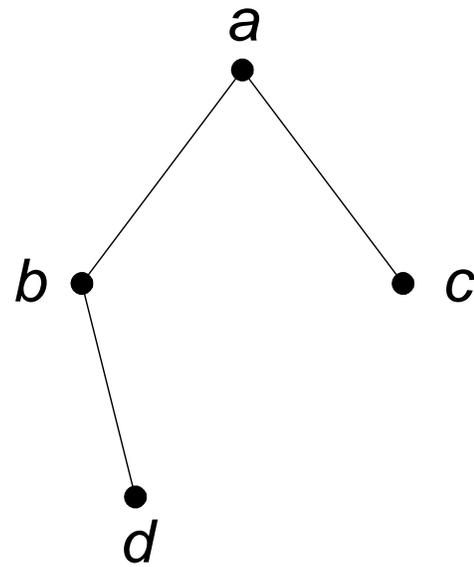
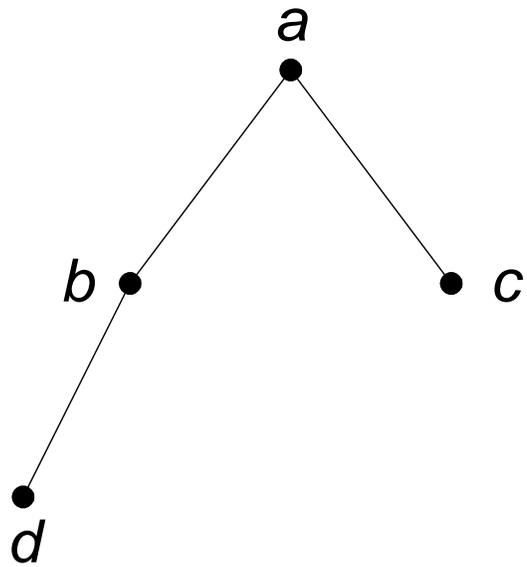


**Gambar** Pohon parsing dari kalimat *A tall boy wears a red hat*

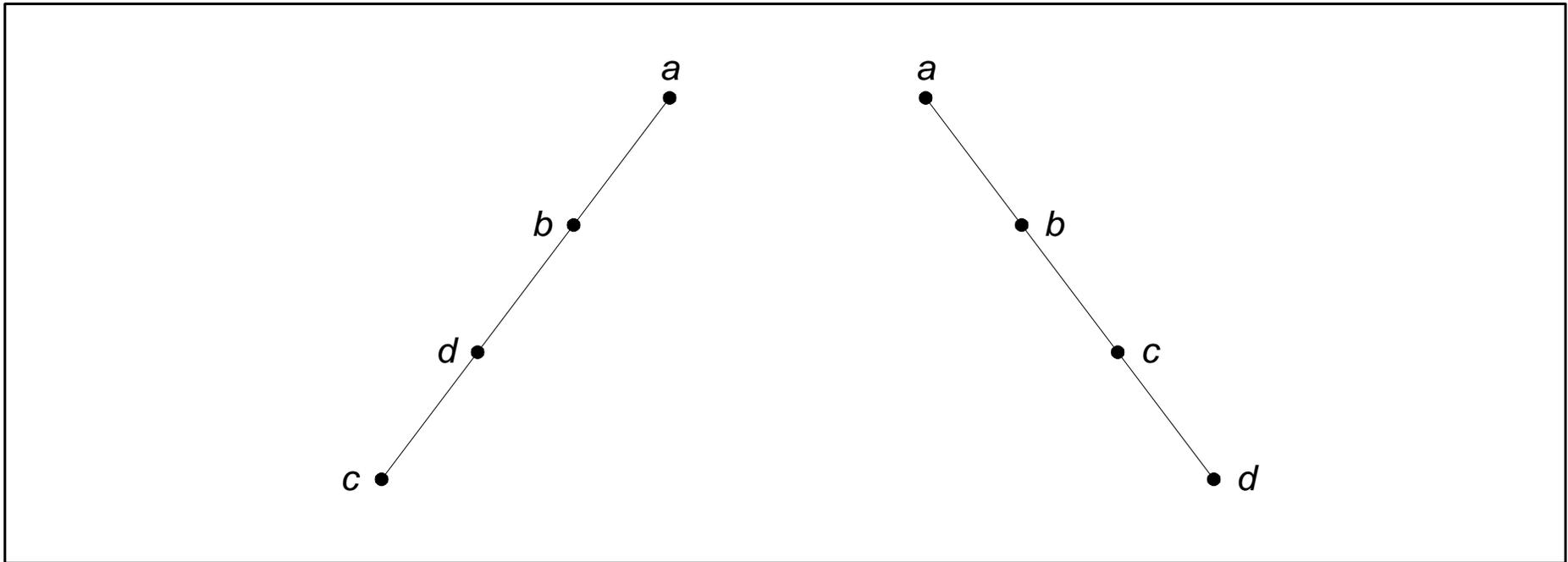
- Pohon  $n$ -ary dikatakan **teratur** atau **penuh** (*full*) jika setiap simpul cabangnya mempunyai tepat  $n$  anak.

# Pohon Biner (*binary tree*)

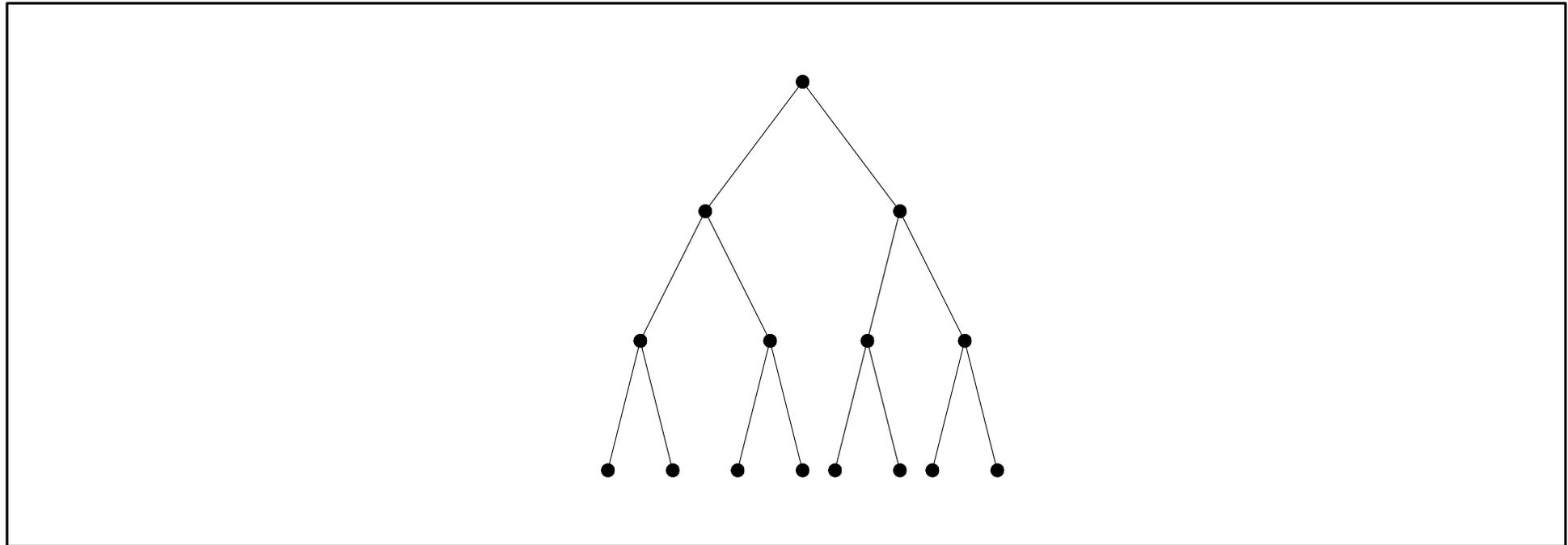
- Adalah pohon *n-ary* dengan  $n = 2$ .
- Pohon yang paling penting karena banyak aplikasinya.
- Setiap simpul di adlam pohon biner mempunyai paling banyak 2 buah anak.
- Dibedakan antara anak kiri (*left child*) dan anak kanan (*right child*)
- Karena ada perbedaan urutan anak, maka pohon biner adalah pohon terurut.



**Gambar** Dua buah pohon biner yang berbeda



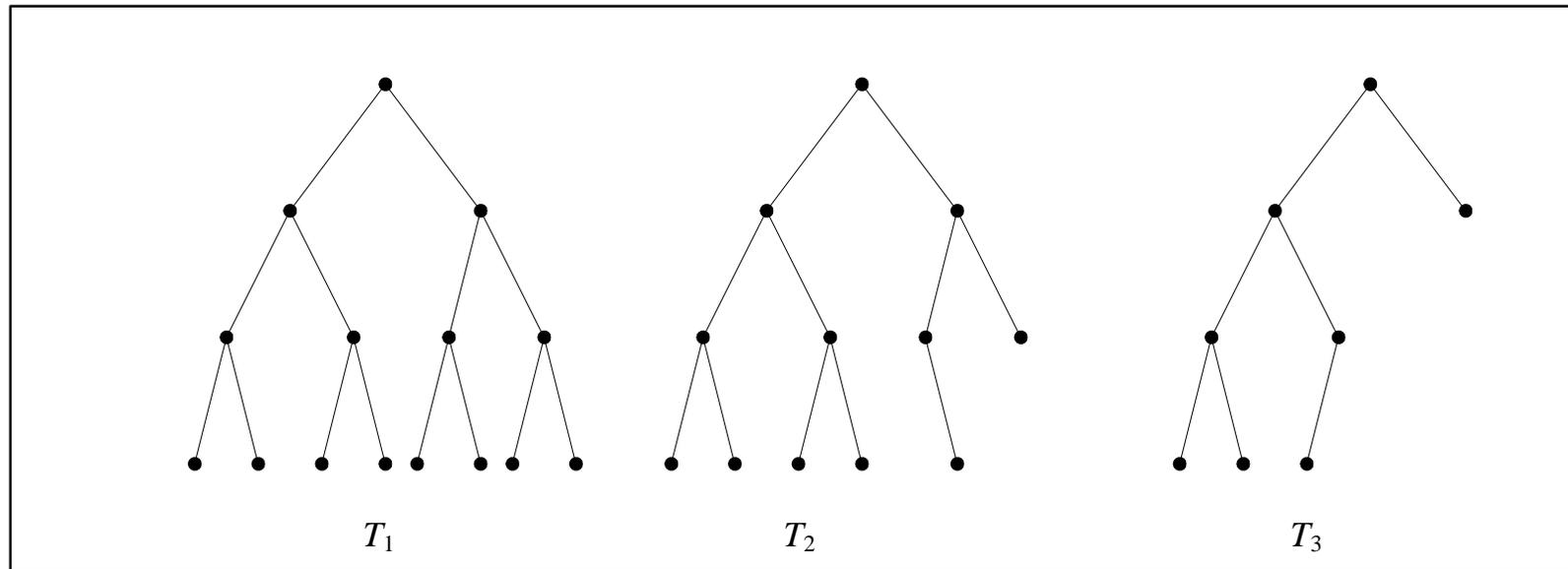
**Gambar** (a) Pohon condong-kiri, dan (b) pohon condong kanan



**Gambar** Pohon biner penuh

## Pohon Biner Seimbang

Pada beberapa aplikasi, diinginkan tinggi upapohon kiri dan tinggi upapohon kanan yang seimbang, yaitu berbeda maksimal 1.

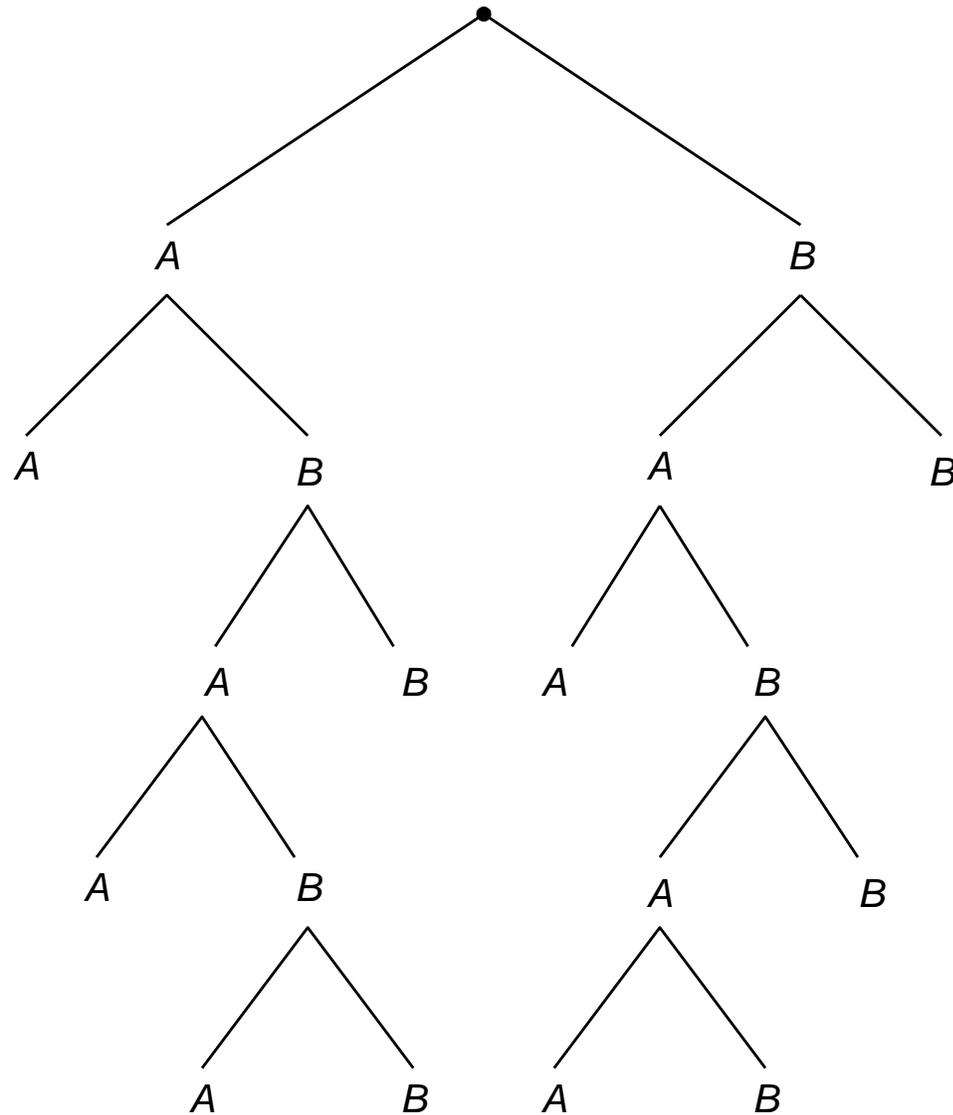


**Gambar**  $T_1$  dan  $T_2$  adalah pohon seimbang, sedangkan  $T_3$  bukan pohon seimbang.

# Latihan

Gunakan pohon berakar untuk menggambarkan semua kemungkinan hasil dari pertandingan tenis antara dua orang pemain, Anton dan Budi, yang dalam hal ini pemenangnya adalah pemain yang pertama memenangkan dua set berturut-turut atau pemain yang pertama memenangkan total tiga set.

Jawaban:



# Latihan

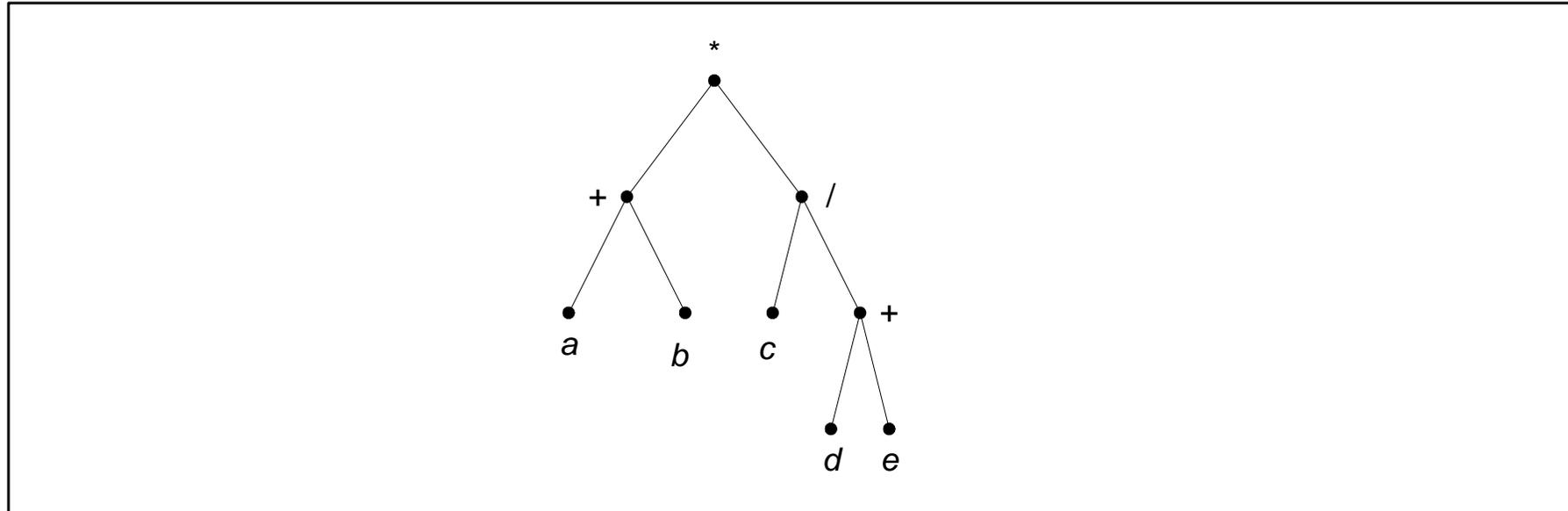
Diketahui 8 buah koin uang logam. Satu dari delapan koin itu ternyata palsu. Koin yang palsu mungkin lebih ringan atau lebih berat daripada koin yang asli. Misalkan tersedia sebuah timbangan neraca yang sangat teliti. Buatlah pohon keputusan untuk mencari uang palsu dengan cara menimbang paling banyak hanya 3 kali saja.



(Petunjuk: misalkan 8 koin tersebut adalah a, b, c, d, e, f, g, dan h)

# Terapan Pohon Biner

## 1. Pohon Ekspresi

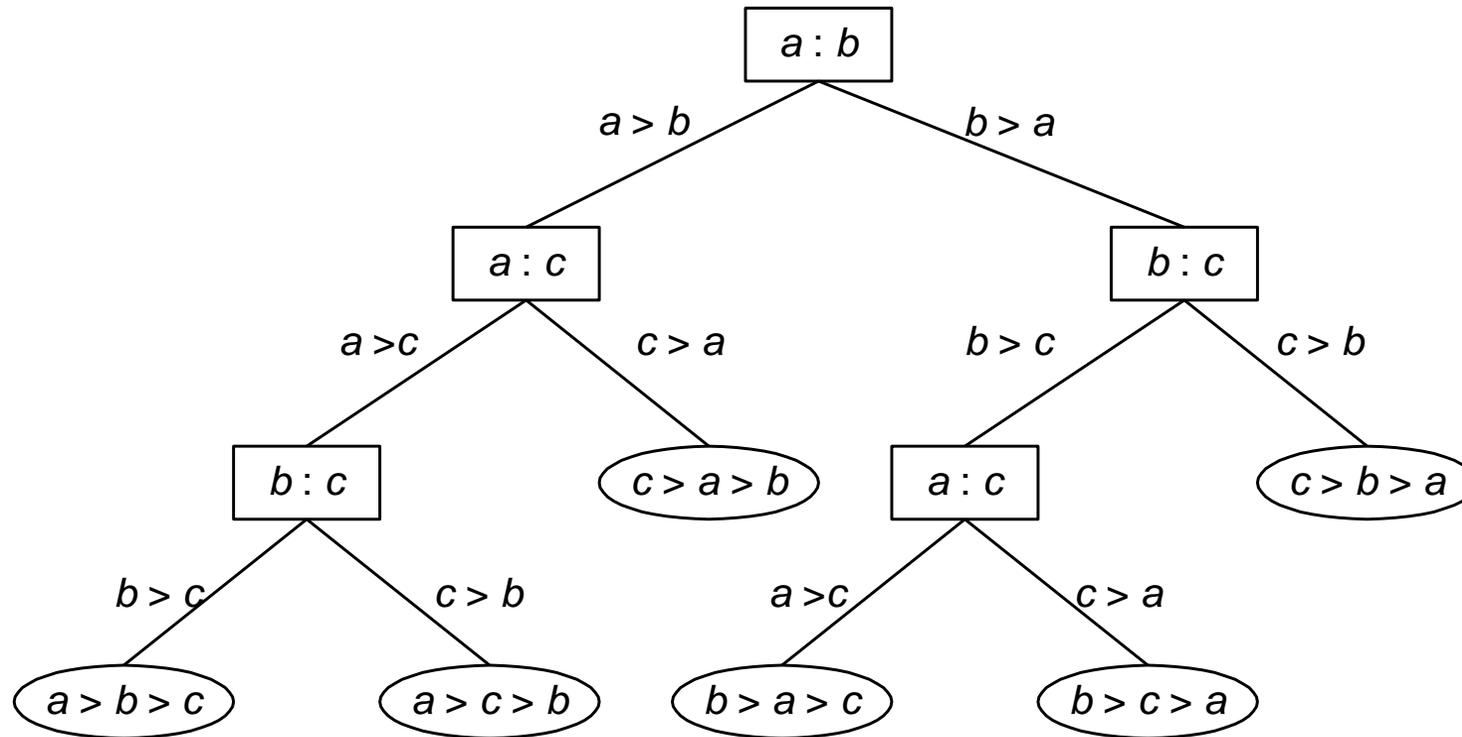


**Gambar** Pohon ekspresi dari  $(a + b) * (c / (d + e))$

daun  $\rightarrow$  *operand*

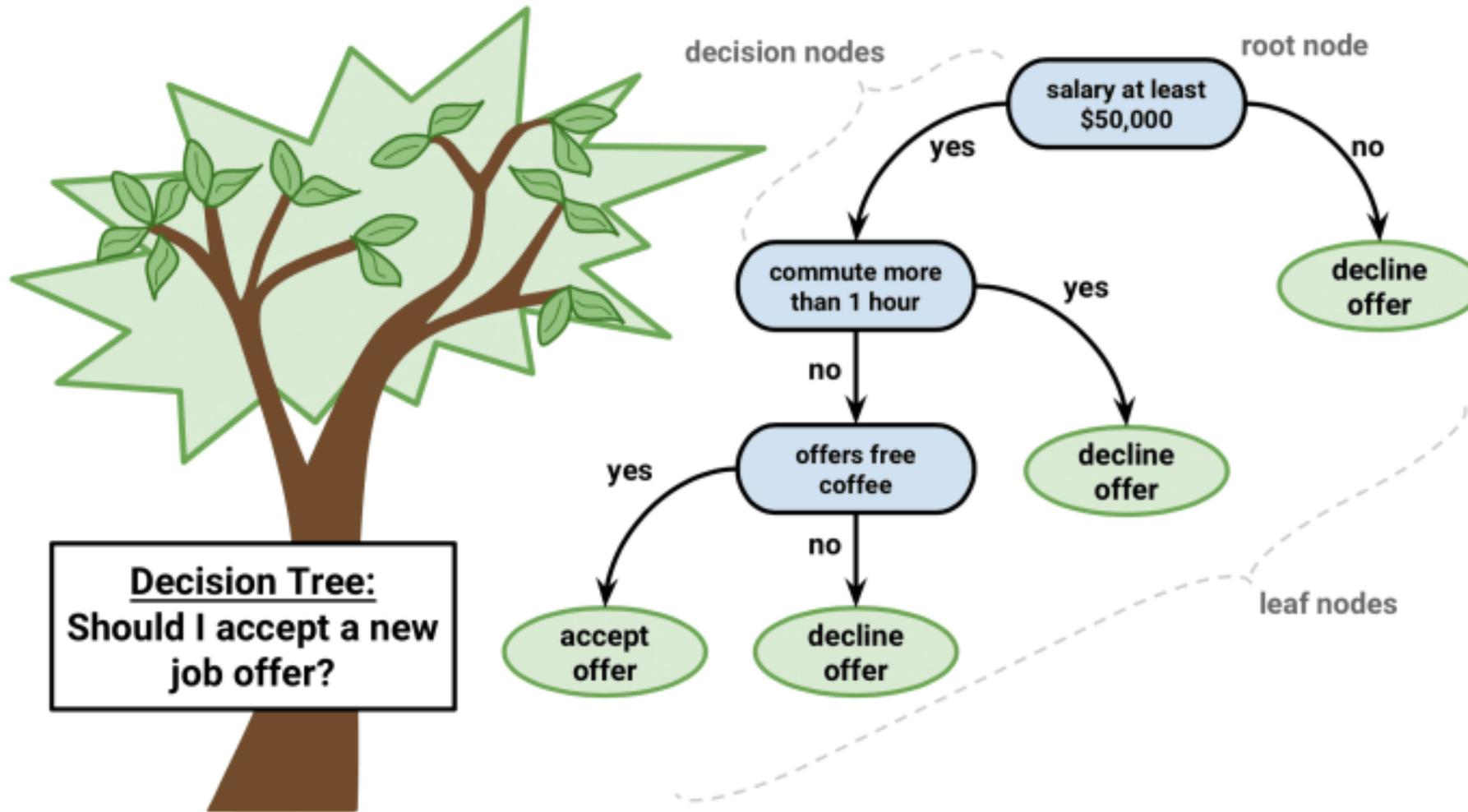
simpul dalam  $\rightarrow$  *operator*

## 2. Pohon Keputusan

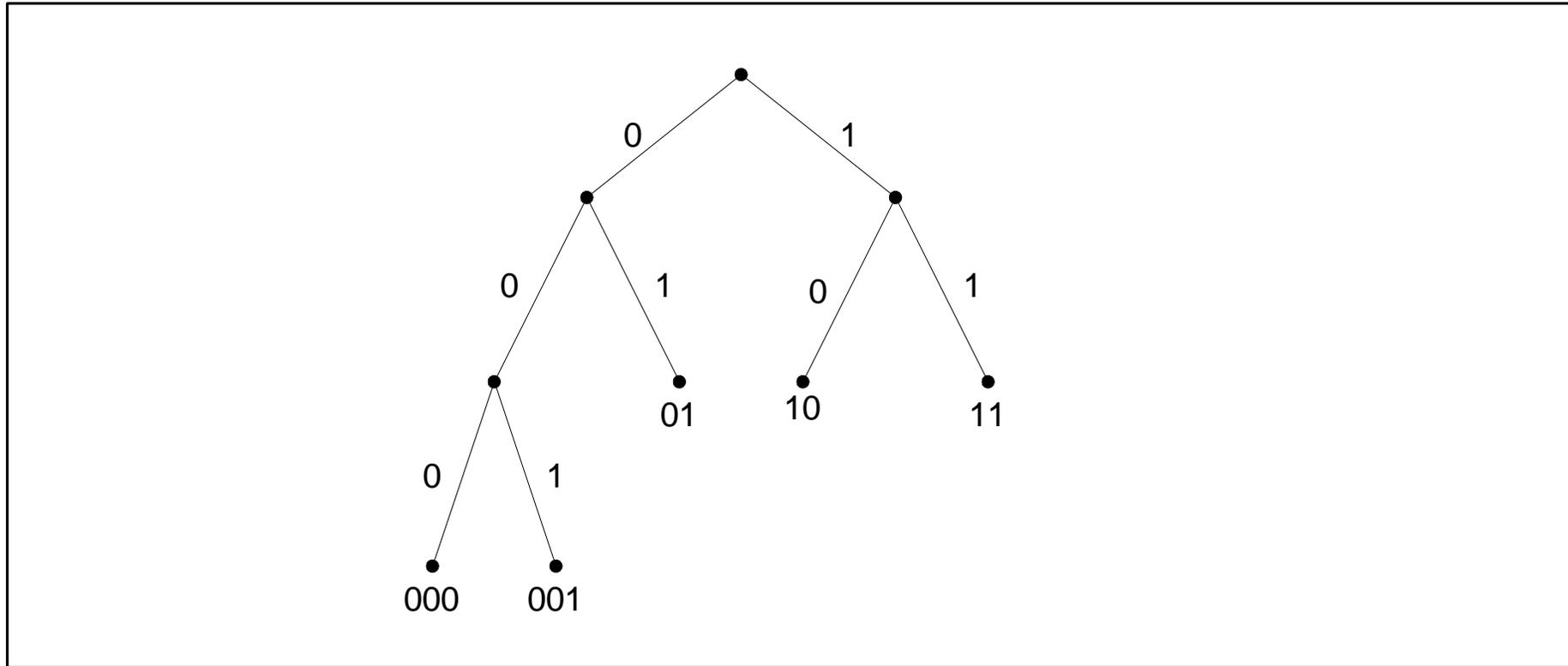


**Gambar** Pohon keputusan untuk mengurutkan 3 buah elemen

# Pohon keputusan di dalam *Machine Learning (ML)*



### 3. Kode Awalan



**Gambar** Pohon biner dari kode prefiks { 000, 001, 01, 10, 11 }

## 4. Kode Huffman

**Tabel** Kode ASCII

Simbol	Kode ASCII
<i>A</i>	01000001
<i>B</i>	01000010
<i>C</i>	01000011
<i>D</i>	01000100

rangkaian bit untuk string ‘*ABACCDA*’:

01000001010000010010000010100000110100000110100010001000001

atau  $7 \times 8 = 56$  bit (*7 byte*).

**Tabel** Tabel kekerapan (frekuensi) dan kode Huffman untuk *string* *ABACCCA*

Simbol	Kekerapan	Peluang	Kode Huffman
<i>A</i>	3	$3/7$	0
<i>B</i>	1	$1/7$	110
<i>C</i>	2	$2/7$	10
<i>D</i>	1	$1/7$	111

Dengan kode Huffman, rangkaian bit untuk '*ABACCCA*':

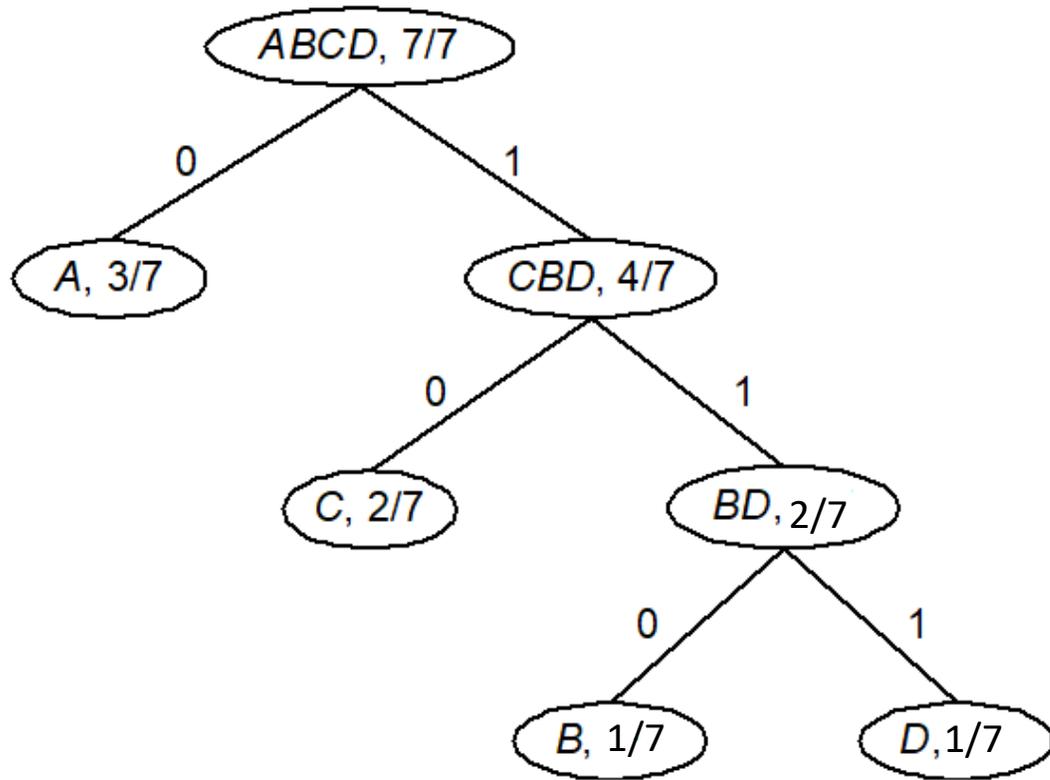
0110010101110

hanya 13 bit!

**Bagaimana cara memperoleh kode Huffman?**

## Algoritma pembentukan kode Huffman

1. Pilih dua simbol dengan peluang (*probability*) paling kecil (pada contoh di atas simbol  $B$  dan  $D$ ). Kedua simbol tadi dikombinasikan sebagai simpul orangtua dari simbol  $B$  dan  $D$  sehingga menjadi simbol  $BD$  dengan peluang  $1/7 + 1/7 = 2/7$ , yaitu jumlah peluang kedua anaknya.
2. Selanjutnya, pilih dua simbol berikutnya, termasuk simbol baru, yang mempunyai peluang terkecil.
3. Ulangi langkah 1 dan 2 sampai seluruh simbol habis.
4. Beri label secara konsisten sisi kiri dengan 0 dan sisi kanan dengan 1.
5. Lintasan dari akar ke daun berisi sisi-sisi pohon yang deretan labelnya menyatakan kode Huffman untuk simbol daun tersebut



- Kode Huffman: A = 0, C = 10, B = 110, D = 111

### Encoding:

- Kodekan setiap simbol di dalam pesan tadi dengan kode Huffman:
- Contoh: ABACCCA = 0110010101110

### Decoding:

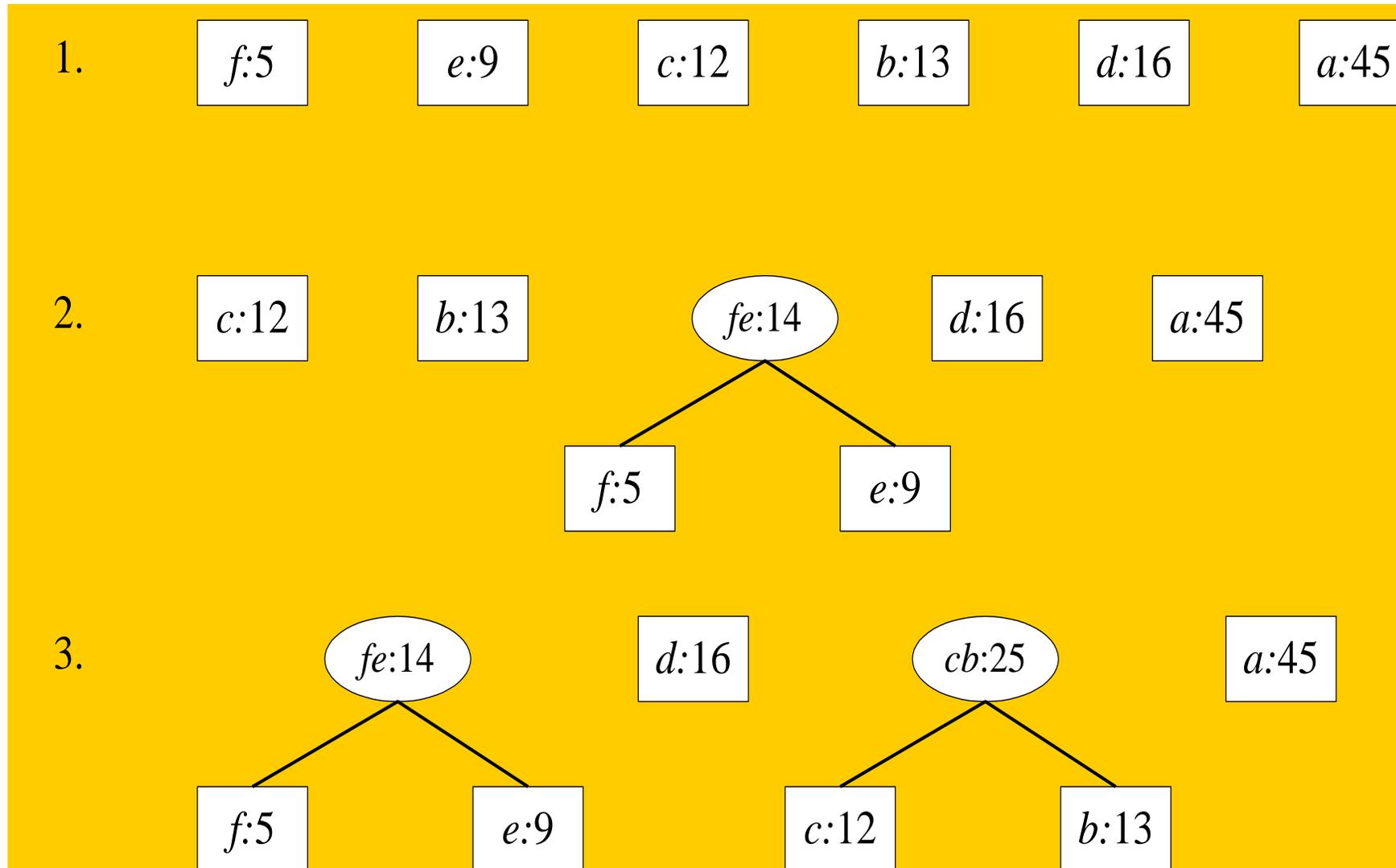
1. Baca simbol biner pertama
2. Mulai traversal dari akar mengikuti sisi yang sesuai dengan simbol biner tersebut
3. Baca terus simbol biner dan traversal sisi yang bersesuaian sampai ketemu daun. Kodekan barisan biner yang sudah dibaca dengan simbol daun
4. Baca simbol biner berikutnya dan ulangi traversal dari akar. Ulangi langkah 4 sampai semua simbol biner habis

**Contoh lain:** Sebuah pesan sepanjang 100 karakter disusun oleh huruf a, b, c, d, e, dan f. Frekuensi kemunculan setiap huruf di dalam pesan adalah sebagai berikut:

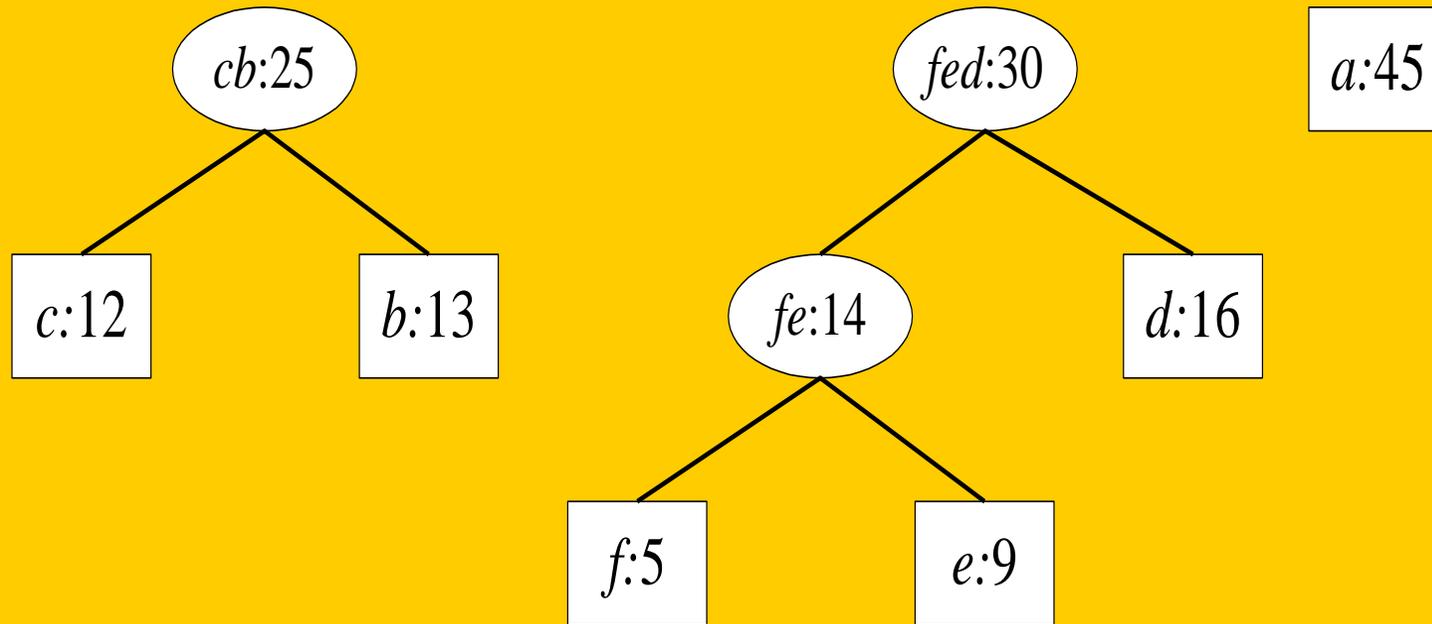
Karakter	<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>
Frekuensi	45	13	12	16	9	5

Tentukan kode Huffman untuk setiap huruf tersebut!

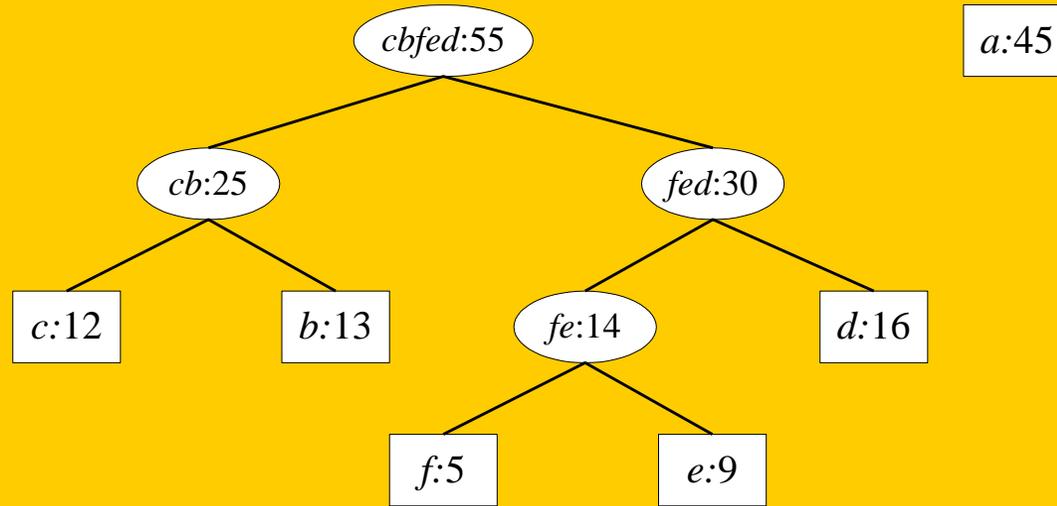
# Penyelesaian:



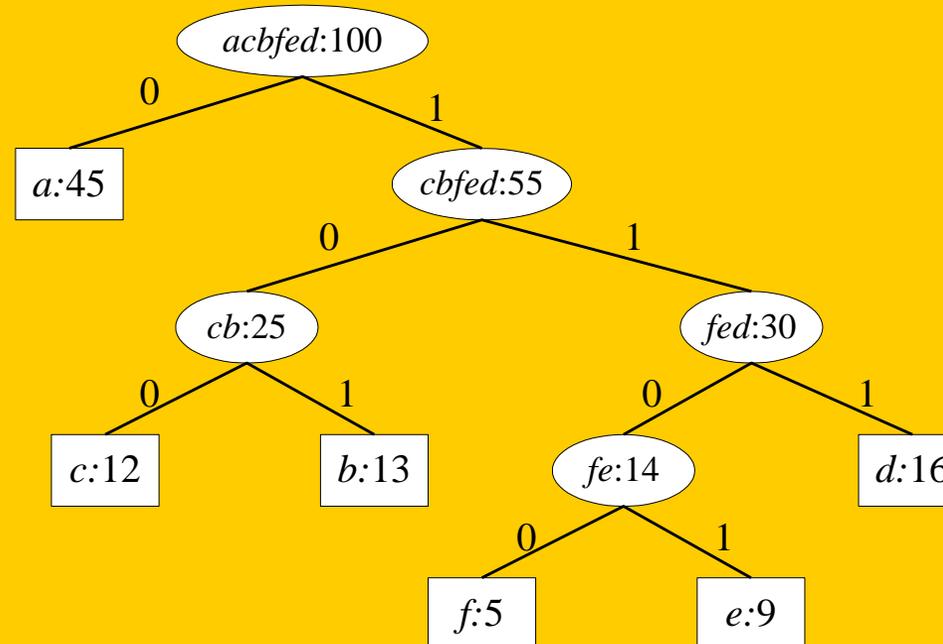
4.



5.



6



Kode Huffman:

a = 0

b = 101

c = 100

d = 111

e = 1101

f = 1100

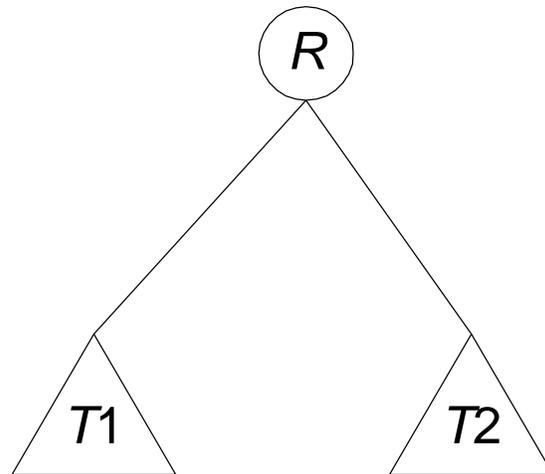
# Latihan

Lakukan kompresi pesan ADA APA DENGAN CINTA (termasuk spasi) dengan menggunakan kode Huffman. Gambarkan pohonnya, lalu kodekan pesan tersebut dengan kode Huffman, lalu hitung rasio kompresinya. Rasio kompresi dihitung dengan rumus:

$$\text{rasio} = (\text{jumlah bit setelah kompresi} / \text{jumlah bit sebelum kompresi}) \times 100\%$$

Jawaban:

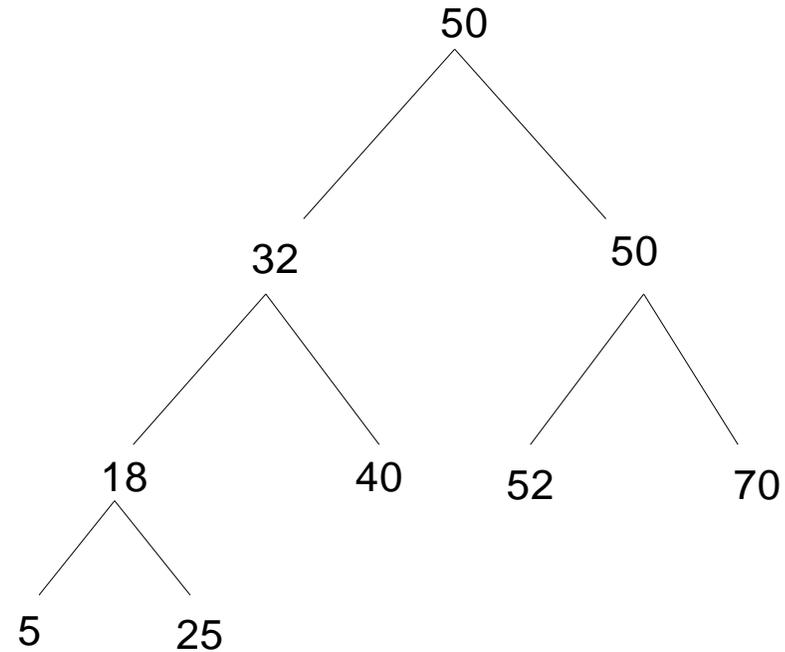
## 5. Pohon Pencarian Biner



$\text{Kunci}(T1) < \text{Kunci}(R)$

$\text{Kunci}(T2) > \text{Kunci}(R)$

Data: 50, 32, 18, 40, 60, 52, 5, 25, 70



Berapa banyak perbandingan yang dilakukan untuk menemukan 25?  
Berapa banyak perbandingan yang dilakukan untuk menemukan 100?

# Penelusuran (traversal) Pohon Biner

1. *Preorder* :  $R, T_1, T_2$

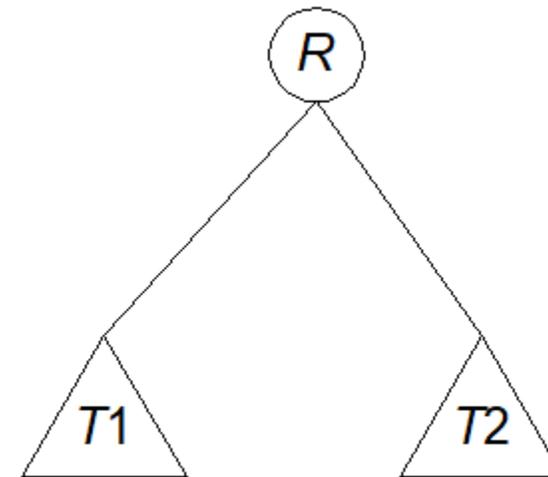
- kunjungi  $R$
- kunjungi  $T_1$  secara *preorder*
- kunjungi  $T_2$  secara *preorder*

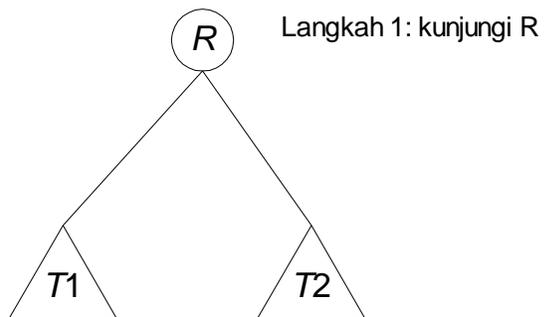
2. *Inorder* :  $T_1, R, T_2$

- kunjungi  $T_1$  secara *inorder*
- kunjungi  $R$
- kunjungi  $T_2$  secara *inorder*

3. *Postorder* :  $T_1, T_2, R$

- kunjungi  $T_1$  secara *postorder*
- kunjungi  $T_2$  secara *postorder*
- kunjungi  $R$

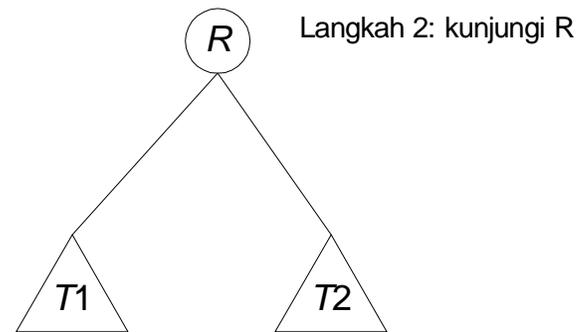




Langkah 2: kunjungi T1  
secara *preorder*

Langkah 3: kunjungi T2  
secara *preorder*

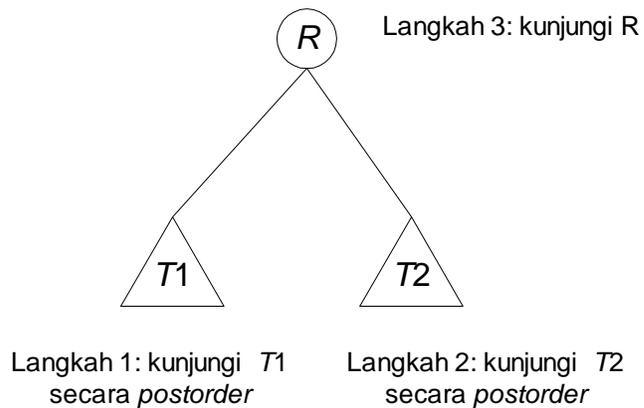
(a) *preorder*



Langkah 1: kunjungi T1  
secara *inorder*

Langkah 3: kunjungi T2  
secara *inorder*

(b) *inorder*

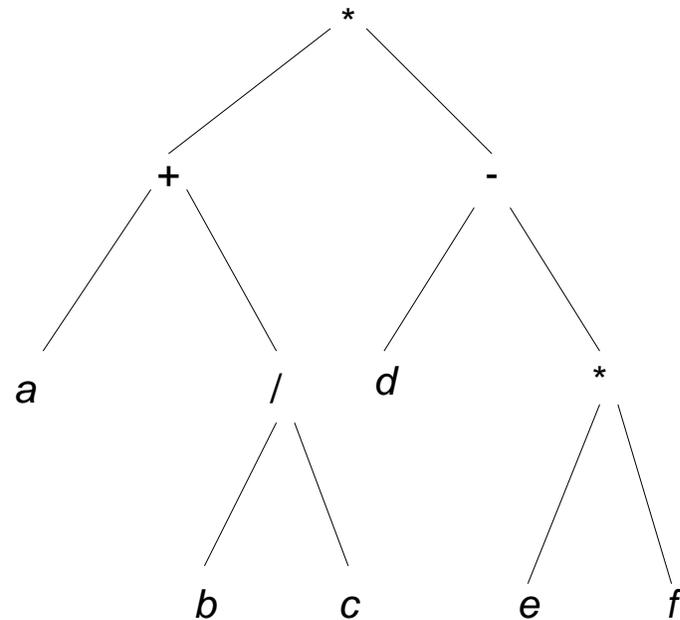


Langkah 1: kunjungi T1  
secara *postorder*

Langkah 2: kunjungi T2  
secara *postorder*

(c) *postorder*

*preorder* :  $* + a / b c - d * e f$  (prefix)  
*inorder* :  $a + b / c * d - e * f$  (infix)  
*postorder* :  $a b c / + d e f * - *$  (postfix)



# Latihan

Diberikan masukan berupa rangkaian karakter dengan urutan sebagai berikut:

*P, T, B, F, H, K, N, S, A, U, M, I, D, C, W, O*

- (a) Gambarkan pohon pencarian (*search tree*) yang terbentuk.
- (b) Tentukan hasil penelusuran *preorder*, *inorder*, dan *postorder*, dari pohon jawaban (a) di atas.

# Latihan

Tentukan hasil kunjungan *preorder*, *inorder*, dan *postorder* pada pohon 4-*ary* berikut ini:

