Pohon (Bag. 2)

(Update 2024)

Bahan Kuliah

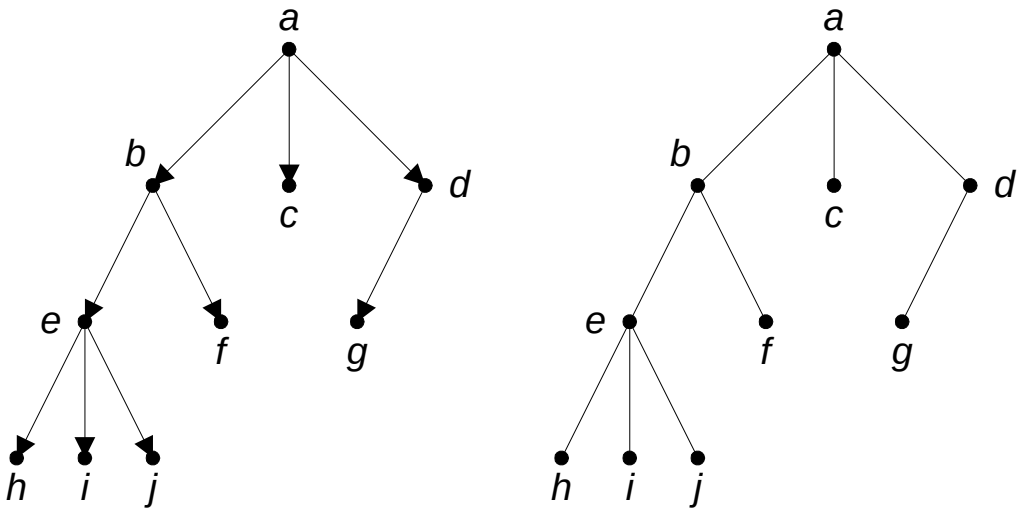
IF1220 Matematika Diskrit

Oleh: Rinaldi Munir

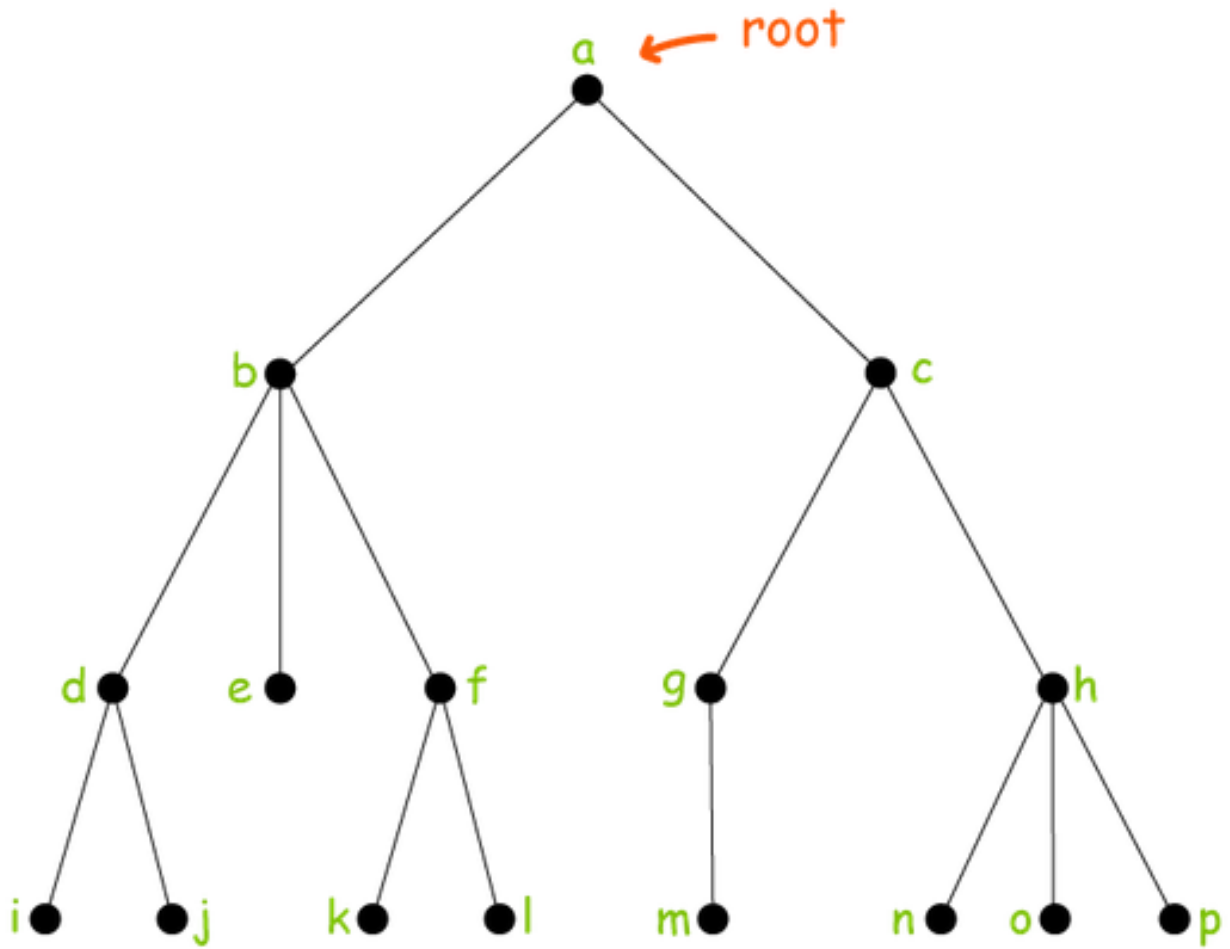
Program Studi Teknik Informatika
STEI- ITB

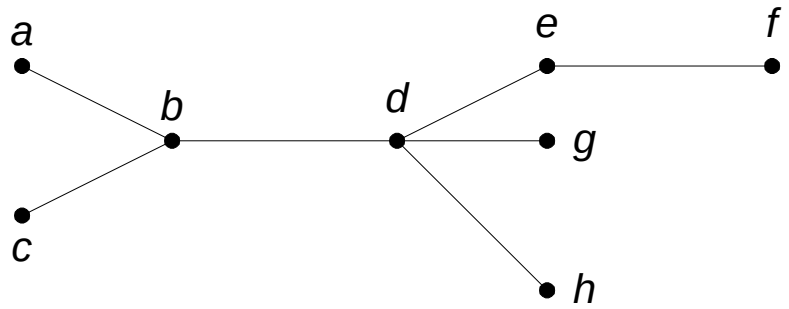
Pohon berakar (*rooted tree*)

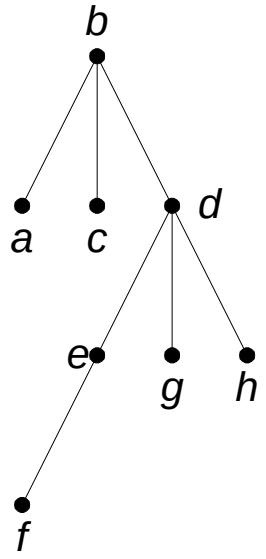
- Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan **pohon berakar** (*rooted tree*).



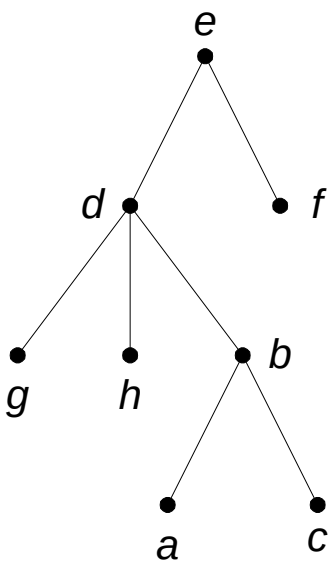
(a) Pohon berakar (b) sebagai perjanjian, tanda panah pada sisi dapat dibuang


root
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p





b sebagai akar



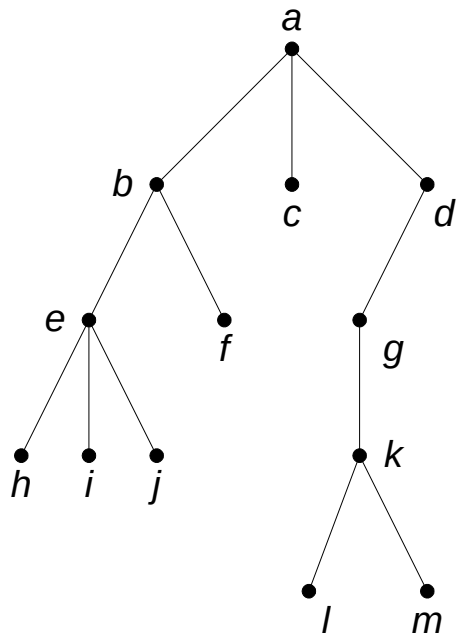
e sebagai akar

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



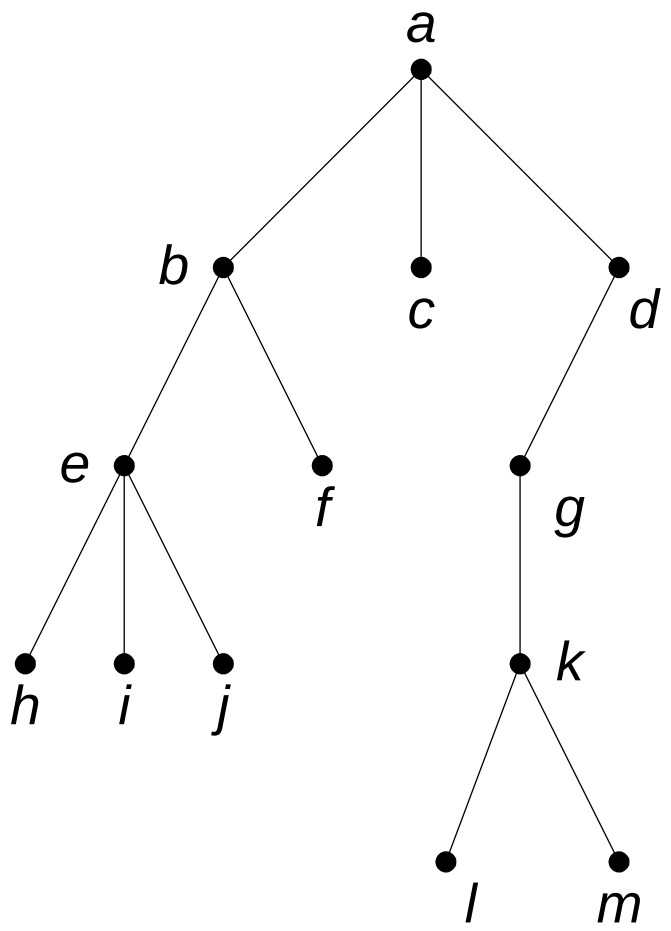
2. Lintasan (*path*)

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

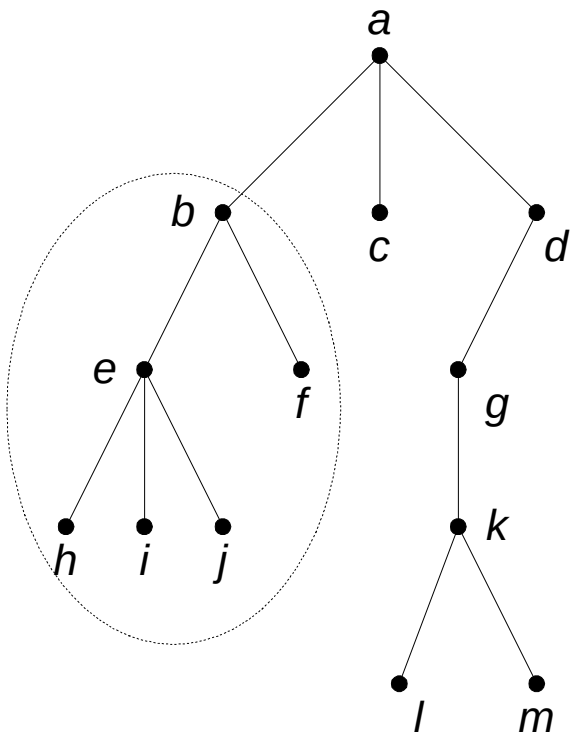
Panjang lintasan dari *a* ke *j* adalah 3.

3. Saudara kandung (*sibling*)

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



4. Upapohon (*subtree*)



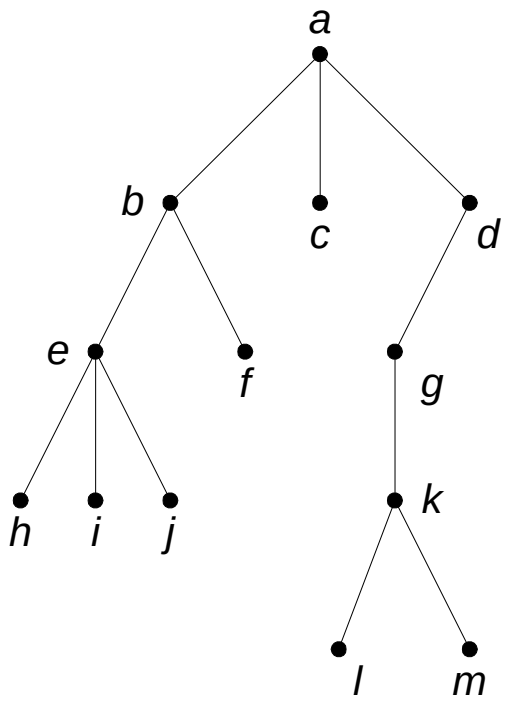
5. Derajat (*degree*)

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

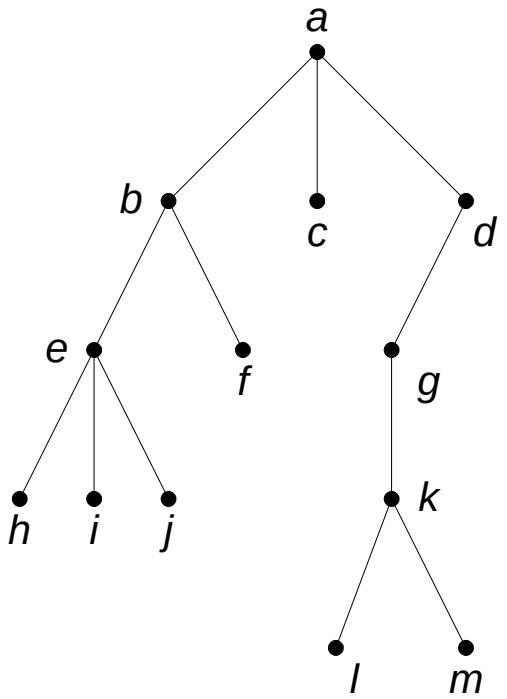


6. Daun (*leaf*)

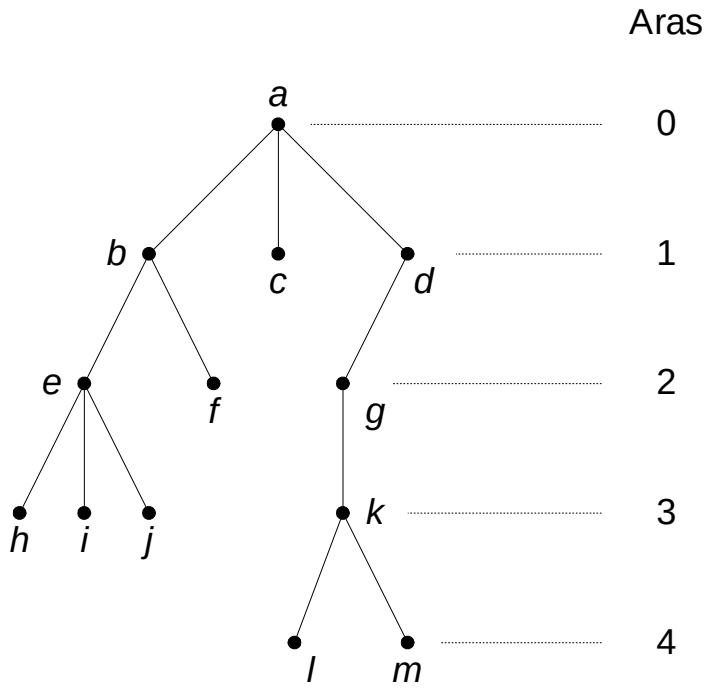
Simpul yang berderajat nol (atau tidak mempunyai anak) disebut **daun**. Simpul *h*, *i*, *j*, *f*, *c*, *l*, dan *m* adalah daun.

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*)

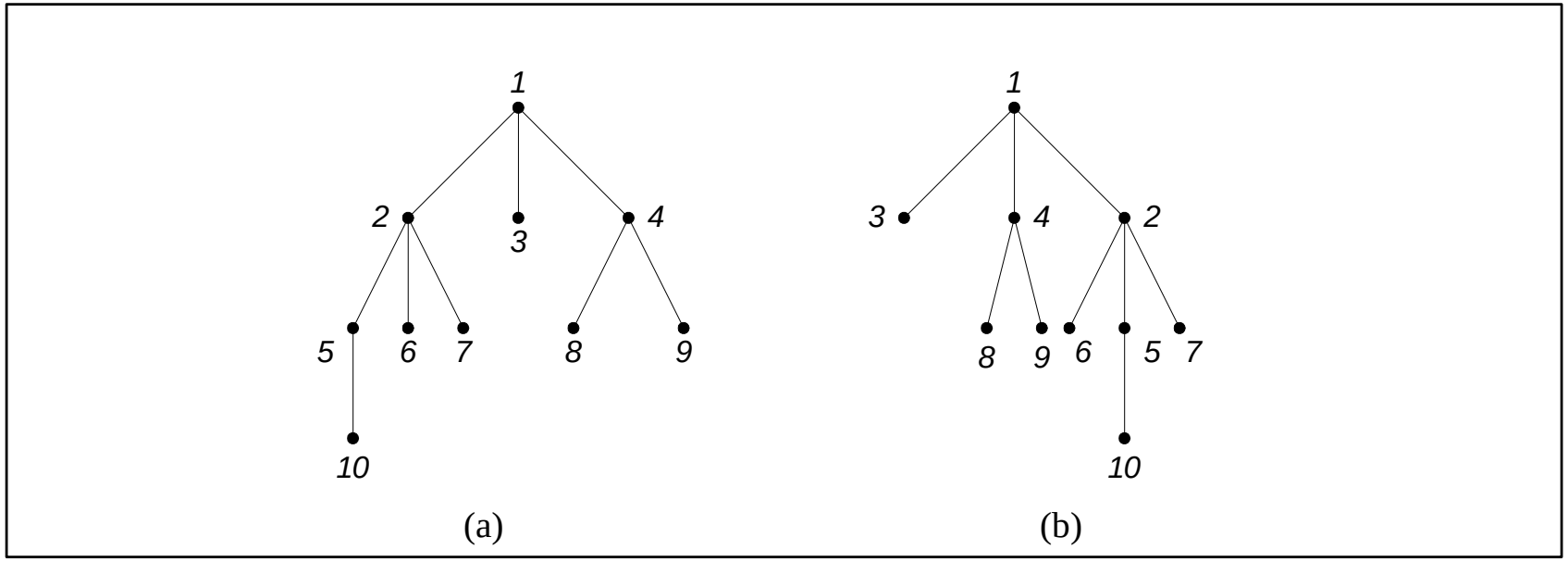
Aras maksimum dari suatu pohon disebut **tinggi** atau **kedalaman** pohon tersebut. Pohon di atas mempunyai tinggi 4.

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

*) Akarnya di atas

Pohon Terurut (*ordered tree*)

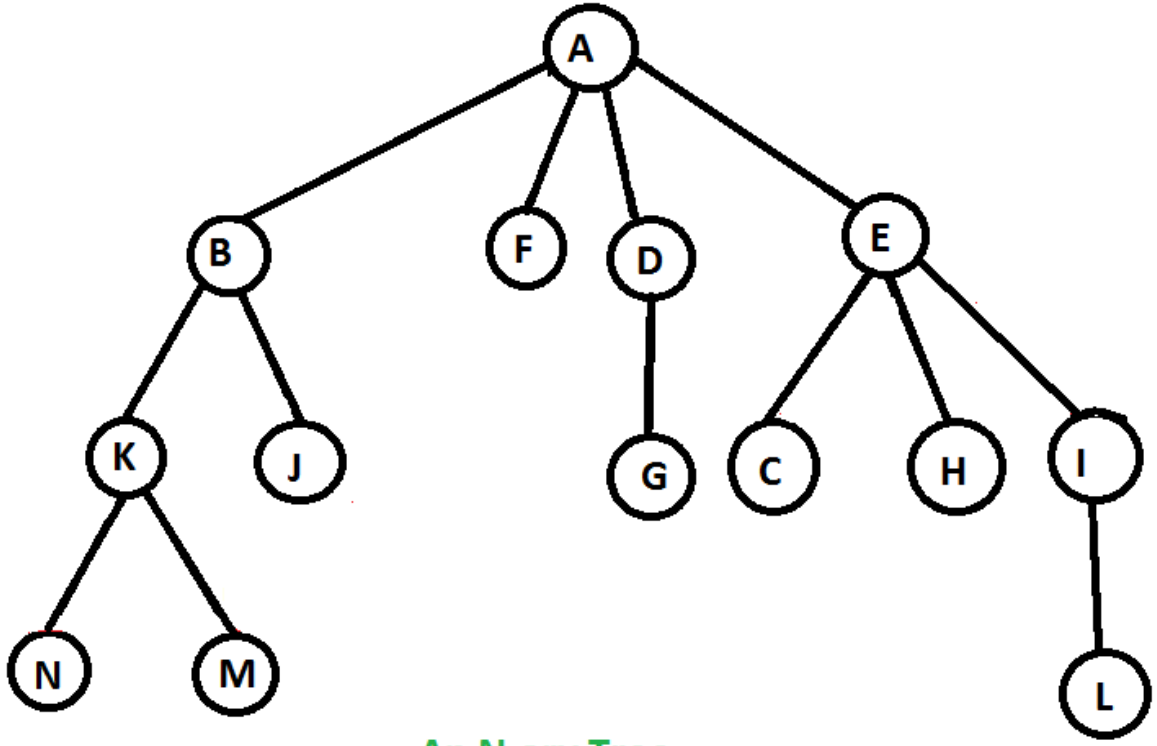
Pohon berakar yang urutan anak-anaknya penting disebut **pohon terurut** (*ordered tree*).



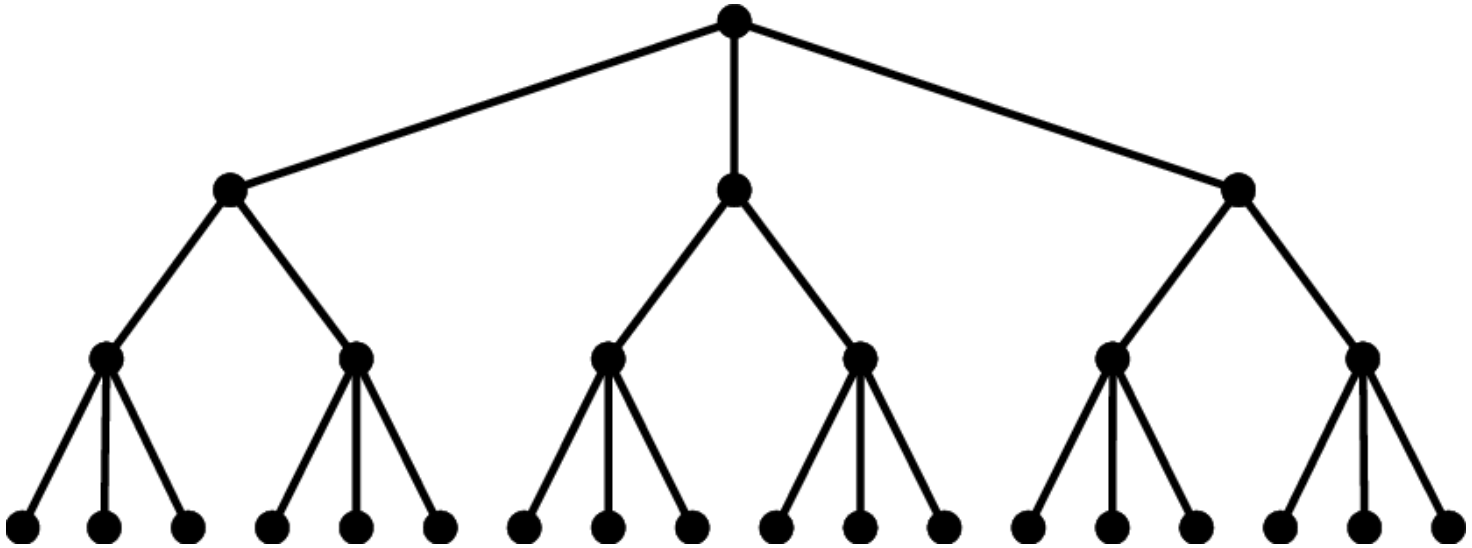
(a) dan (b) adalah dua pohon terurut yang berbeda

Pohon *n-ary*

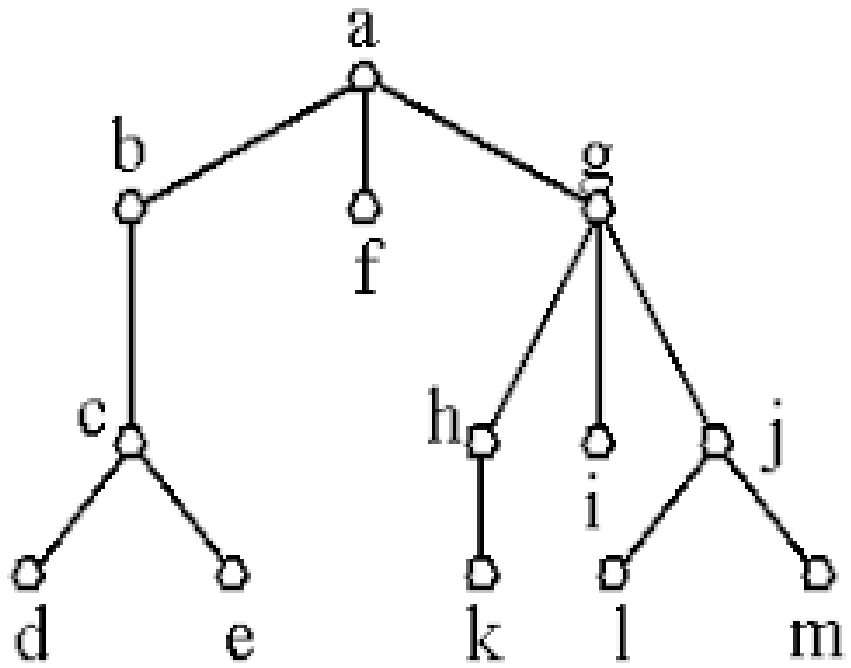
- Pohon berakar yang setiap simpul cabangnya mempunyai paling banyak *n* buah anak disebut **pohon *n-ary***.



An N-ary Tree.



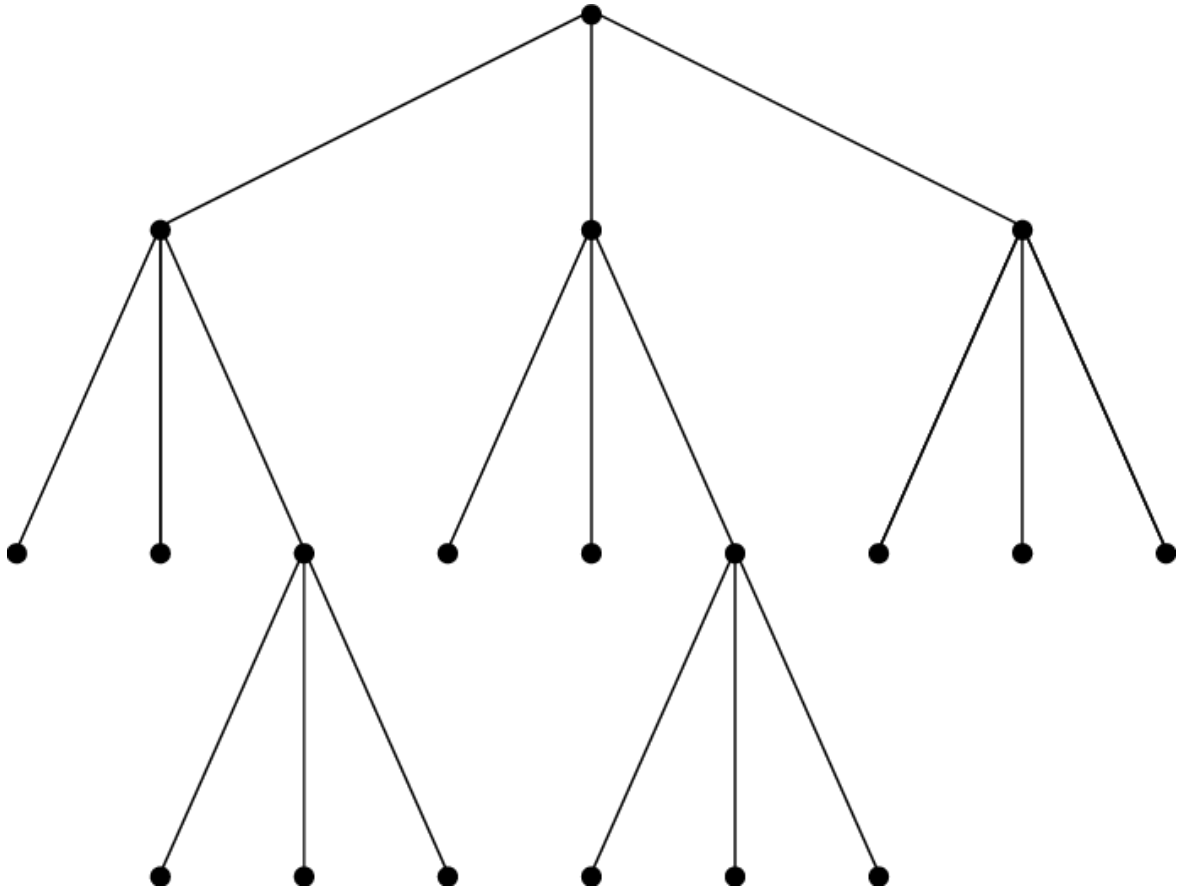
Pohon 3-ary

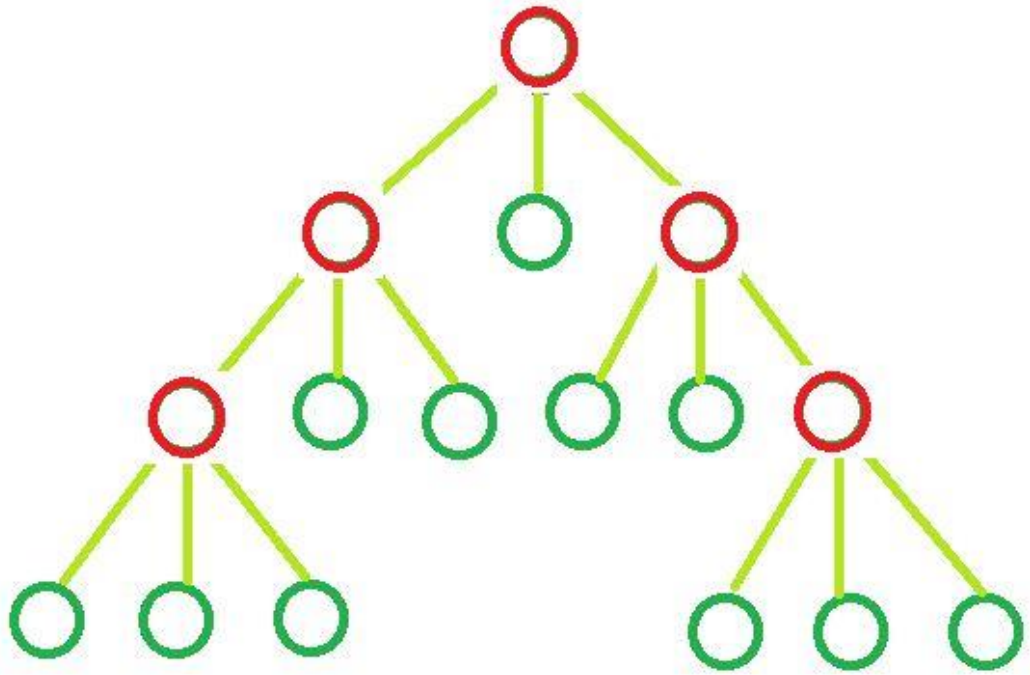


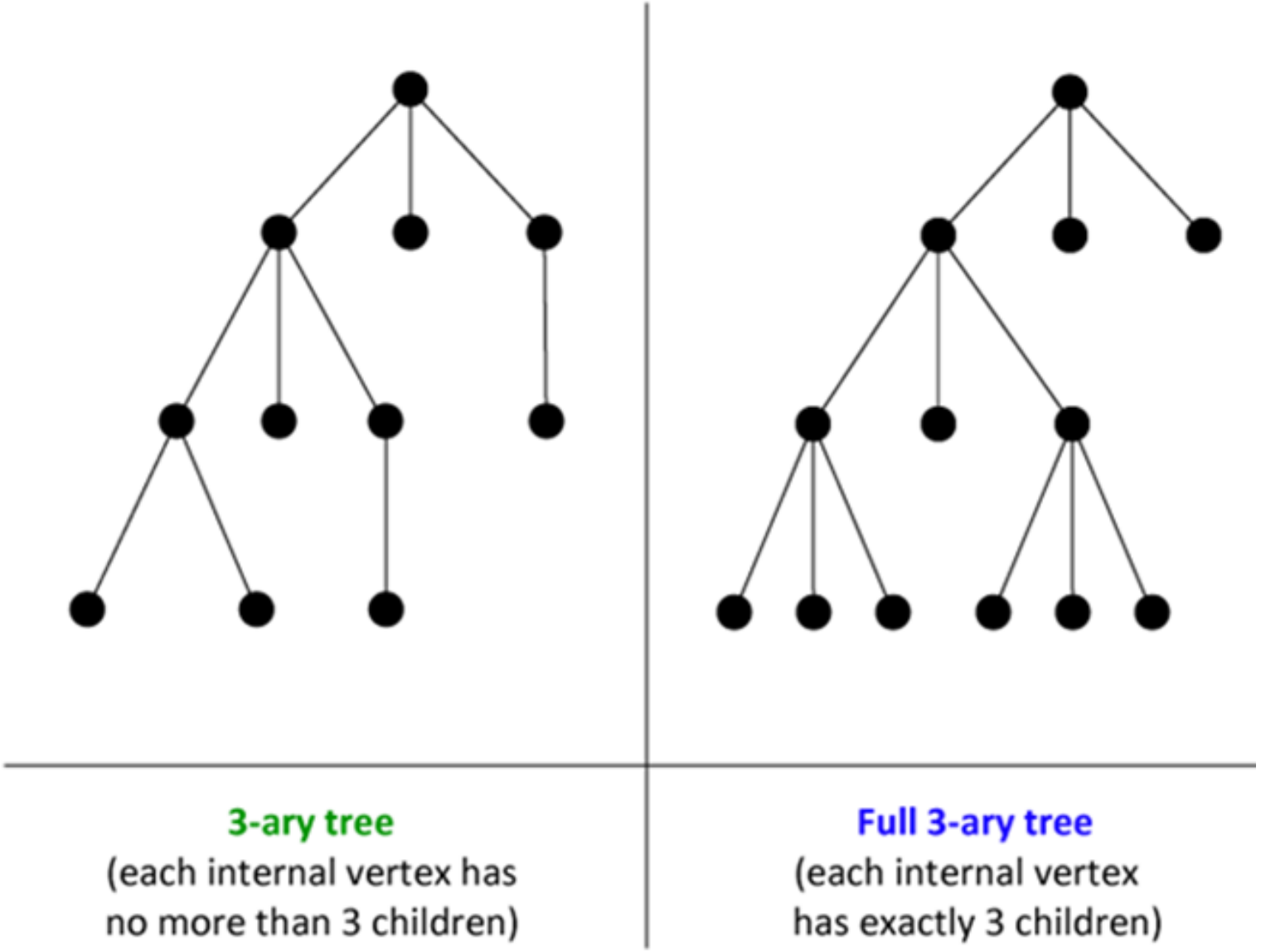
Pohon 3-ary

- Pohon *n-ary* dikatakan **teratur** atau **penuh** (*full*) jika setiap simpul, kecuali simpul pada aras daun, memiliki tepat *n* anak.

- Di bawah ini pohon 3-ary penuh:



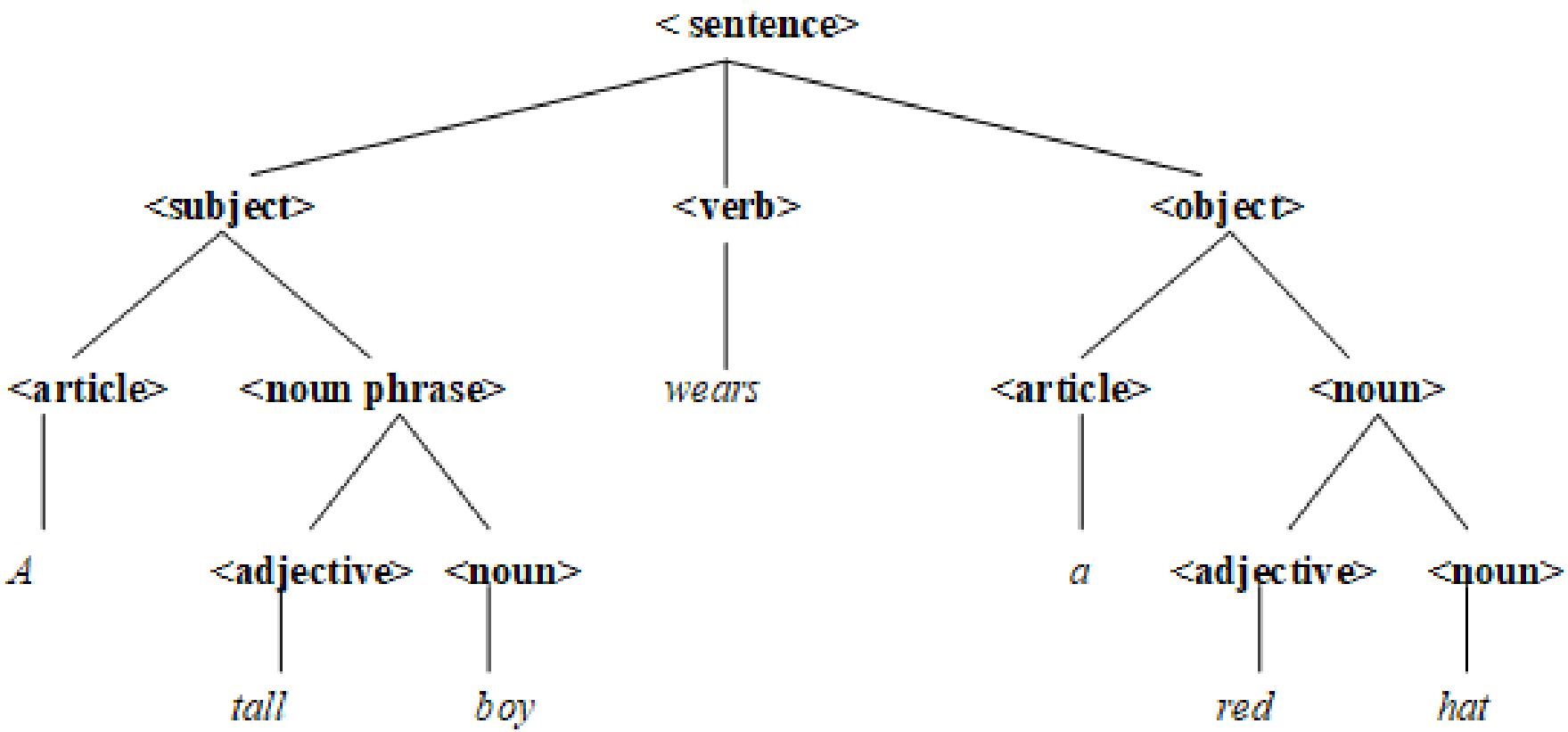




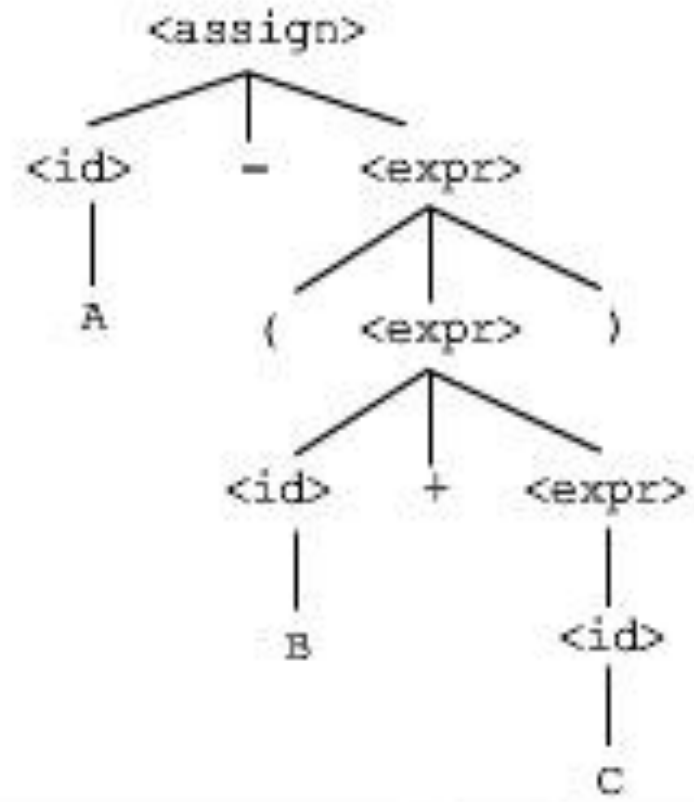
3-ary tree
(each internal vertex has
no more than 3 children)

Full 3-ary tree
(each internal vertex
has exactly 3 children)

Salah satu kegunaan pohon n-ary: *parsing* kalimat



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



Parse Tree for the Statement A = (B + C)

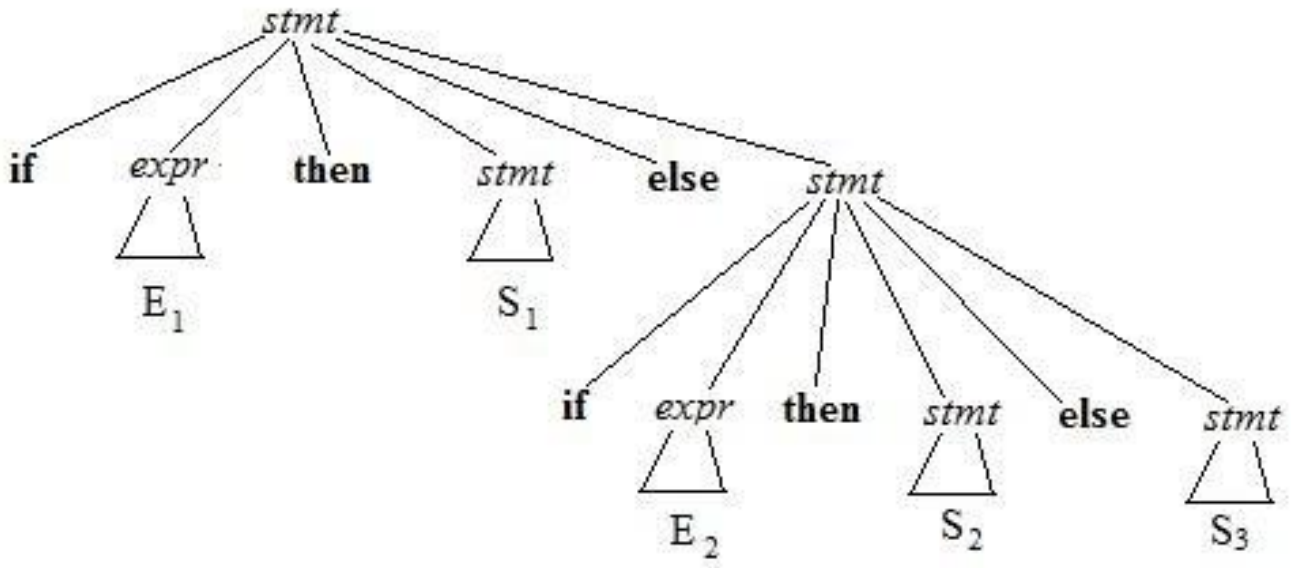
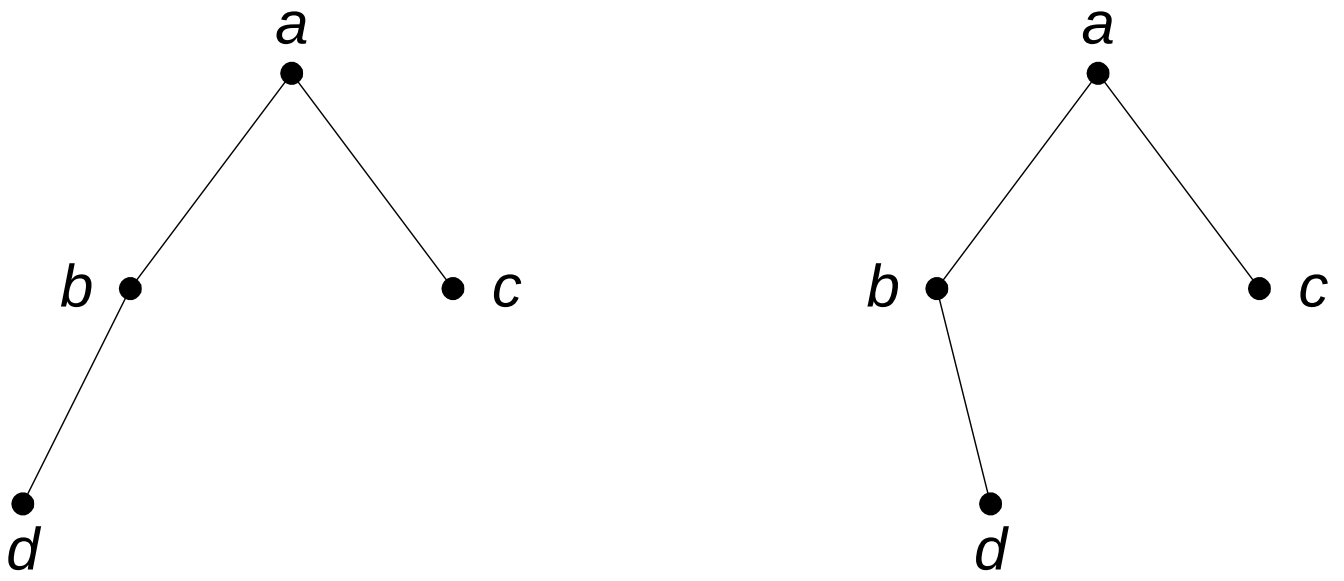


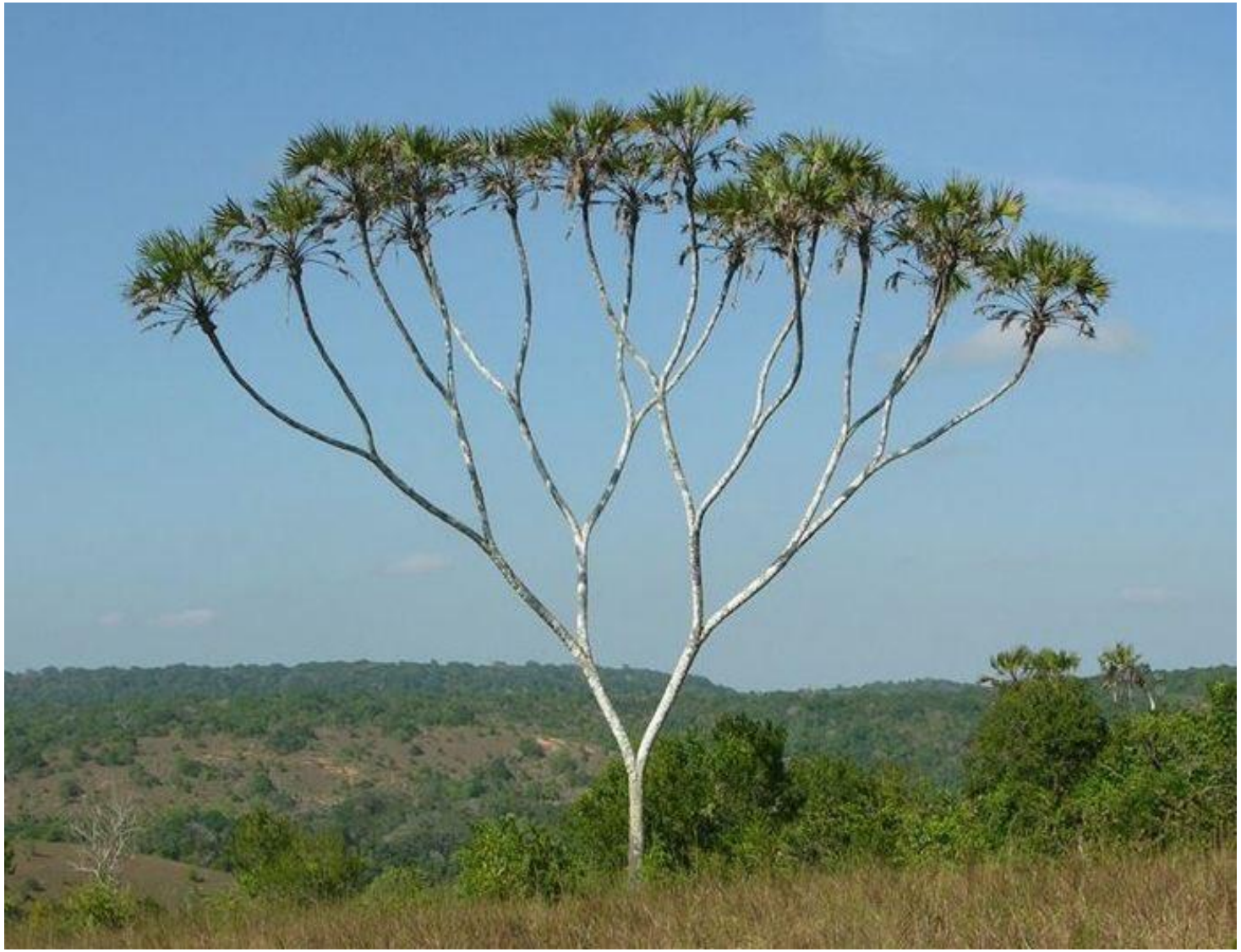
Fig 2.3 Parse tree for conditional statement

Pohon Biner (*binary tree*)

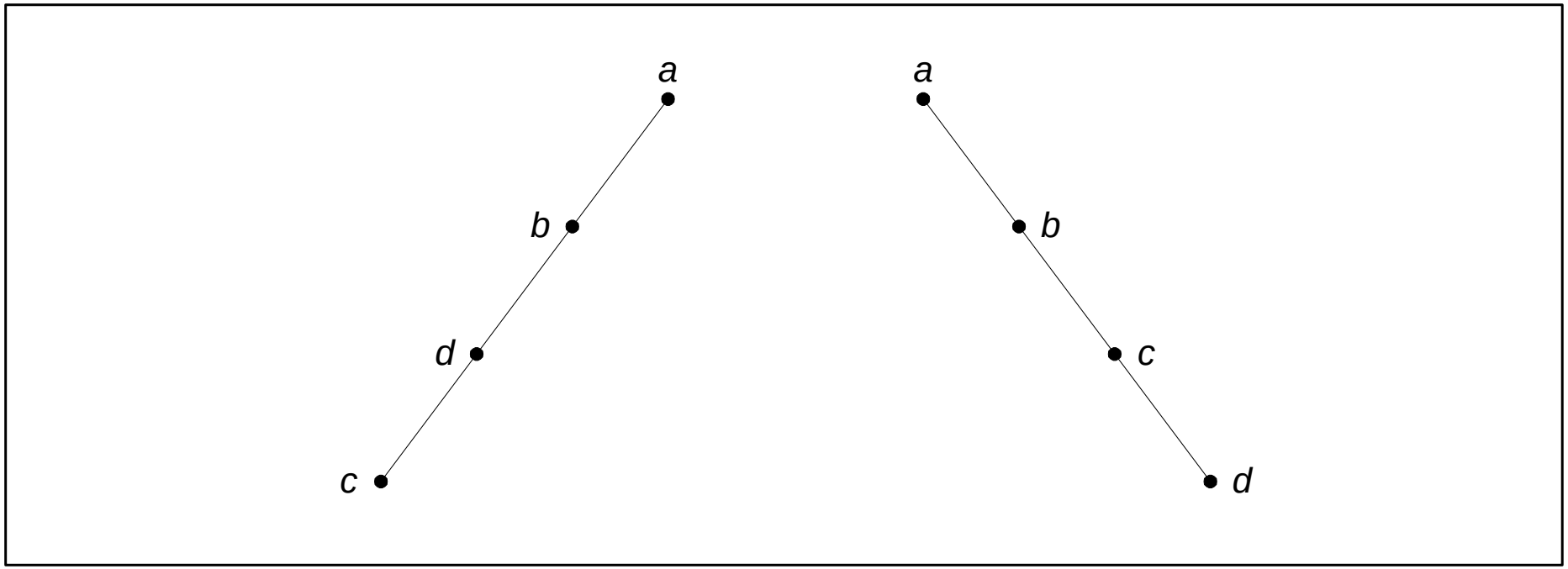
- Adalah pohon *n-ary* dengan *n* = 2.
- Pohon yang paling penting karena banyak aplikasinya.
- Setiap simpul di adlam pohon biner mempunyai paling banyak 2 buah anak.
- Dibedakan antara anak kiri (*left child*) dan anak kanan (*right child*)
- Karena ada perbedaan urutan anak, maka pohon biner adalah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



Gambar Dua buah pohon biner yang berbeda

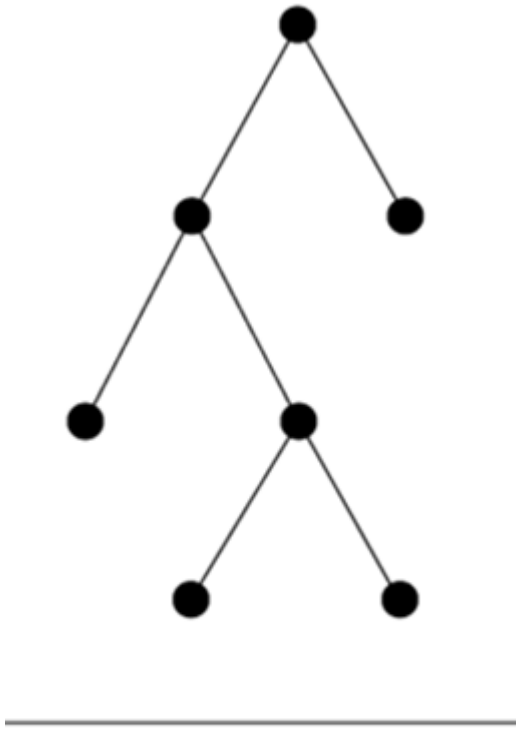


Real binary tree

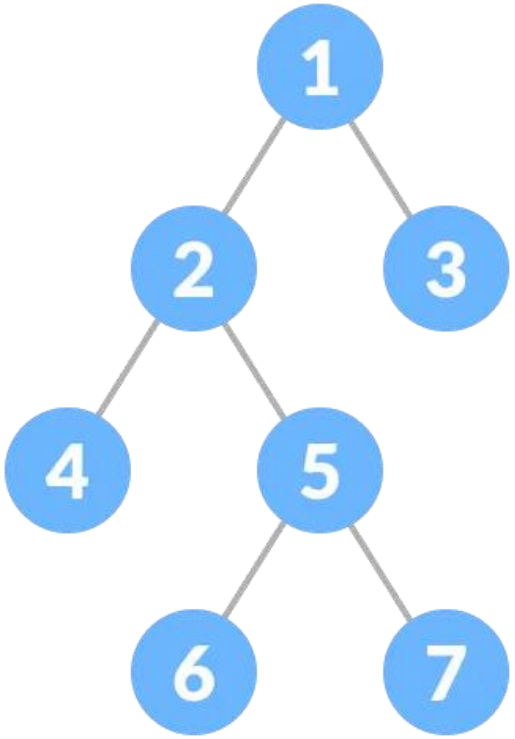


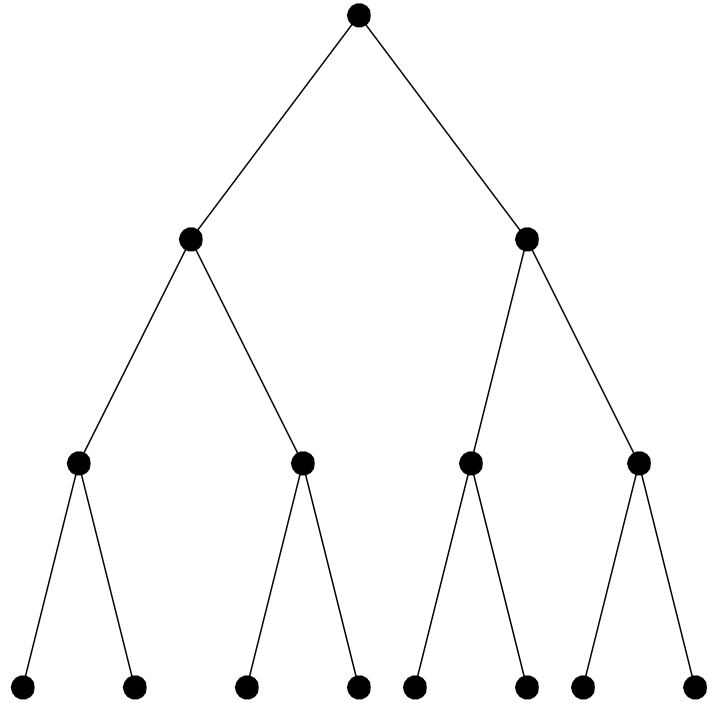
Gambar (a) Pohon condong-kiri, dan (b) pohon condong kanan

Full binary tree: setiap simpul, kecuali simpul pada aras daun, memiliki tepat dua anak

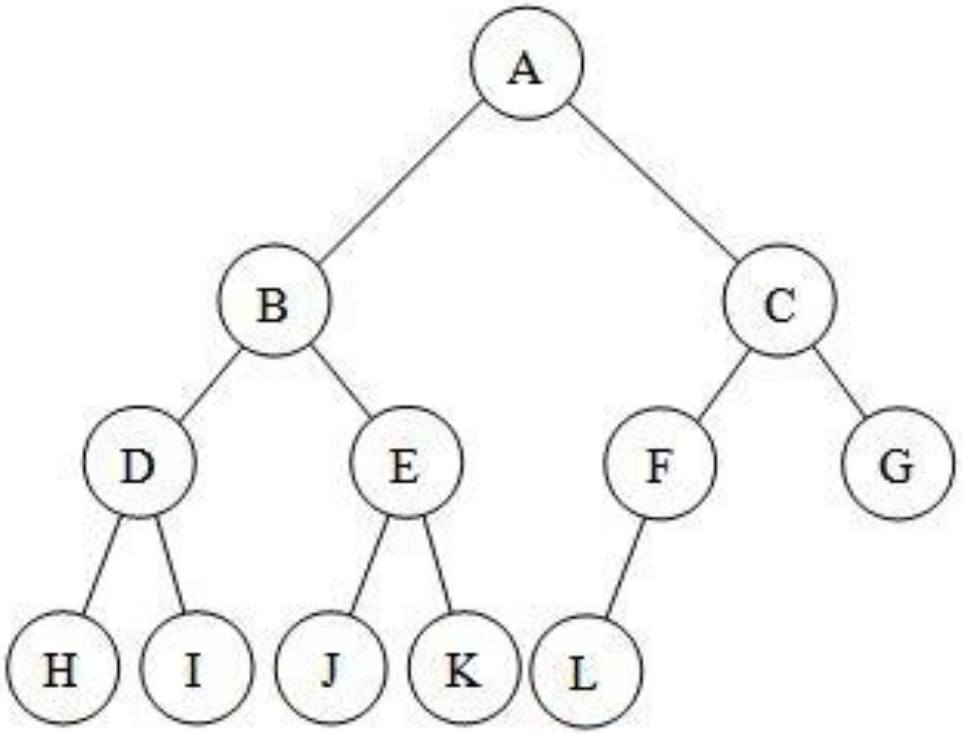


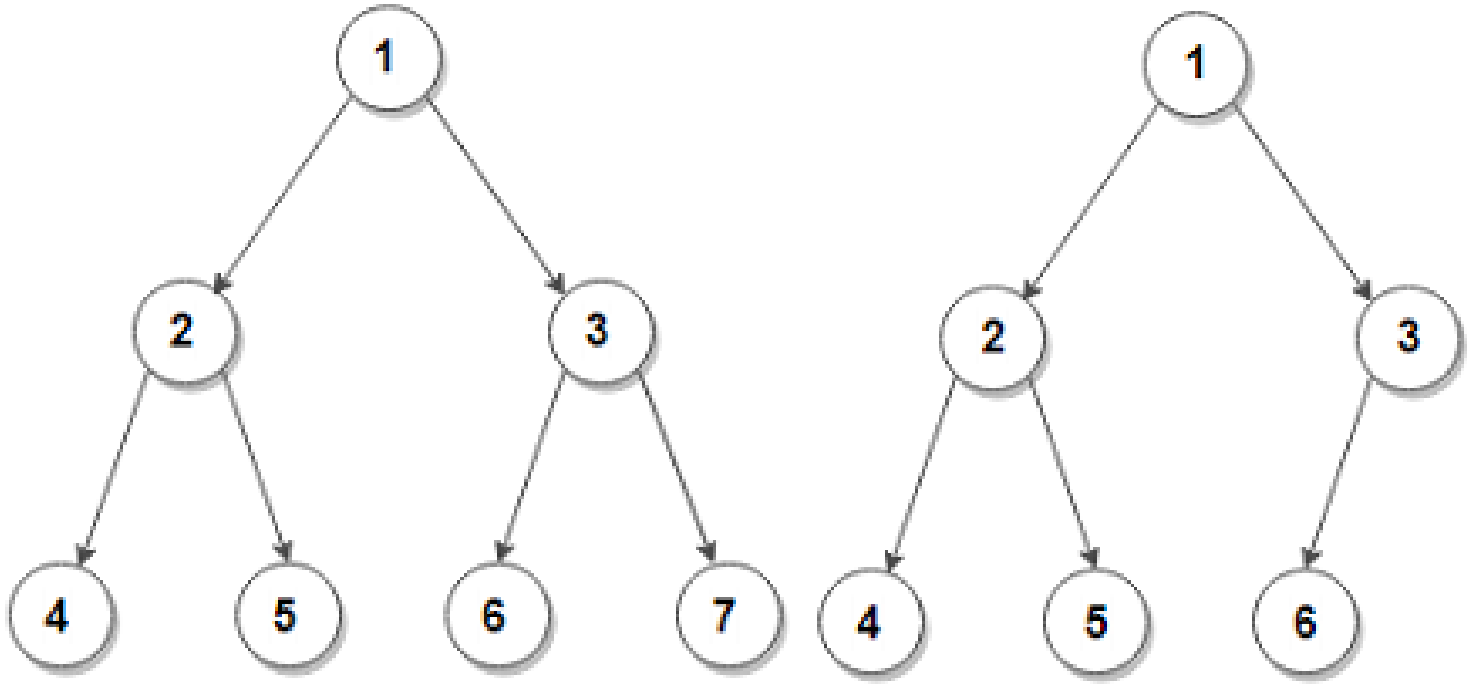
Full Binary tree
(each internal node
has exactly 2 children)



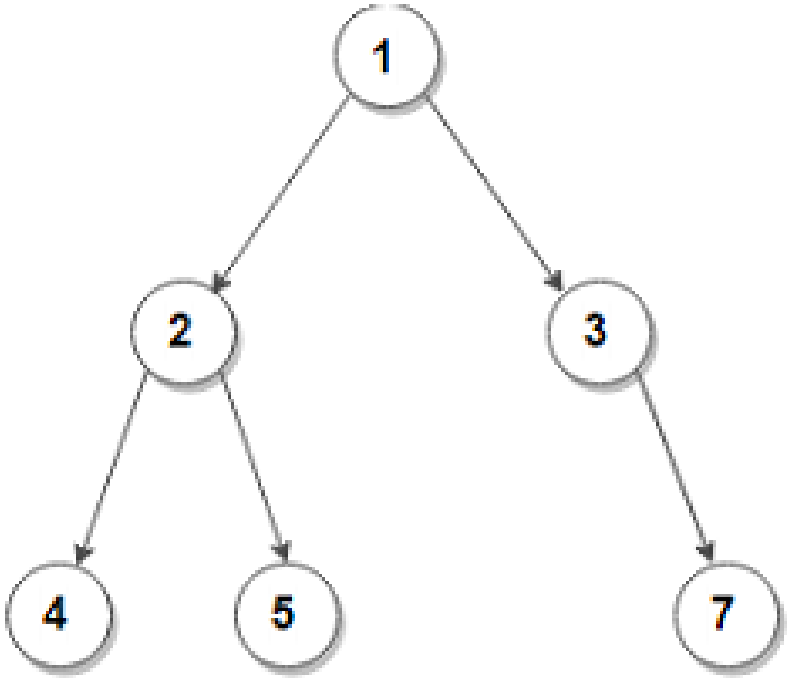


Complete binary tree: setiap aras, mungkin kecuali pada aras terakhir, terisi lengkap, dan semua simpul dipadatkan sejauh mungkin ke bagian kiri





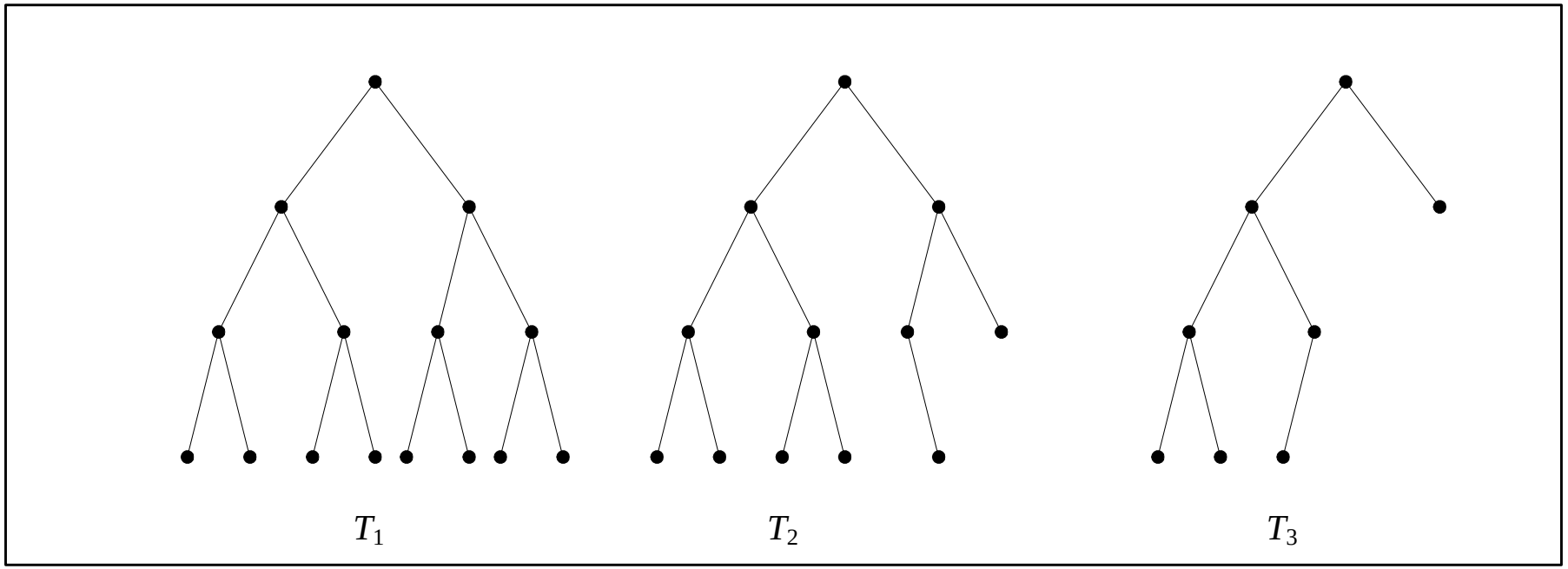
Complete Binary Tree



Not a Complete Binary Tree

Pohon Biner Seimbang

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

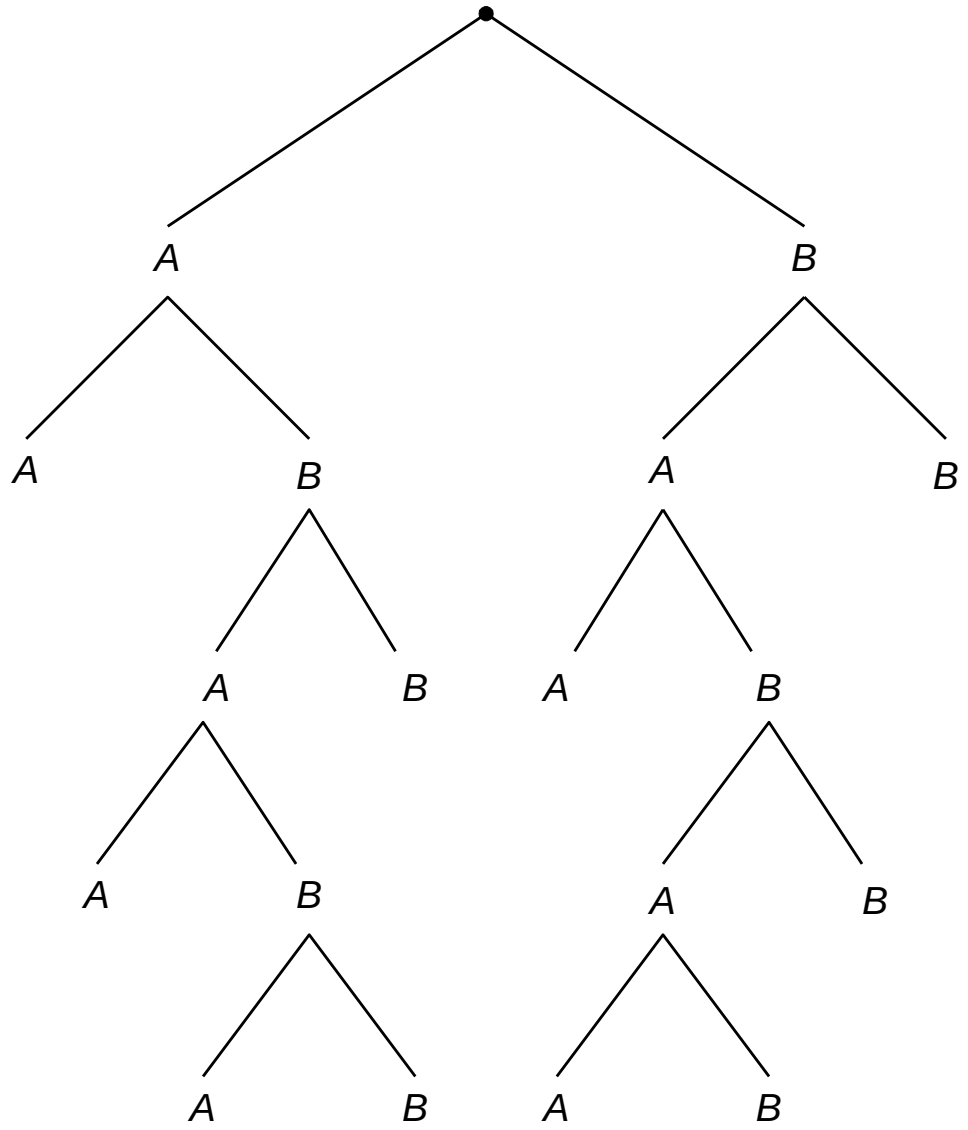


Gambar T_1 dan T_2 adalah pohon seimbang, sedangkan T_3 bukan pohon seimbang.

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:



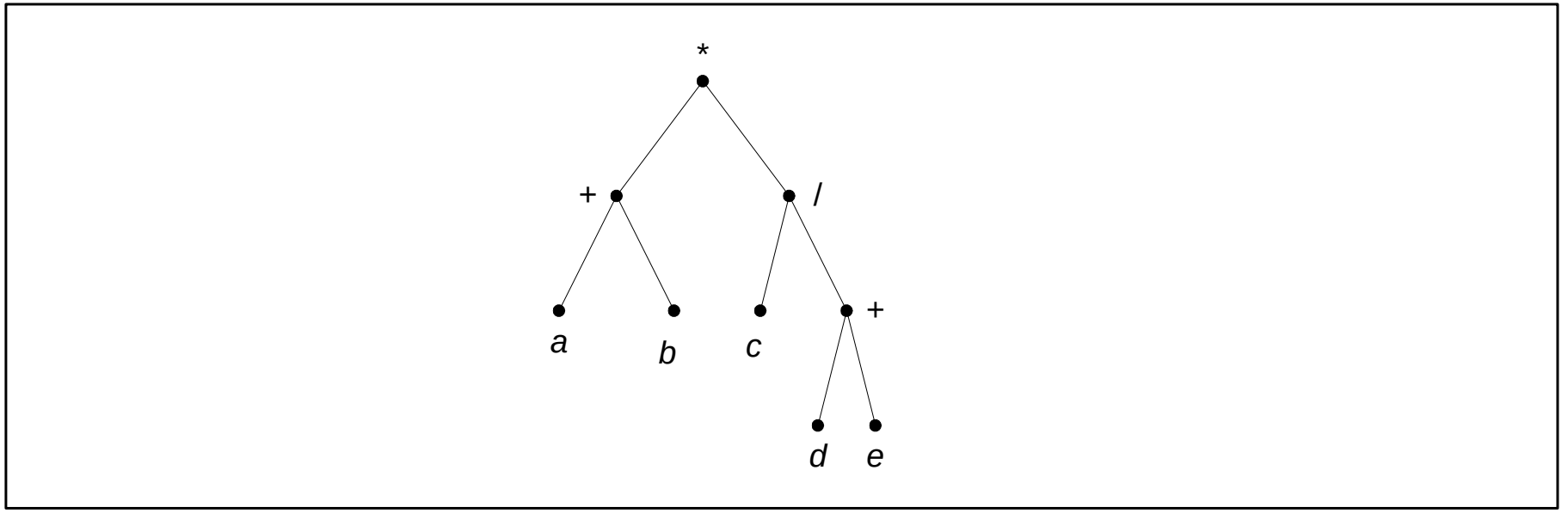
Latihan

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

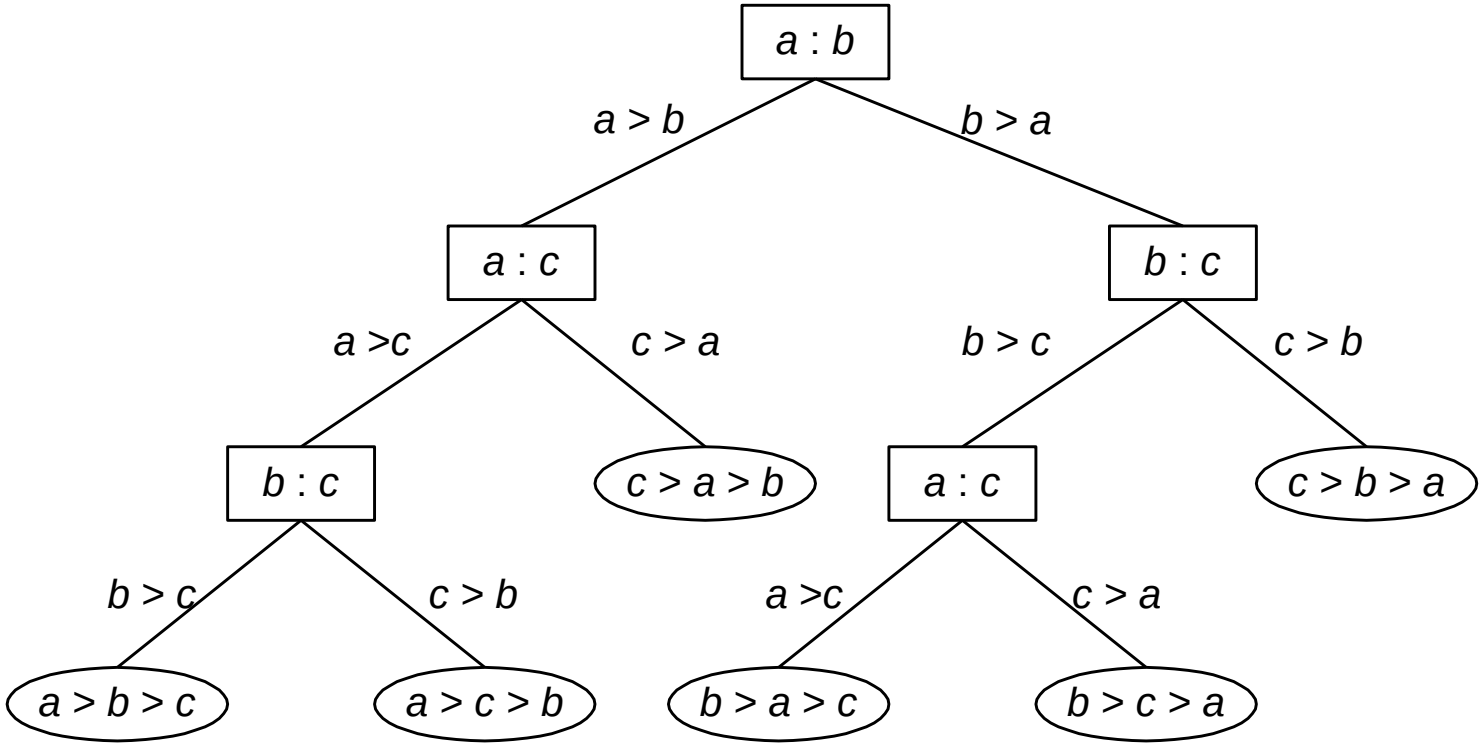
1. Pohon Ekspresi



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

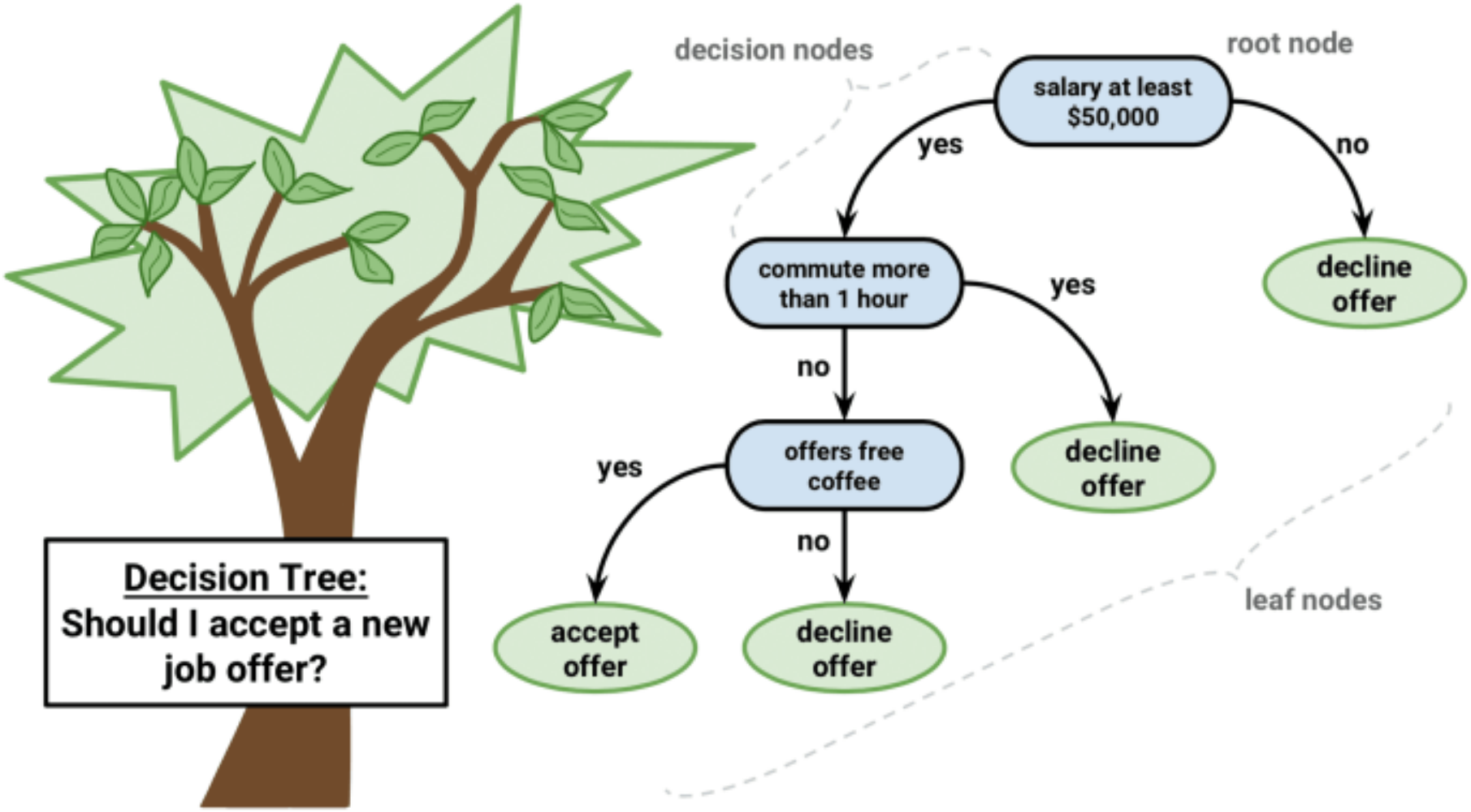
daun → *operand*
simpul dalam → operator

2. Pohon Keputusan

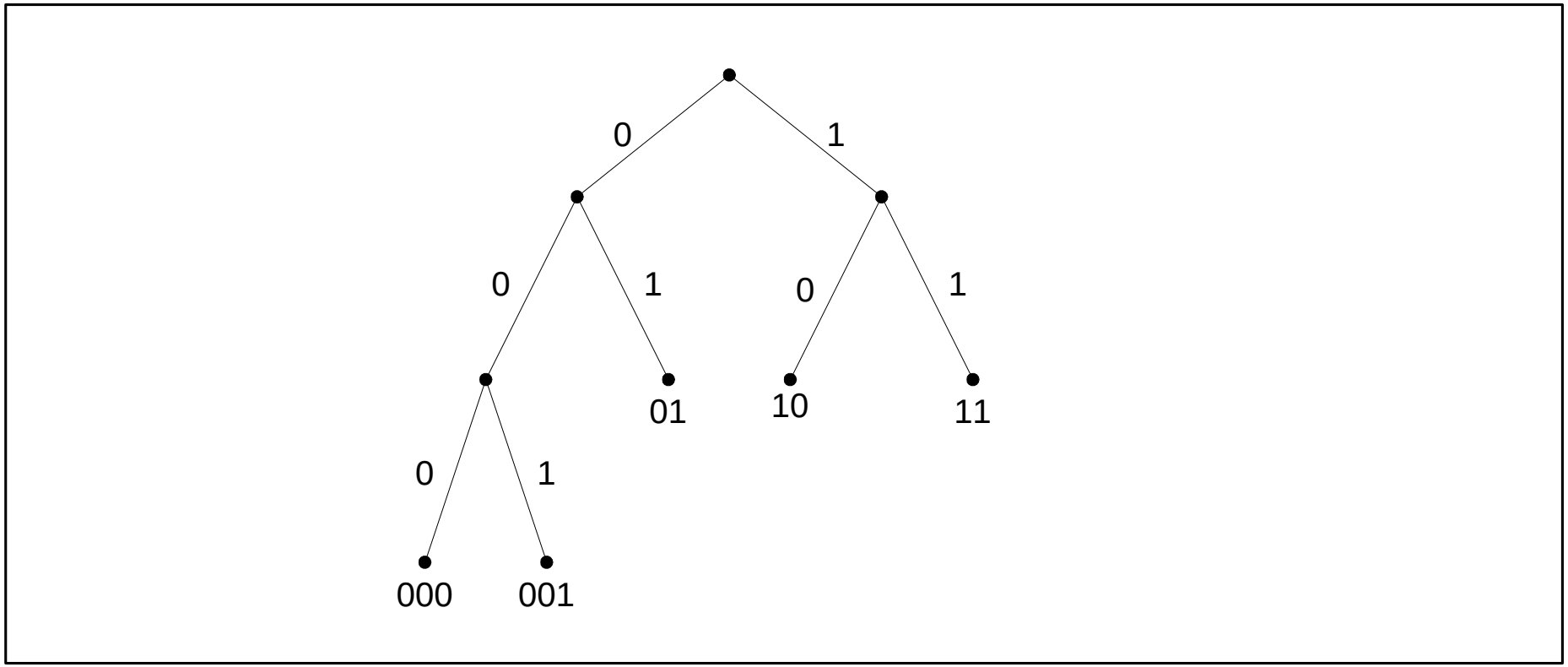


Gambar Pohon keputusan untuk mengurutkan 3 buah elemen

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
A	01000001
B	01000010
C	01000011
D	01000100

rangkaian bit untuk string '*ABACCDA*':

01000001010000010010000010100000110100000110100010001000001

atau 7 × 8 = 56 bit (7 *byte*).

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

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

Dengan kode Hufman, rangkaian bit untuk *'ABACCDA'*:

0110010101110

hanya 13 bit!

Bagaimana cara memperoleh kode Huffman?

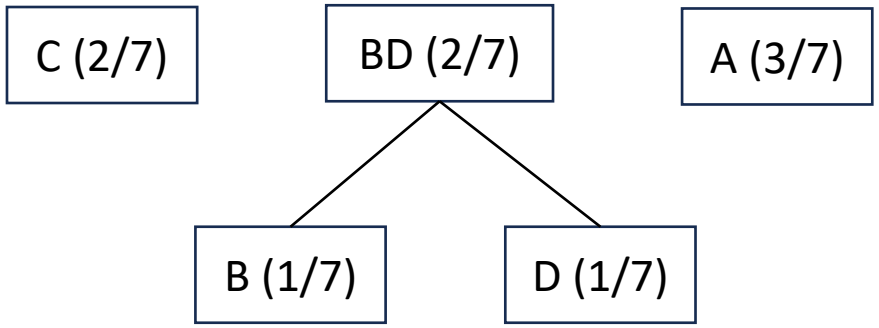
Algoritma Kompresi Huffman

Algoritma pembentukan kode Huffman

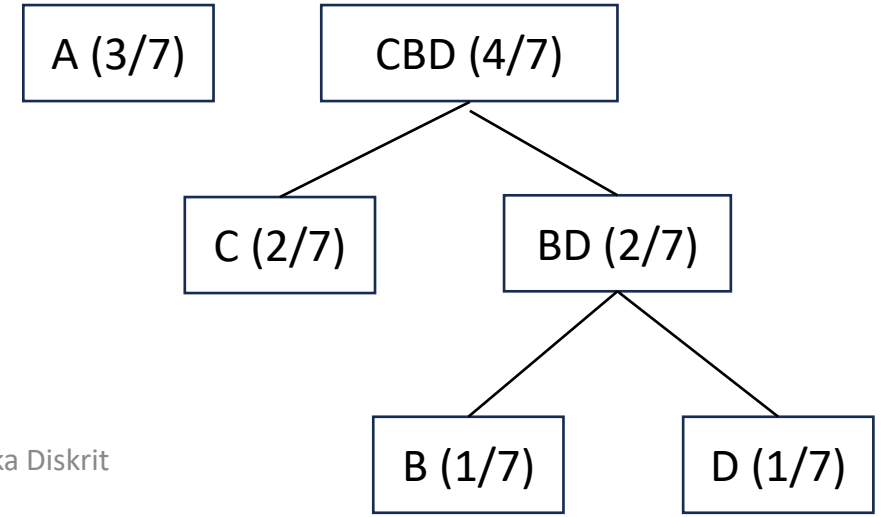
1. Urutkan simbol berdasarkan peluangnya (atau jumlah kemunculan) dari kecil ke besar
2. Pilih dua simbol dengan peluang paling kecil (Pada contoh di atas simbol *B* dan *D*). Kedua simbol tadi dikombinasikan sebagai simpul orangtua. Simbol orangtua adalah gabungan (*concatenation*) dari kedua simbol tersebut. Peluang simbol hasil penggabungan adalah jumlah peluang dari kedua simbol (Pada contoh di atas simbol *B* dan *D* digabung menjadi simbol *BD* dengan peluang 1/7 + 1/7 = 2/7).
3. Selanjutnya, pilih dua simbol berikutnya, termasuk simbol baru, yang mempunyai peluang terkecil.
4. Ulangi langkah 1 dan 2 sampai seluruh simbol habis.
5. Beri label secara konsisten sisi kiri dengan 0 dan sisi kanan dengan 1.
6. Lintasan dari akar ke daun berisi sisi-sisi pohon yang deretan labelnya menyatakan kode Huffman untuk simbol daun tersebut

1. B (1/7) D (1/7) C (2/7) A (3/7) (terurut membesar)

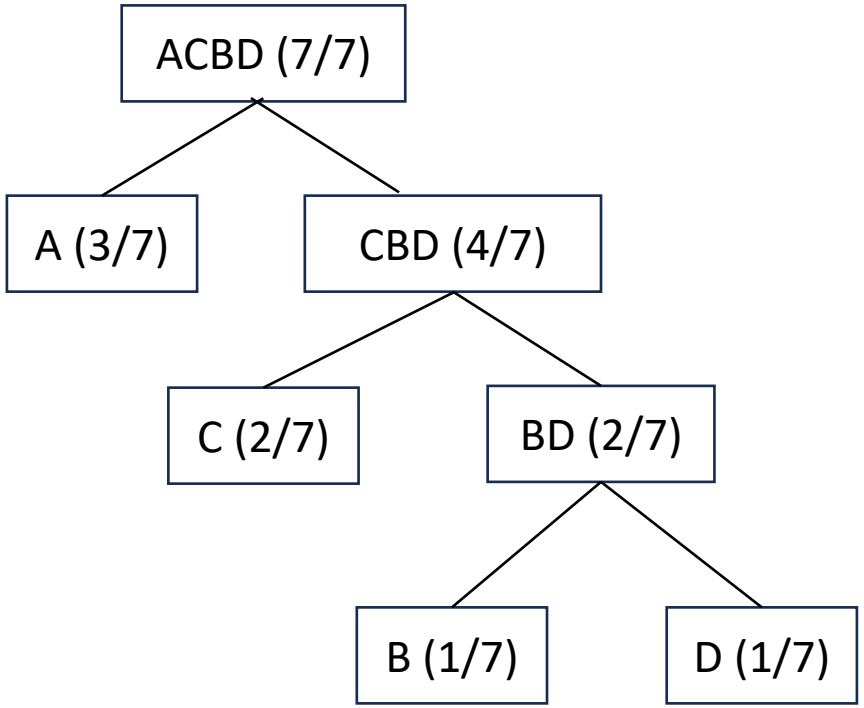
2. Gabungkan B dan D menjadi BD, peluang BD = 1/7 + 1/7 = 2/7, tempatkan simbol baru BD di dalam urutan yang terurut



3. Gabungkan C dan BD menjadi CBD, peluang CBD = 2/7 + 2/7 = 4/7, tempatkan simbol baru CBD di dalam urutan yang terurut

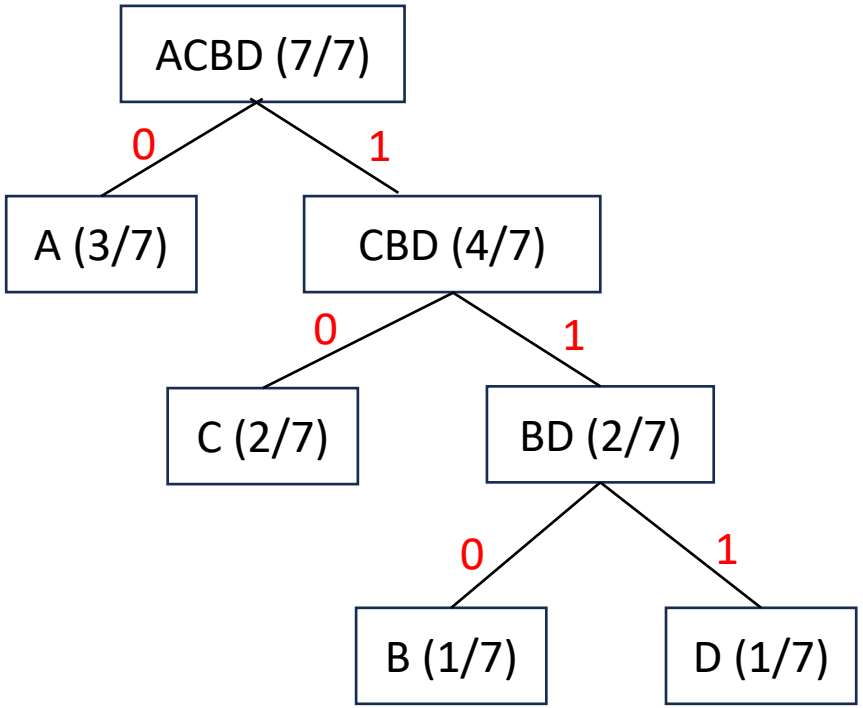


3. Gabungkan A dan CBD menjadi ACBD, peluang CCBD = 3/7 + 4/7 = 7/7



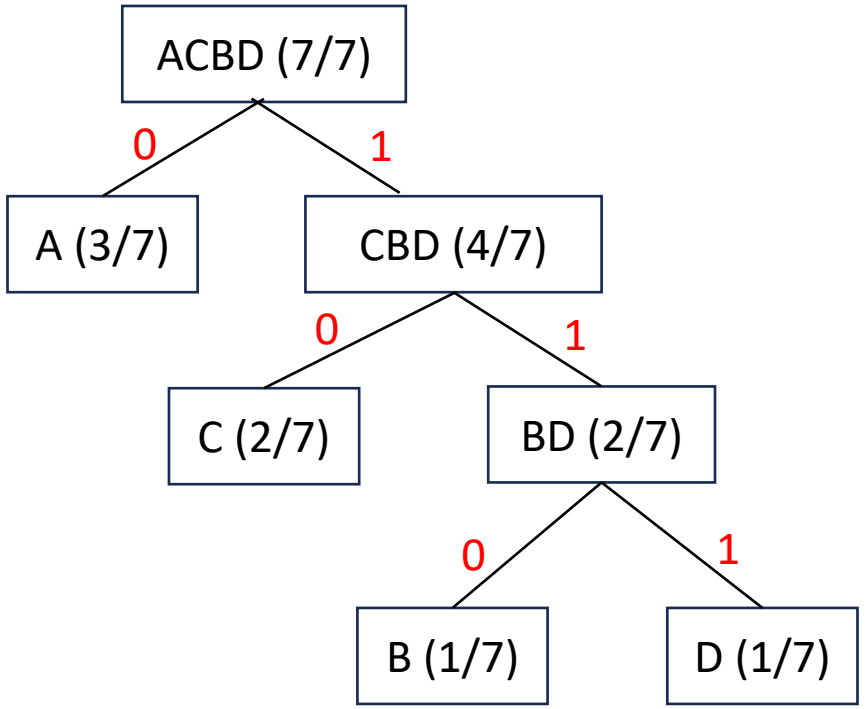
4. Simbol sudah habis. Stop. Pohon Huffman sudah terbentuk. Selanjutnya ke Langkah 5.

5. Sisi-sisi kiri di dalam pohon Huffam diberi label 0, sisi-sisi kanan diberi label 1.



6. Telusuri pohon dari akar ke daun. Lintasan dari akar ke daun berisi rangakaian bit yang menyatakan kode Huffman untuk setiap simbol di dalam daun.

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



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

Encoding:

- Kodekan setiap simbol di dalam pesan tadi dengan kode Huffman:

- Contoh: ABACCDA = 0110010101110

Decoding:

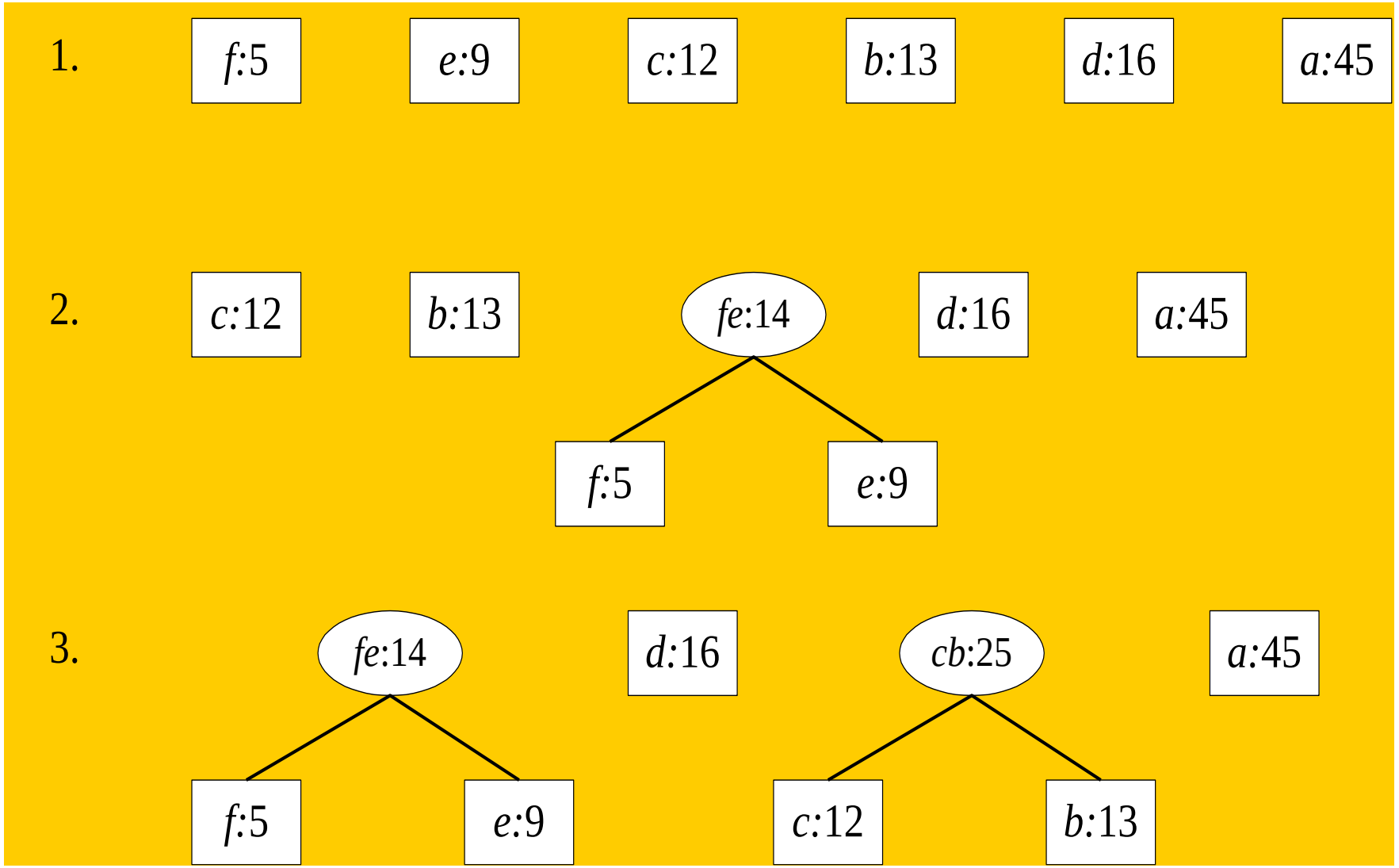
1. Baca simbol biner pertama
2. Mulai traversal dari akar mengikuti sisi yang sesuai dengan simbol biner tersebut
3. Baca terus simbol biner dan traversal sisi yang bersesuaian sampai ketemu daun. Kodekan barisan biner yang sudah dibaca dengan simbol daun
4. Baca simbol biner berikutnya dan ulangi traversal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

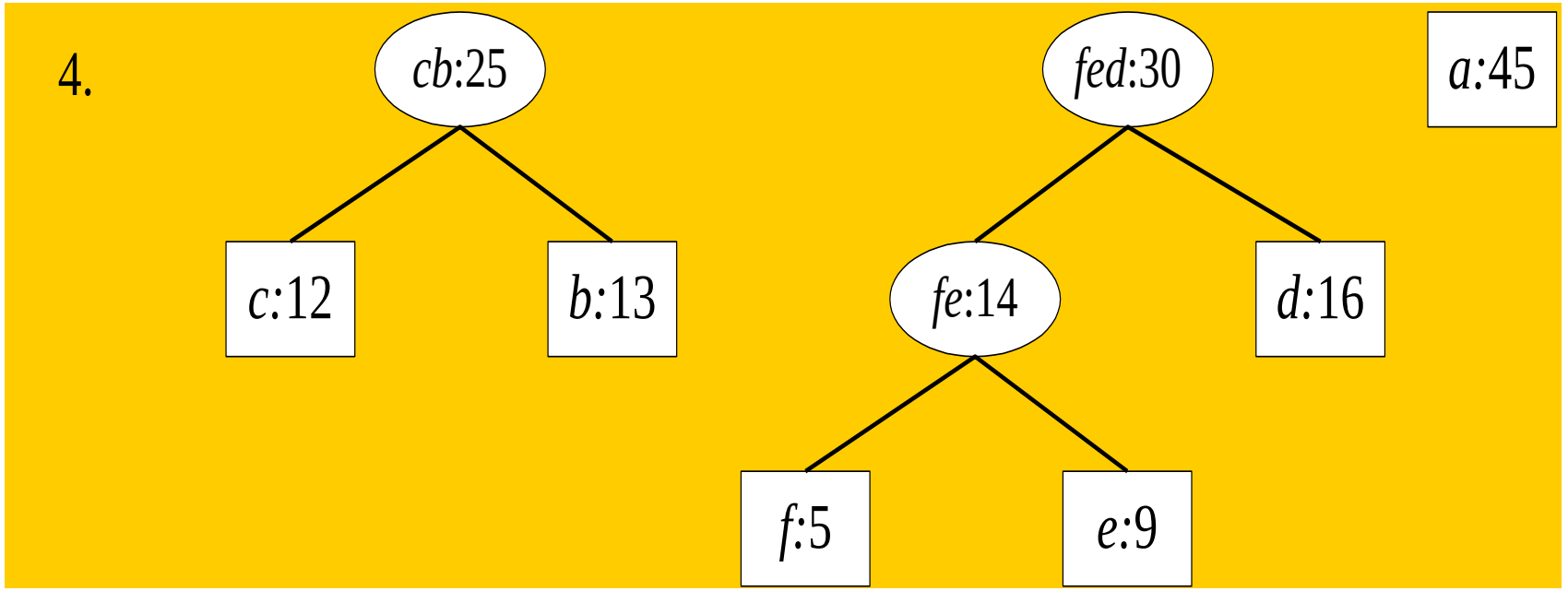
Contoh 2: Sebuah pesan sepanjang 100 karakter disusun oleh huruf a, b, c, d, e, dan f. Frekuensi kemunculan setiap huruf di dalam pesan adalah sebagai berikut:

Karakter	*a*	*b*	*c*	*d*	*e*	*f*
Frekuensi	45	13	12	16	9	5

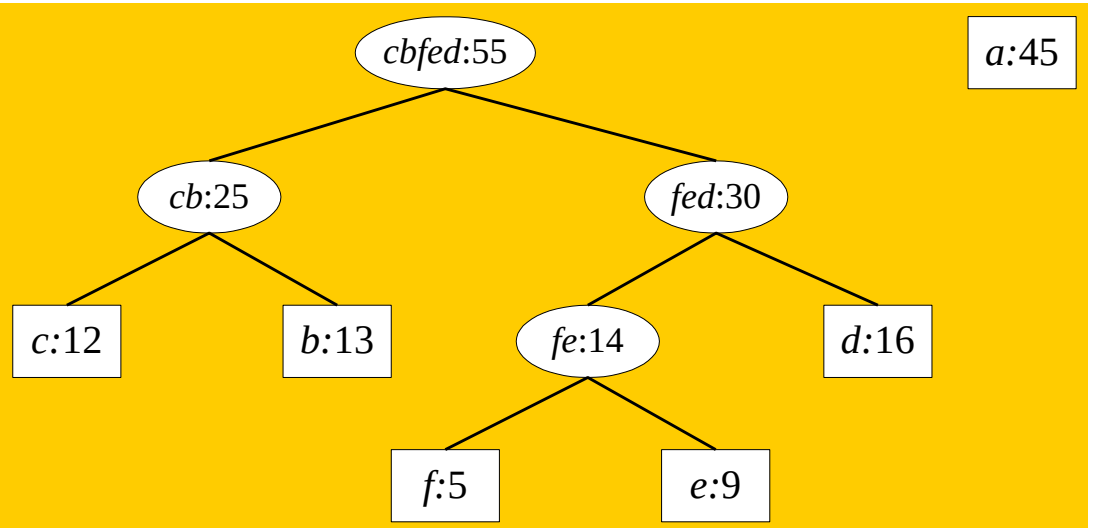
Tentukan kode Huffman untuk setiap huruf tersebut!

Penyelesaian:

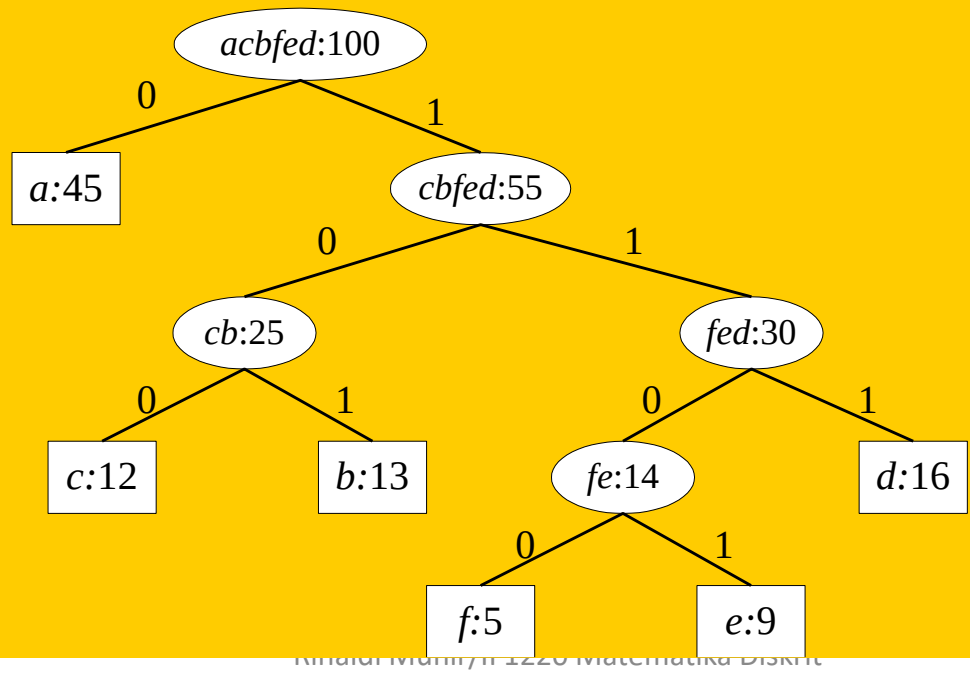



4.
cb:25
fed:30
a:45
c:12
b:13
fe:14
d:16
f:5
e:9

5.



6



Kode Huffman:

a = 0

b = 101

c = 100

d = 111

e = 1101

f = 1100

Latihan (Kuis 2021)

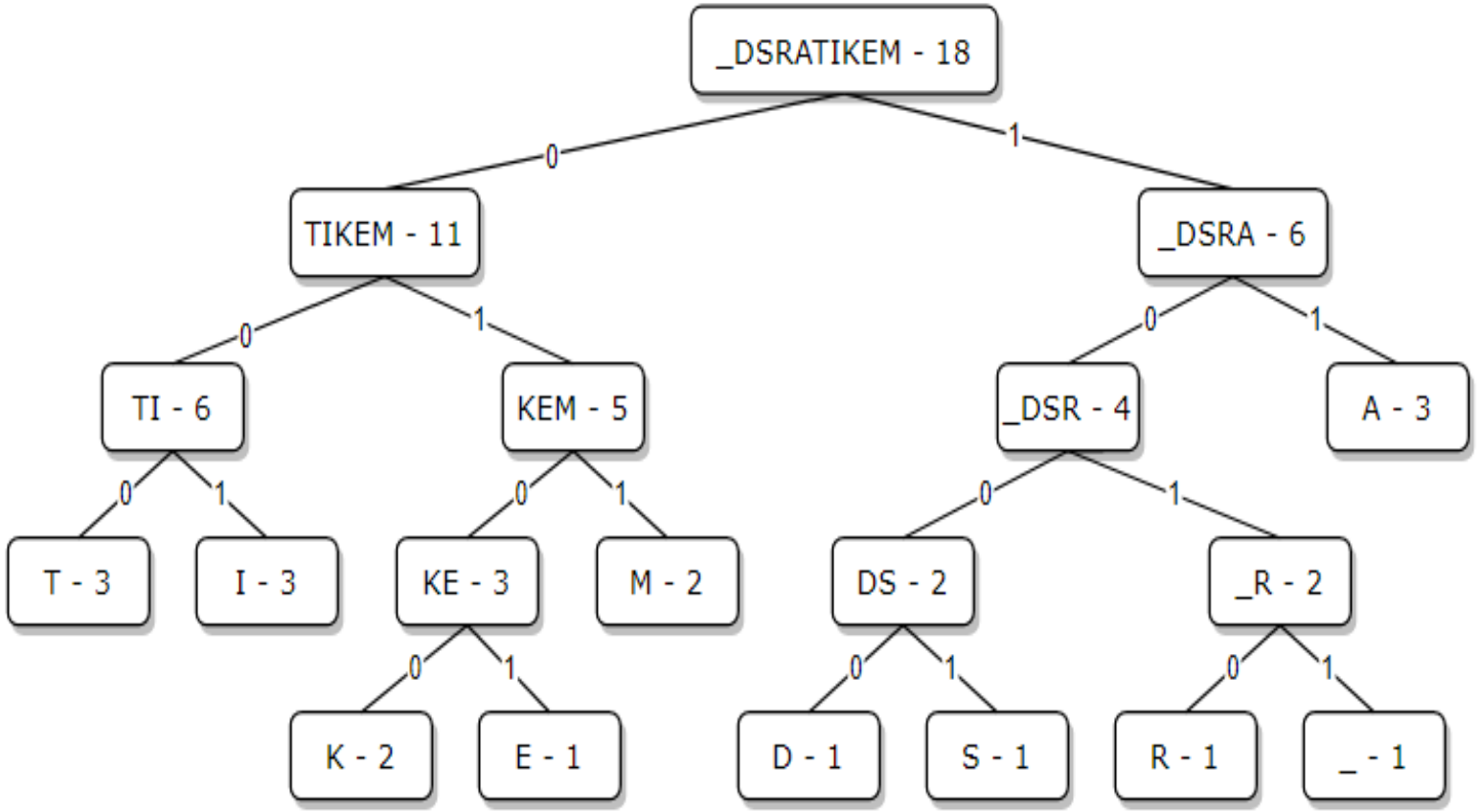
Terdapat sebuah pesan “MATEMATIKA DISKRIT” dalam sebuah *script*. Berapakah panjang kode pesan tersebut jika dikodekan dengan kode Huffman (termasuk spasi)!

Jawaban:

Hitung tiap frekuensi huruf unik di dalam “MATEMATIKA DISKRIT”

M	: 2	K	: 2
A	: 3	D	: 1
T	: 3	S	: 1
E	: 1	R	: 1
I	: 3	_	: 1

Pohon Huffman dan kode Huffman yang terbentuk adalah seperti berikut



Simbol	Frekuensi	Kode
A	3	11
T	3	000
I	3	001
M	2	011
K	2	0100
E	1	0101
D	1	1000
S	1	1001
R	1	1010
_	1	1011

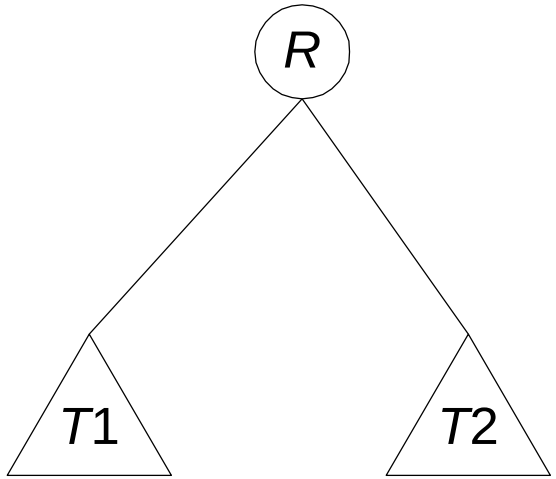
Jadi panjang kode Huffman dari "MATEMATIKA DISKRIT" adalah,
3*2 + 3*3 + 3*3 + 2*3 + 2*4 + 1*4 + 1*4 + 1*4 + 1*4 + 1*4
= 6 + 9 + 9 + 6 + 8 + 4 + 4 + 4 + 4 + 4 = 58 bit

Latihan

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

rasio = (jumlah bit setelah kompresi/jumlah bit sebelum kompresi) x 100%

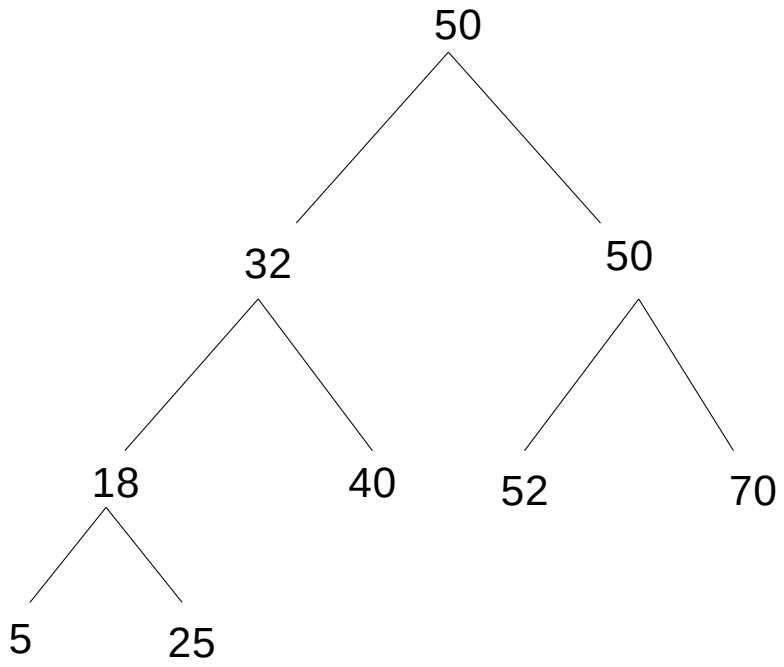
5. Pohon Pencarian Biner



Kunci($T1$) < Kunci(R)

Kunci($T2$) > Kunci(R)

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



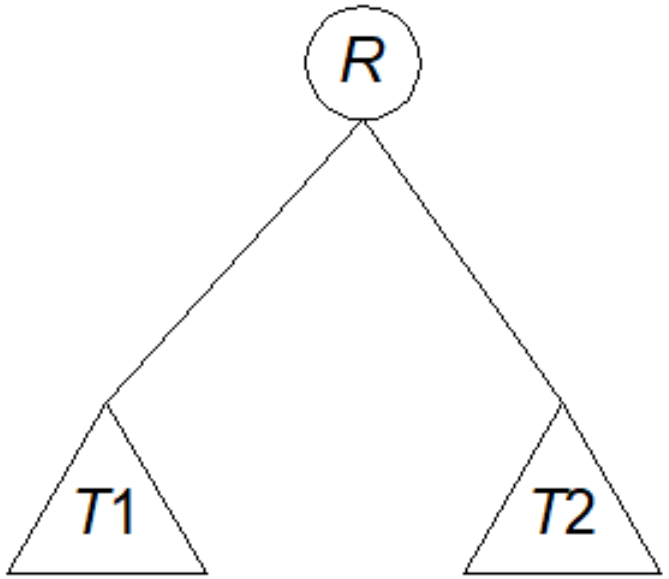
Berapa banyak perbandingan yang dilakukan untuk menemukan 25?
Berapa banyak perbandingan yang dilakukan untuk menemukan 100?

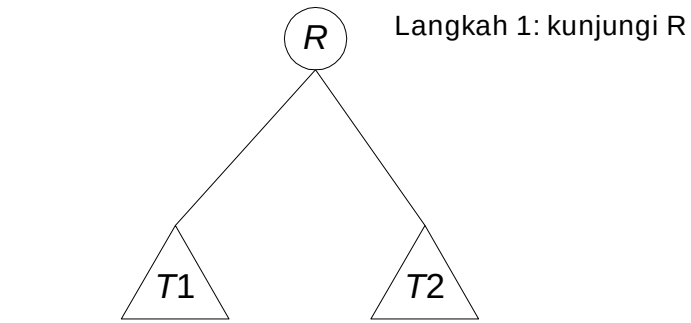
Penelusuran (traversal) Pohon Biner

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



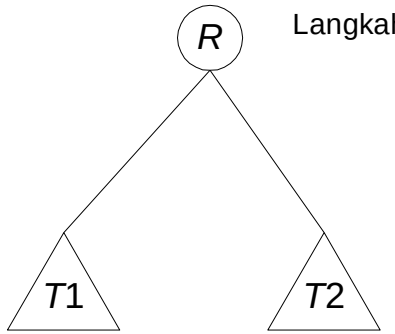


Langkah 1: kunjungi R

Langkah 2: kunjungi *T*1 secara *preorder*

Langkah 3: kunjungi *T*2 secara *preorder*

(a) *preorder*

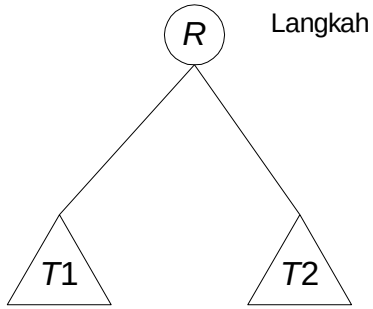


Langkah 2: kunjungi R

Langkah 1: kunjungi *T*1 secara *inorder*

Langkah 3: kunjungi *T*2 secara *inorder*

(b) *inorder*



Langkah 3: kunjungi R

Langkah 1: kunjungi *T*1 secara *postorder*

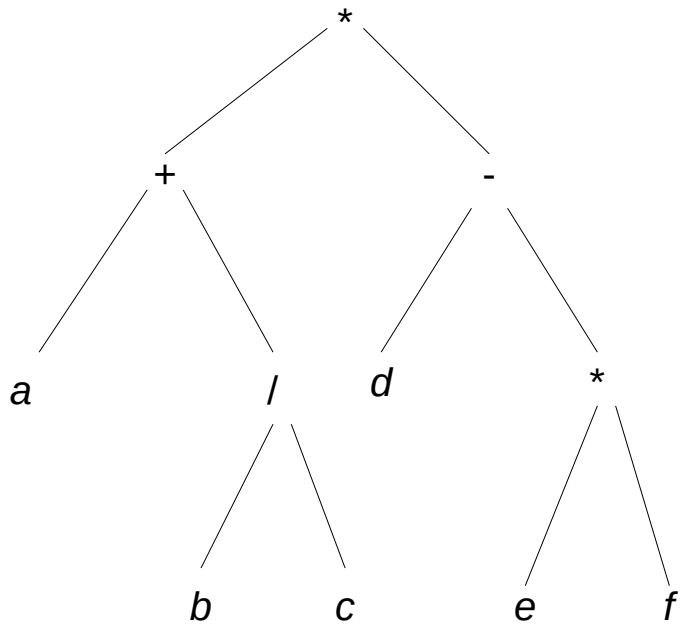
Langkah 2: kunjungi *T*2 secara *postorder*

(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

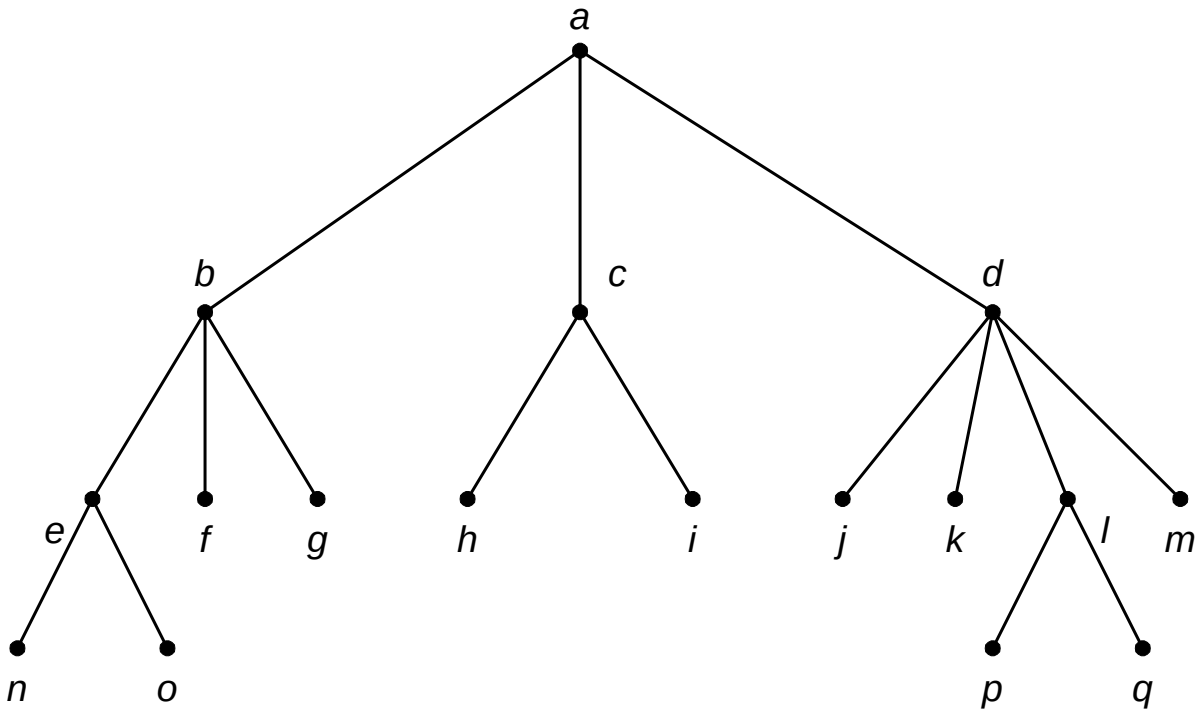
Diberikan masukan berupa rangkaian karakter dengan urutan sebagai berikut:

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:



TAMAT