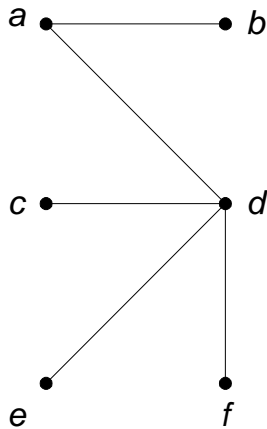
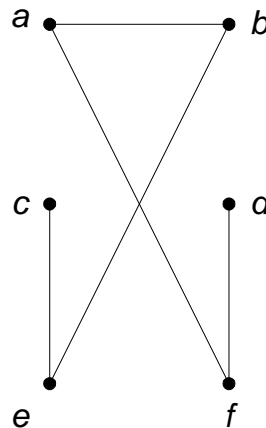


Definisi

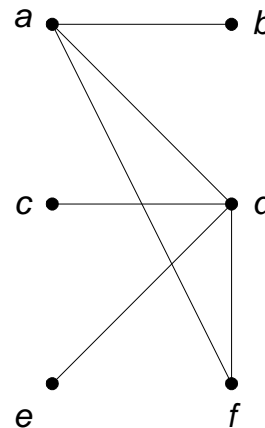
- **Pohon** adalah graf tak-berarah terhubung yang tidak mengandung sirkuit



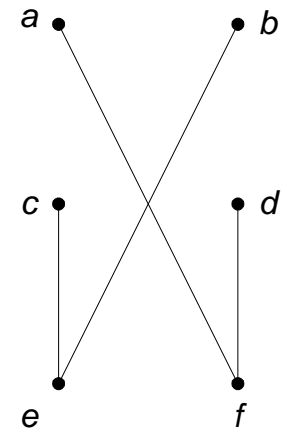
pohon



pohon



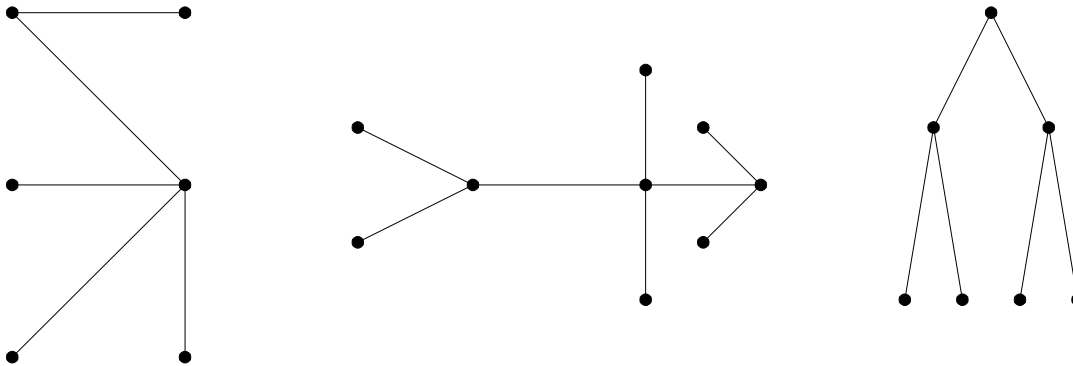
bukan pohon



bukan pohon

Hutan (*forest*) adalah

- kumpulan pohon yang saling lepas, atau
- graf tidak terhubung yang tidak mengandung sirkuit. Setiap komponen di dalam graf terhubung tersebut adalah pohon.



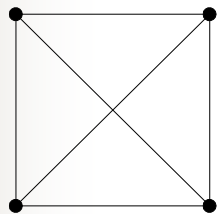
Hutan yang terdiri dari tiga buah pohon

Sifat-sifat (properti) pohon

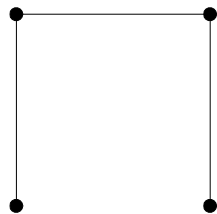
- **Teorema.** Misalkan $G = (V, E)$ adalah graf tak-berarah sederhana dan jumlah simpulnya n . Maka, semua pernyataan di bawah ini adalah ekuivalen:
 1. G adalah pohon.
 2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
 3. G terhubung dan memiliki $m = n - 1$ buah sisi.
 4. G tidak mengandung sirkuit dan memiliki $m = n - 1$ buah sisi.
 5. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
 6. G terhubung dan semua sisinya adalah jembatan.
- Teorema di atas dapat dikatakan sebagai definisi lain dari pohon.

Pohon Merentang (*spanning tree*)

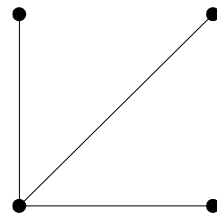
- Pohon merentang dari graf terhubung adalah upagraf merentang yang berupa pohon.
- Pohon merentang diperoleh dengan memutus sirkuit di dalam graf.



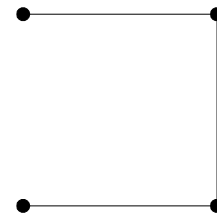
G



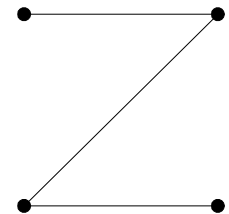
T_1




T_2



T_3

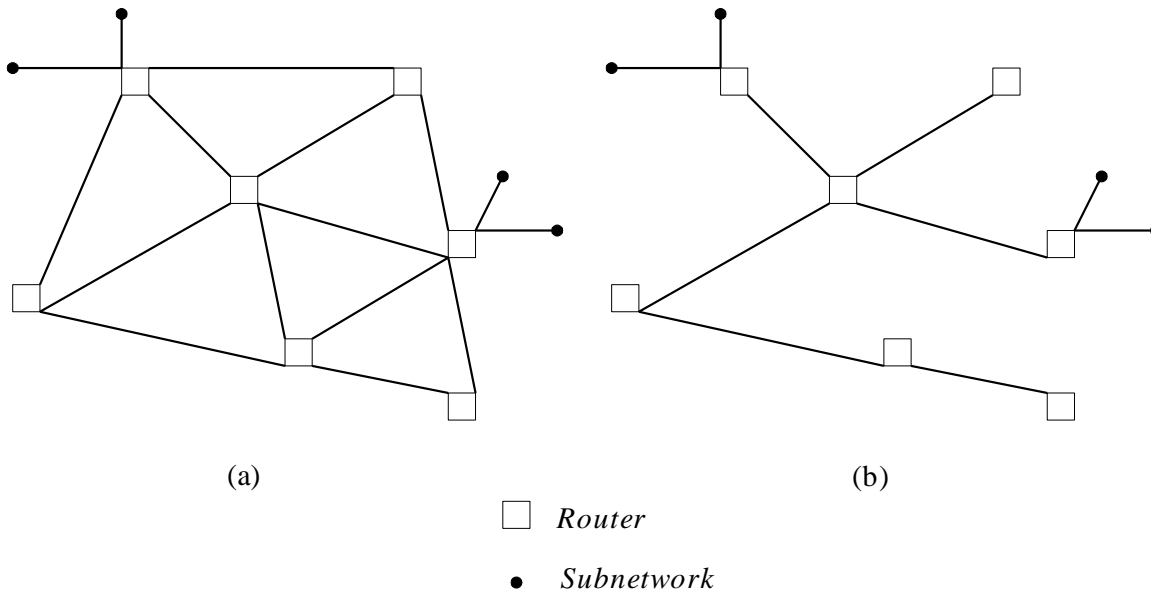


T_4

- 
- Setiap graf terhubung mempunyai paling sedikit satu buah pohon merentang.
 - Graf tak-terhubung dengan k komponen mempunyai k buah hutan merentang yang disebut hutan merentang (*spanning forest*).

Aplikasi Pohon Merentang

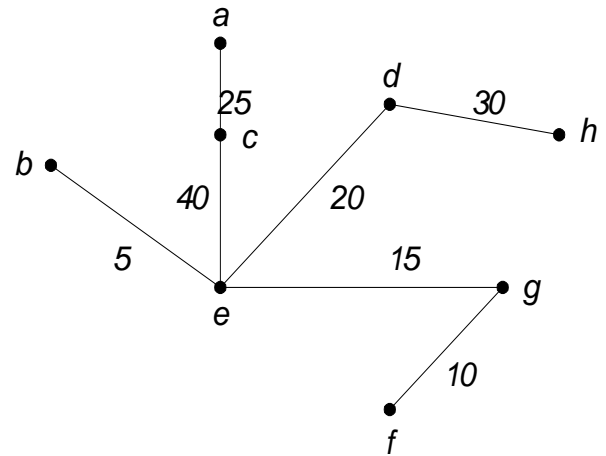
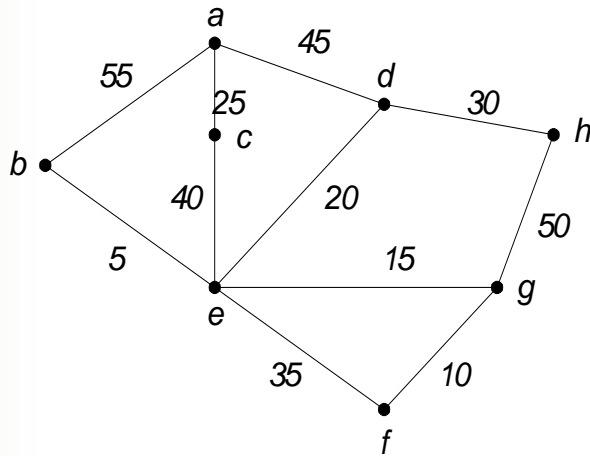
1. Jumlah ruas jalan semimumimum mungkin yang menghubungkan semua kota sehingga setiap kota tetap terhubung satu sama lain.
2. Perutean (*routing*) pesan pada jaringan komputer.



(a) Jaringan komputer, (b) Pohon merentang *multicast*

Pohon Merentang Minimum

- Graf terhubung-berbobot mungkin mempunyai lebih dari 1 pohon merentang.
- Pohon merentang yang berbobot minimum –dinamakan **pohon merentang minimum** (*minimum spanning tree*).





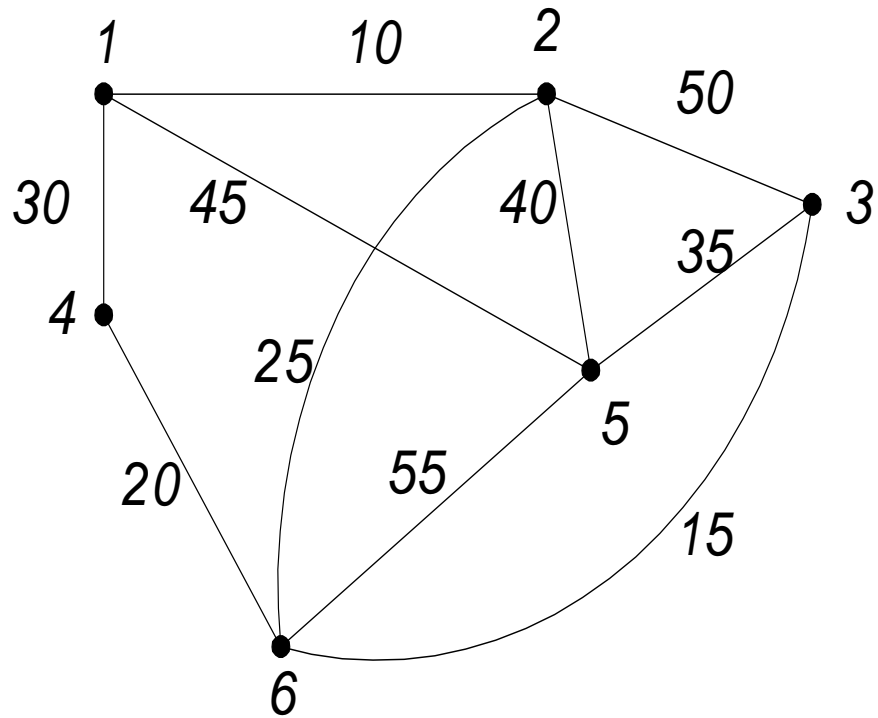
Algoritma Prim

Langkah 1: ambil sisi dari graf G yang berbobot minimum, masukkan ke dalam T .


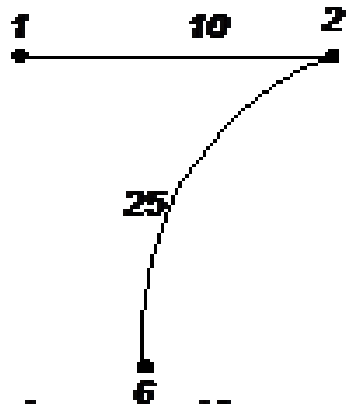
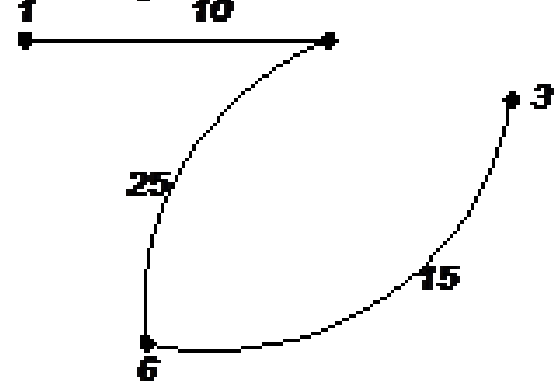
Langkah 2: pilih sisi (u, v) yang mempunyai bobot minimum dan bersisian dengan simpul di T , tetapi (u, v) tidak membentuk sirkuit di T . Masukkan (u, v) ke dalam T .

Langkah 3: ulangi langkah 2 sebanyak $n - 2$ kali.

Contoh:



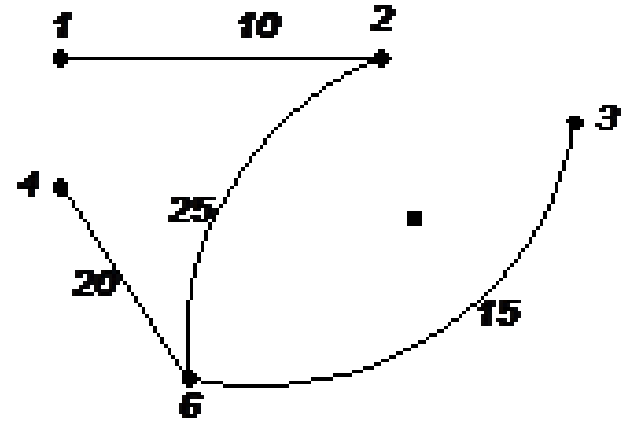


Langkah	Sisi	Bobot	Pohon rentang
1	(1, 2)	10	
2	(2, 6)	25	
3	(3, 6)	15	

4

(4, 6)

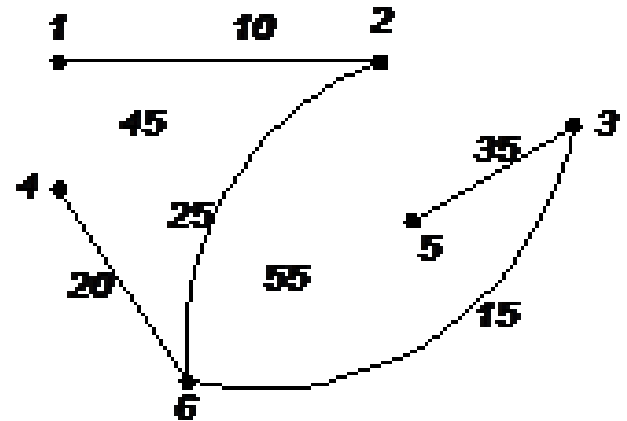
20



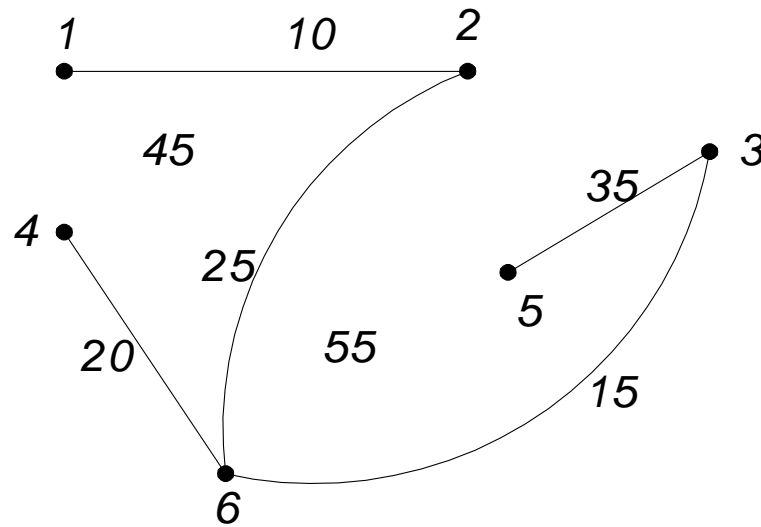
5

(3, 5)

35



Pohon merentang minimum yang dihasilkan:

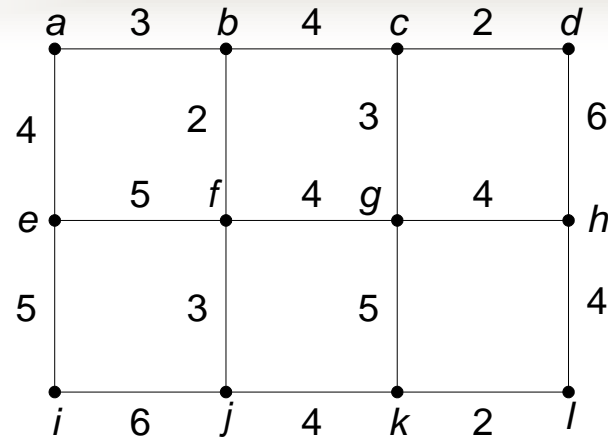


$$\text{Bobot} = 10 + 25 + 15 + 20 + 35 = 105$$

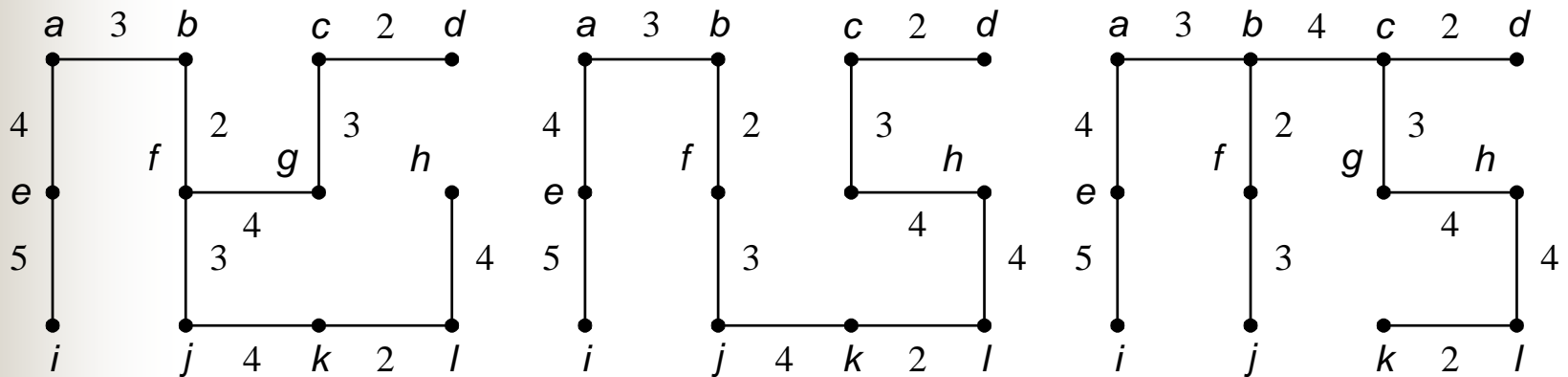
Pohon merentang yang dihasilkan tidak selalu unik meskipun bobotnya tetap sama.

Hal ini terjadi jika ada beberapa sisi yang akan dipilih berbobot sama.

Contoh:



Tiga buah pohon merentang minimumnya:



Bobotnya sama yaitu = 36



Algoritma Kruskal

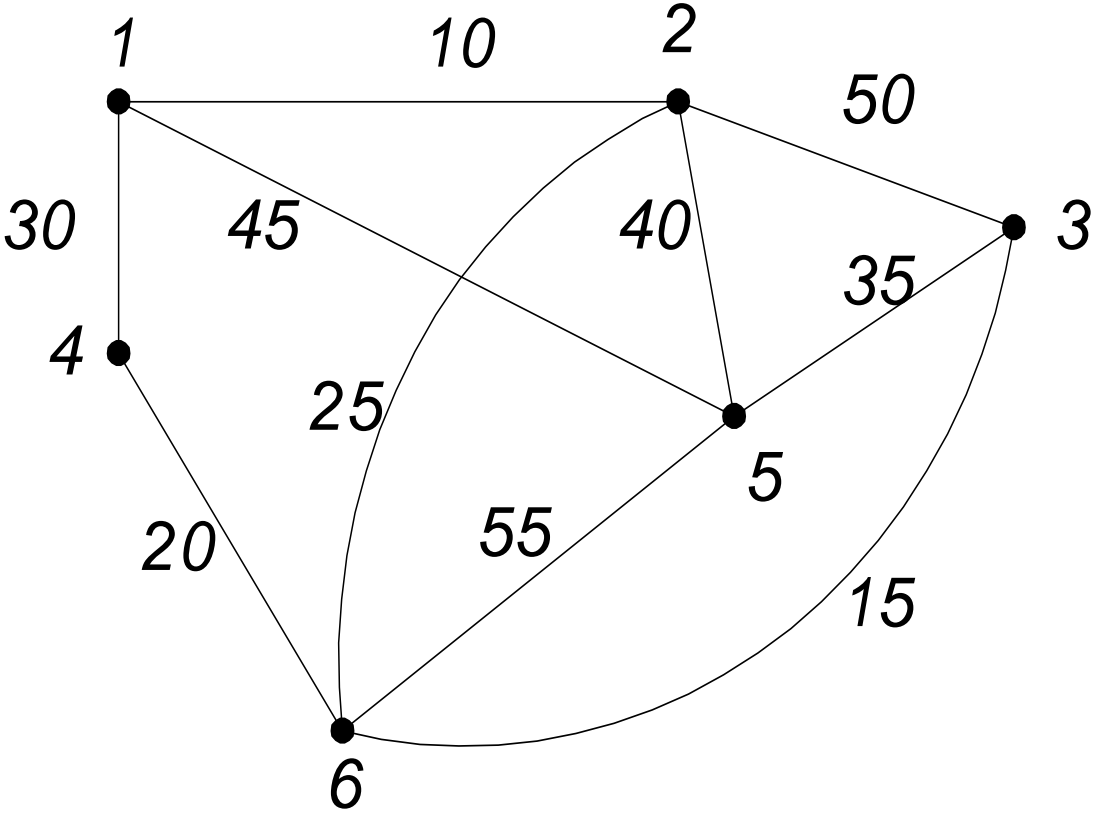
(Langkah 0: sisi-sisi dari graf sudah diurut menaik berdasarkan bobotnya – dari bobot kecil ke bobot besar)

Langkah 1: T masih kosong

Langkah 2: pilih sisi (u, v) dengan bobot minimum yang tidak membentuk sirkuit di T . Tambahkan (u, v) ke dalam T .


Langkah 3: ulangi langkah 2 sebanyak $n - 1$ kali.

Contoh:



Sisi-sisi diurut menaik

Sisi	(1,2)	(3,6)	(4,6)	(2,6)	(1,4)	(3,5)	(2,5)	(1,5)	(2,3)	(5,6)
Bobot	10	15	20	25	30	35	40	45	50	55

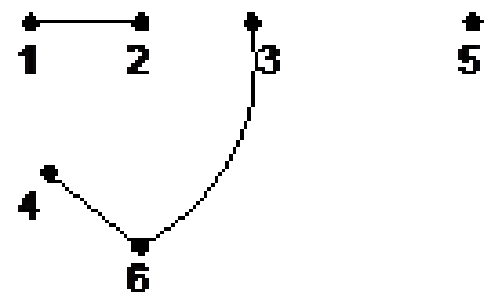
Langkah	Sisi	Bobot	Hutan merentang
0			• 1 • 2 • 3 • 4 • 5 • 6
1	(1, 2)	10	• 1 —• 2
2	(3, 6)	15	• 1 —• 2 • 3 • 4 • 5 <div style="margin-left: 150px;">• 6</div> 



3

(4, 6)

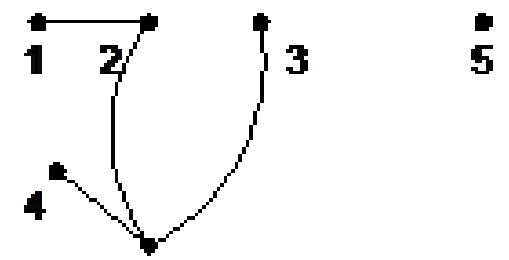
20



4

(2, 6)

25



5

(1, 4)

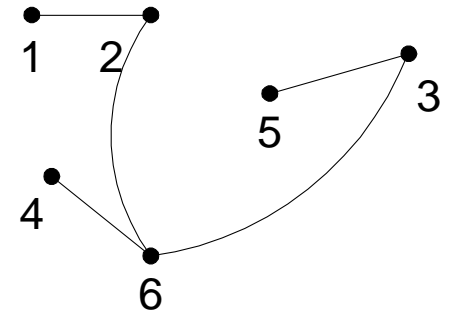
30

ditolak

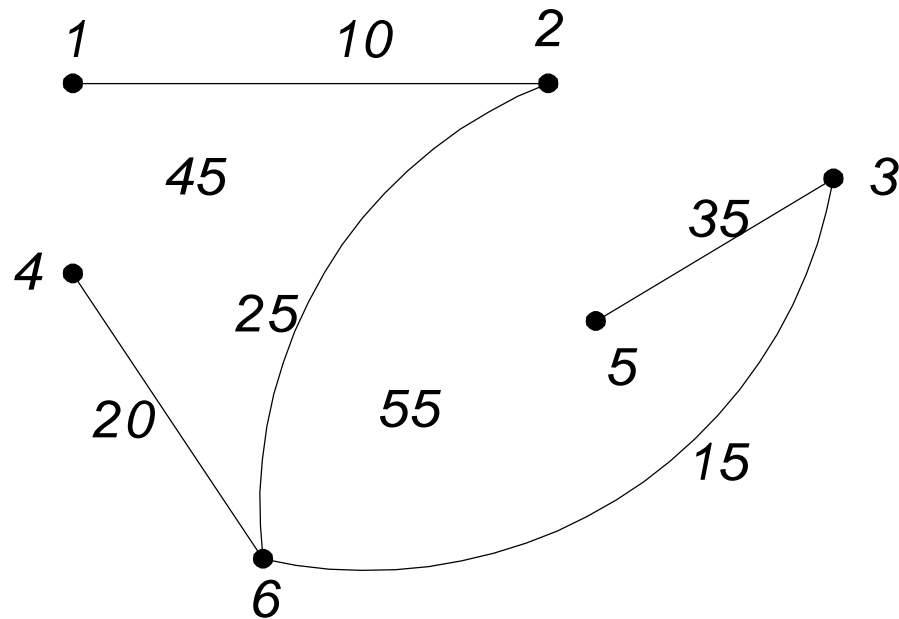
6

(3, 5)

35



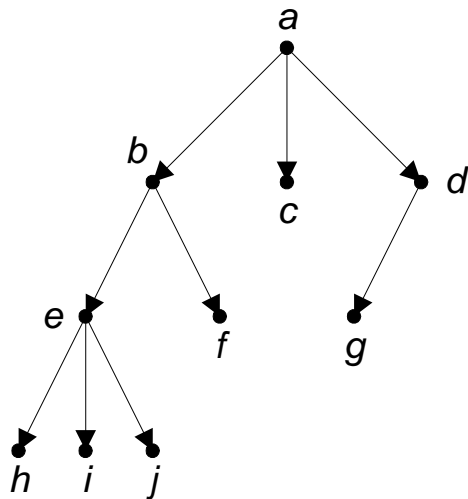
Pohon merentang minimum yang dihasilkan:



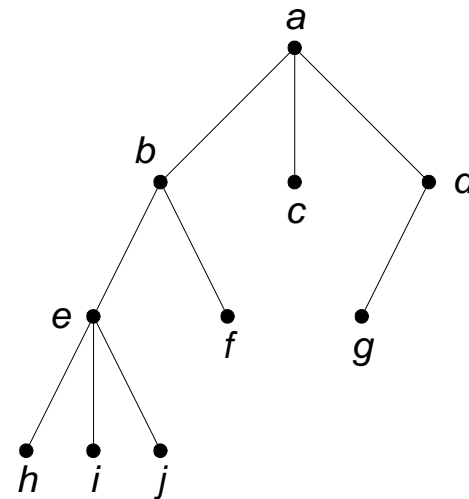
$$\text{Bobot} = 10 + 25 + 15 + 20 + 35 = 105$$

Pohon berakar (*rooted tree*)

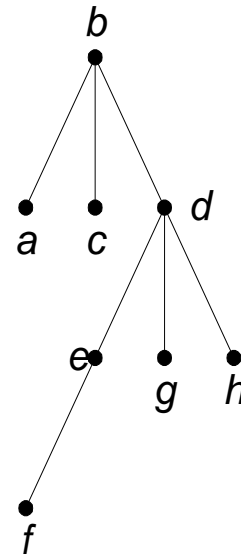
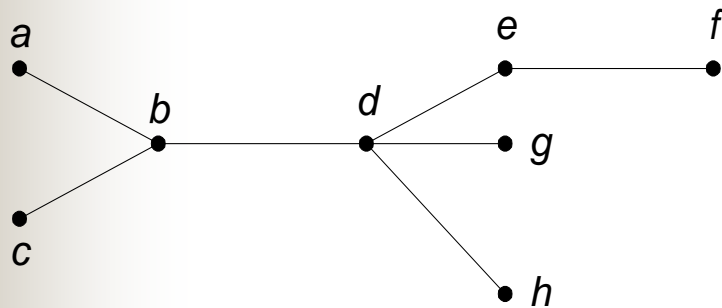
- Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan **pohon berakar** (*rooted tree*).



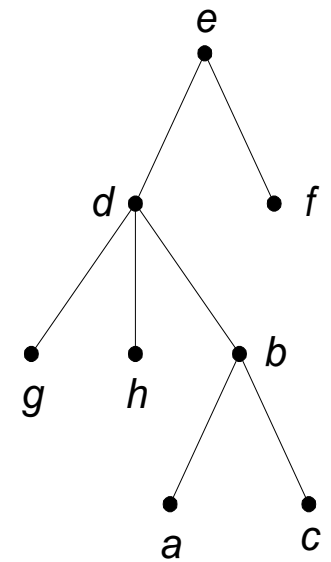
(a) Pohon berakar



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



b sebagai akar



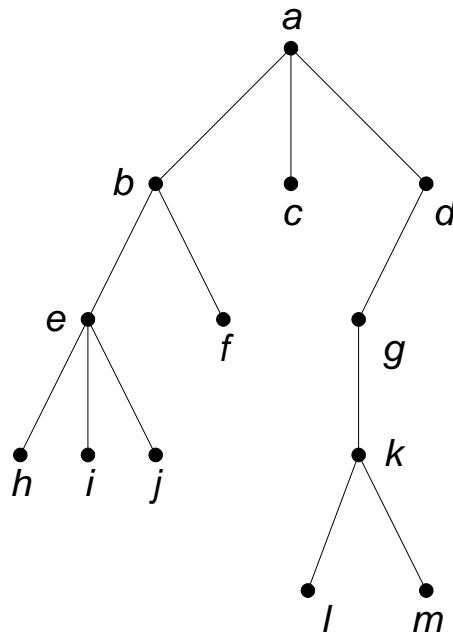
e sebagai akar

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



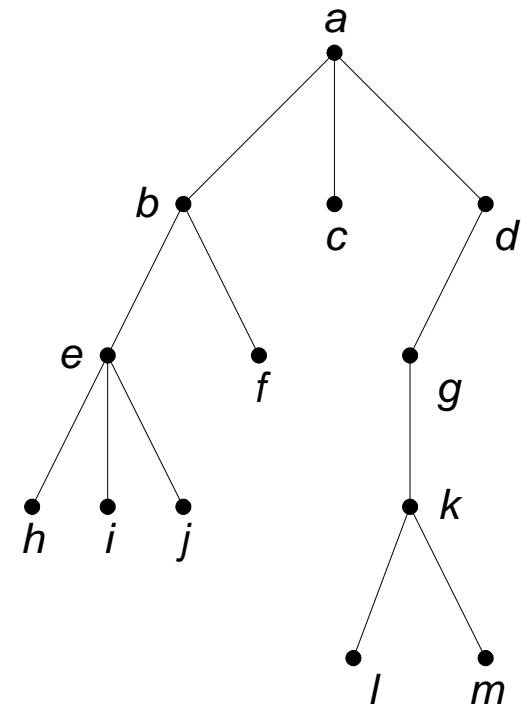
2. Lintasan (*path*)

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

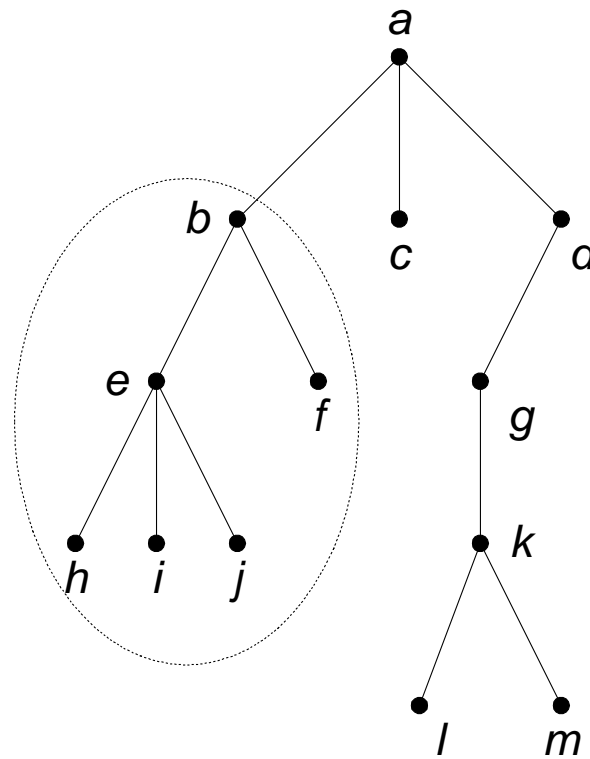
Panjang lintasan dari a ke j adalah 3.

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*)

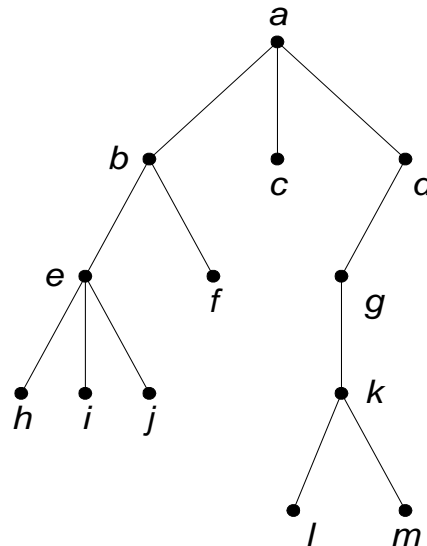
Derajat sebuah simpul adalah jumlah upapohon (atau jumlah anak) pada simpul tersebut.

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 atas berderajat 3

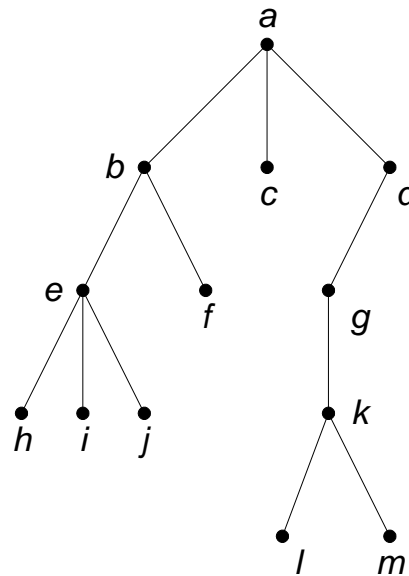


6. Daun (*leaf*)

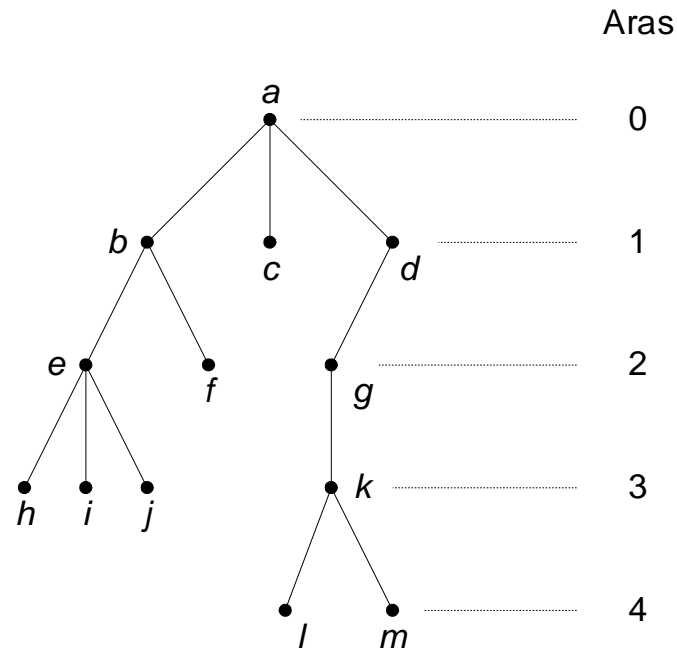
Simpul yang berderajat nol (atau tidak mempunyai anak) disebut **daun**. Simpul h, i, j, f, c, l , dan m adalah daun.

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

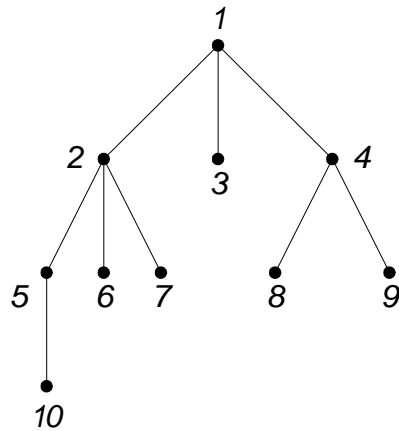


9. Tinggi (*height*) atau Kedalaman (*depth*)

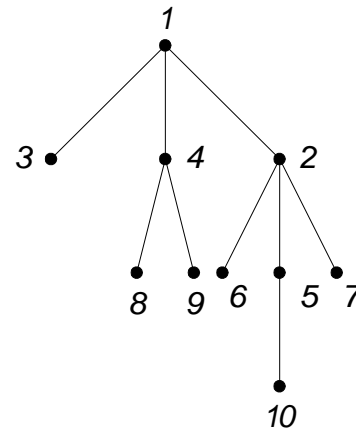
Aras maksimum dari suatu pohon disebut **tinggi** atau **kedalaman** pohon tersebut. Pohon di atas mempunyai tinggi 4.

Pohon Terurut (*ordered tree*)

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



(a)

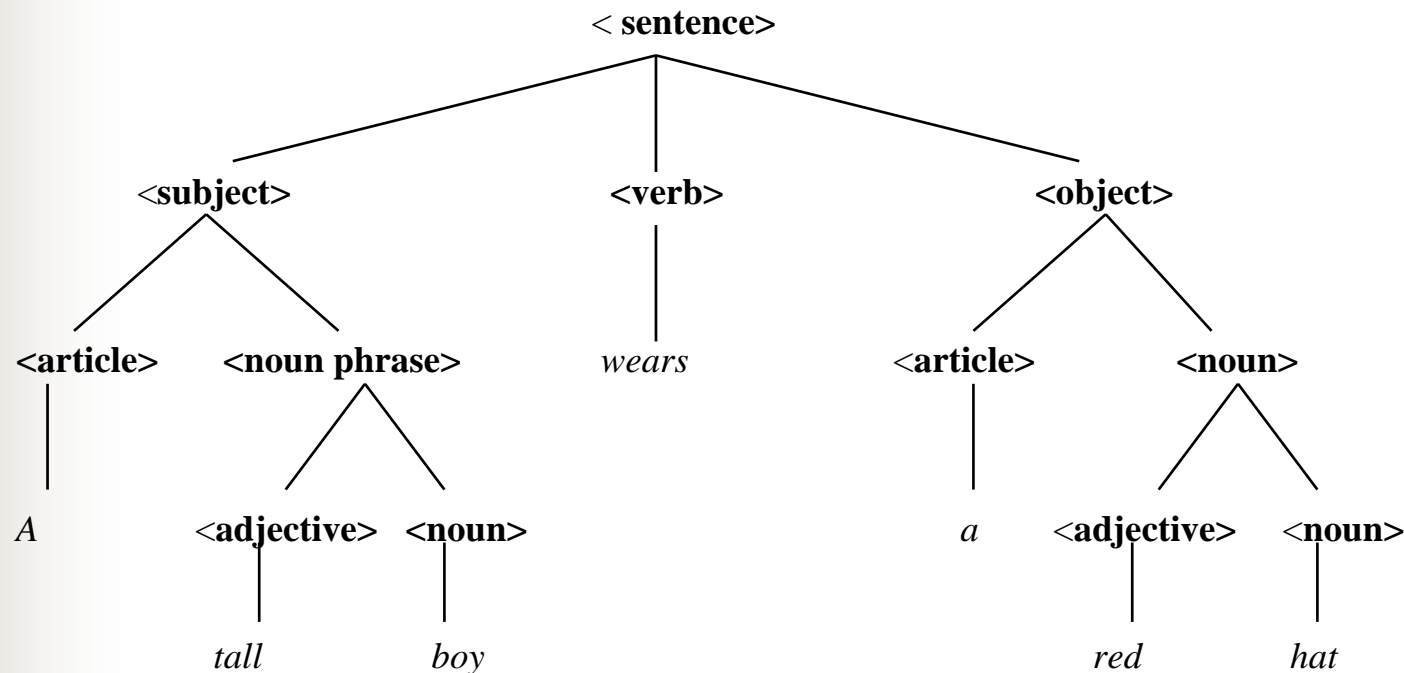


(b)

(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**.



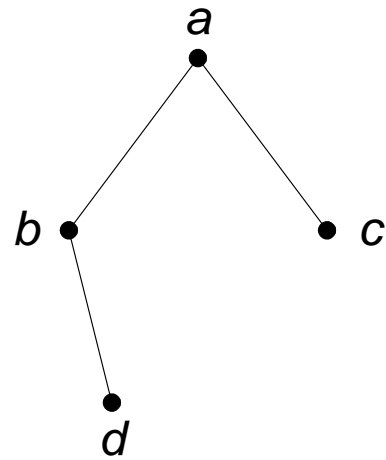
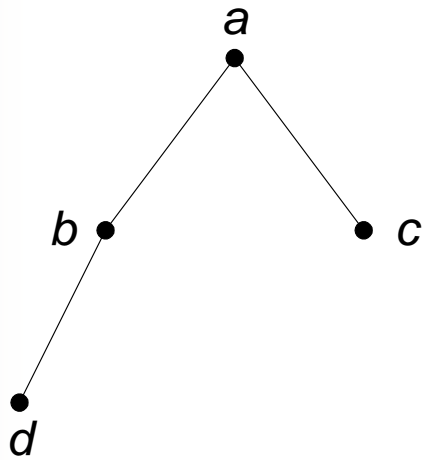
Gambar Pohon parsing dari kalimat *A tall boy wears a red hat*

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

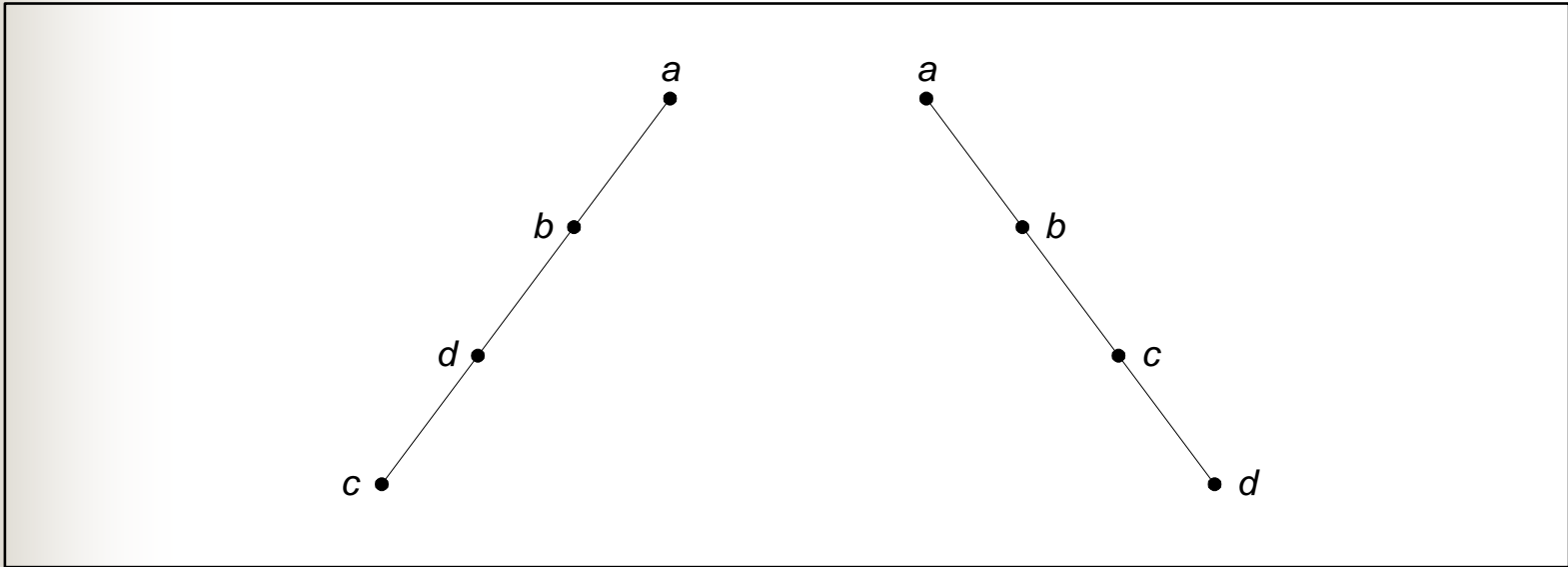


Pohon Biner (*binary tree*)

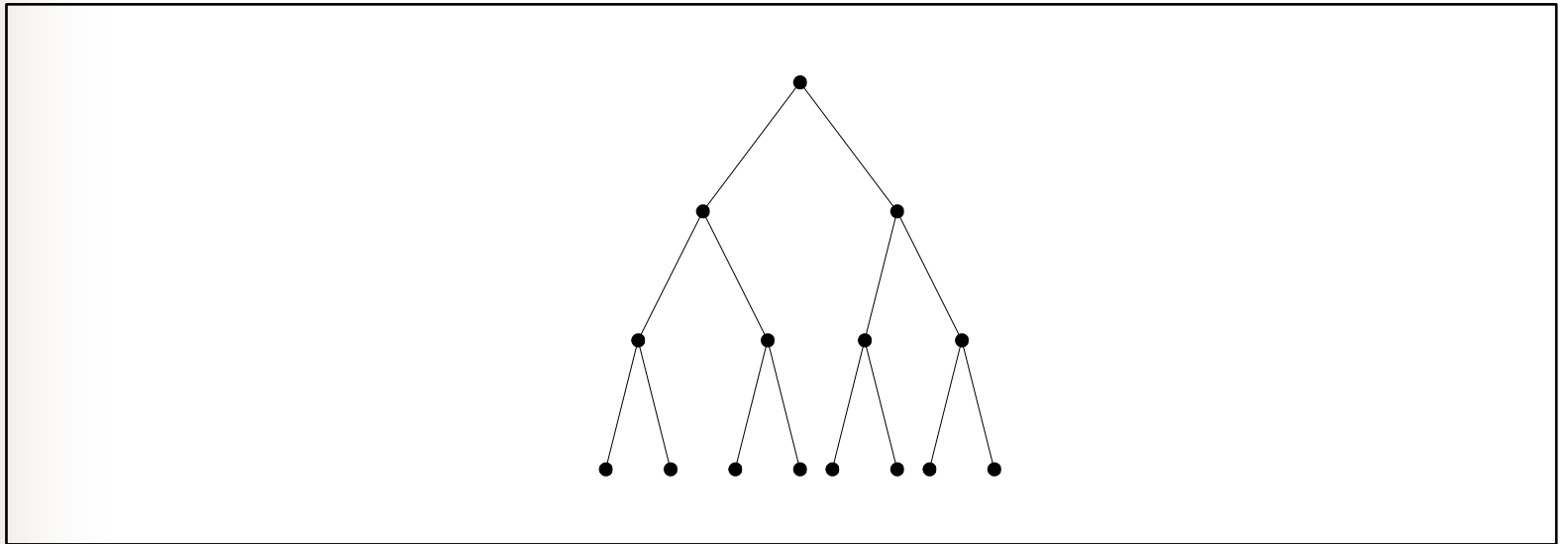
- Adalah pohon *n-ary* dengan $n = 2$.
- Pohon yang paling penting karena banyak aplikasinya.
- Setiap simpul di adlam pohon biner mempunyai paling banyak 2 buah anak.
- Dibedakan antara anak kiri (*left child*) dan anak kanan (*right child*)
- Karena ada perbedaan urutan anak, maka pohon biner adalah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



Gambar Dua buah pohon biner yang berbeda



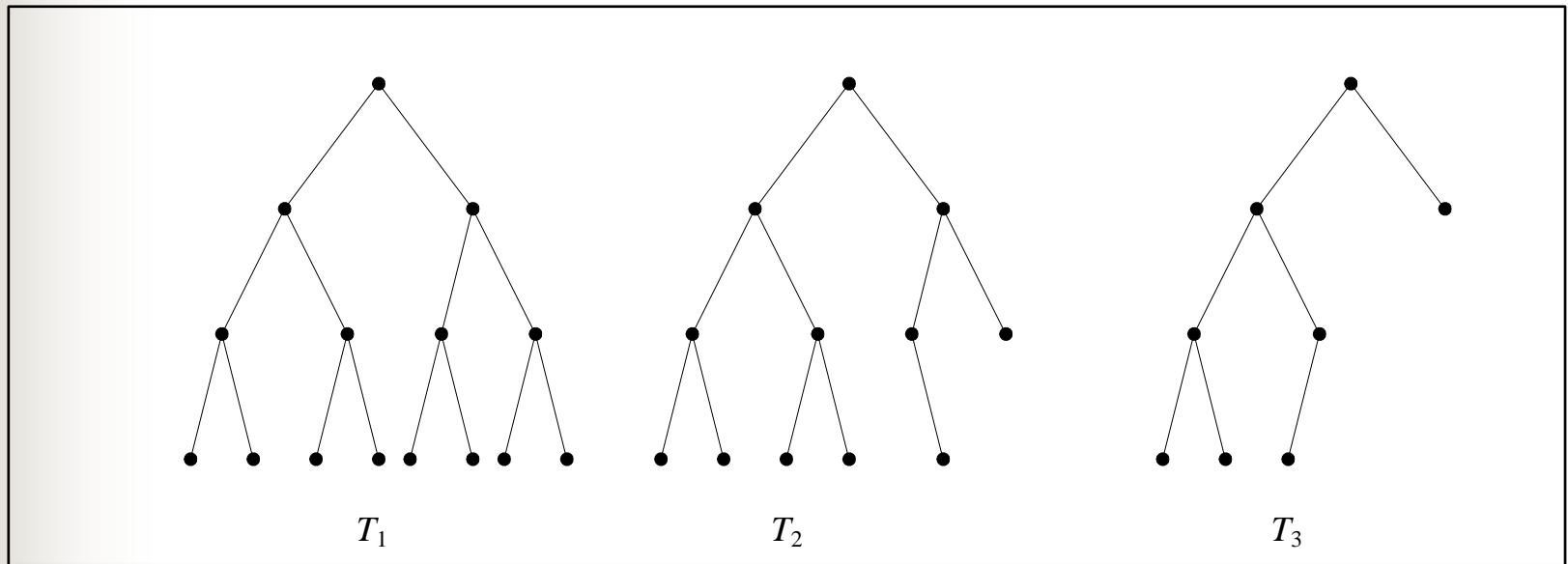
Gambar (a) Pohon condong-kiri, dan (b) pohon condong kanan



Gambar Pohon biner penuh

Pohon Biner Seimbang

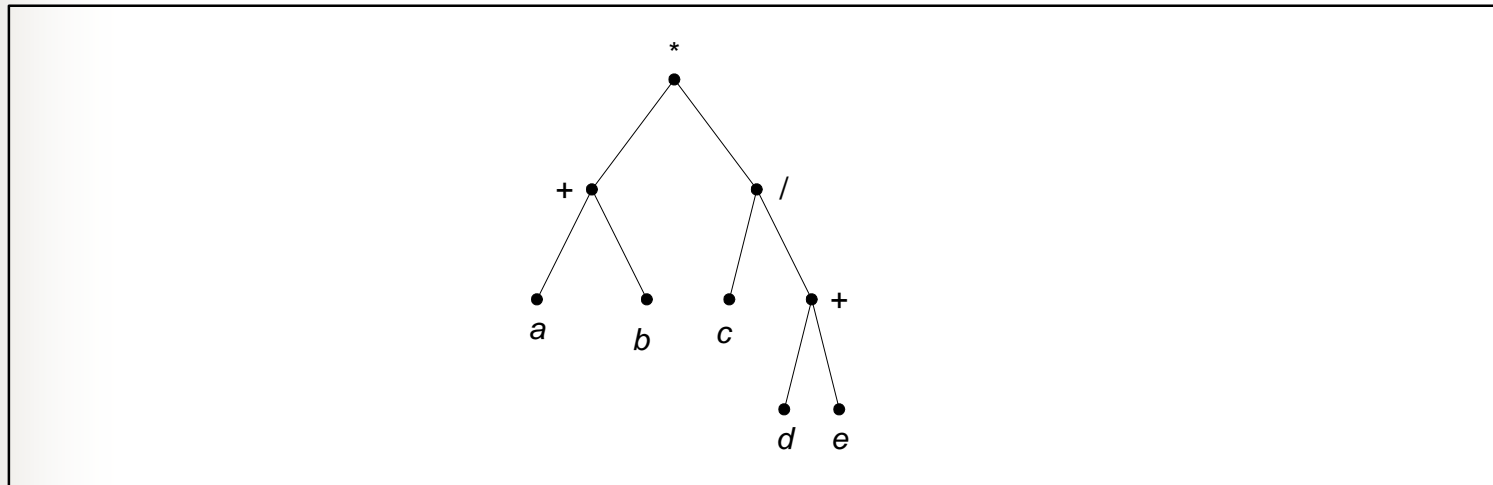
Pada beberapa aplikasi, diinginkan tinggi upapohon kiri dan tinggi upapohon kanan yang seimbang, yaitu berbeda maksimal 1.



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

Terapan Pohon Biner

1. Pohon Ekspresi

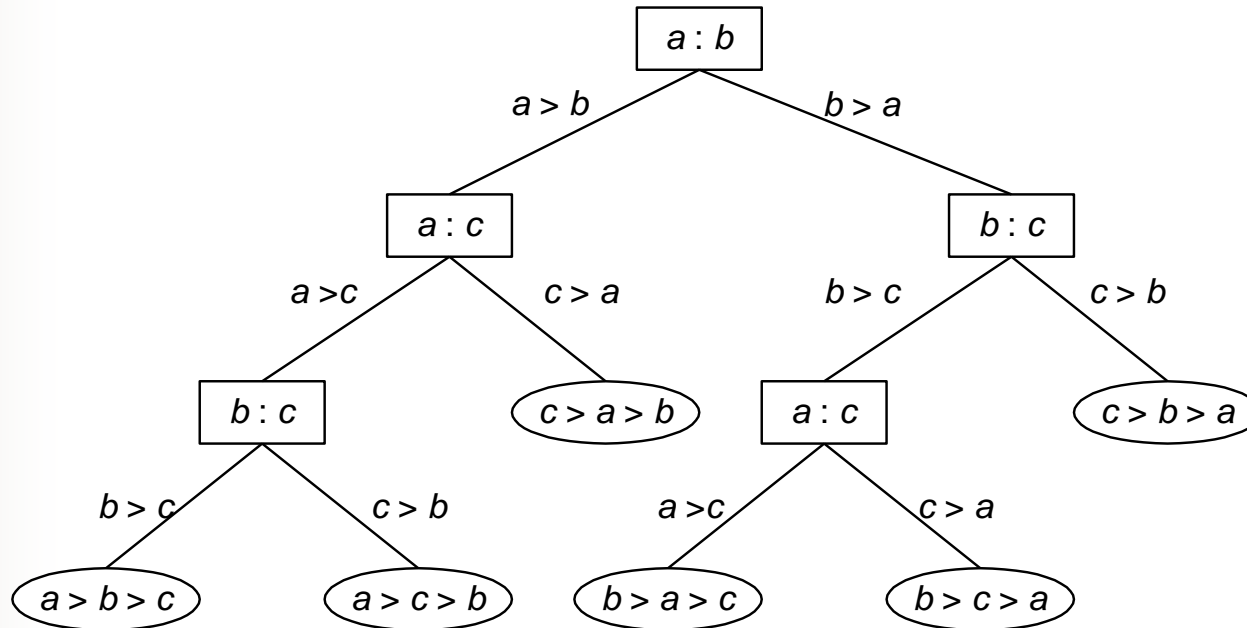


Gambar Pohon ekspresi dari $(a + b) * (c / (d + e))$

daun \rightarrow *operand*

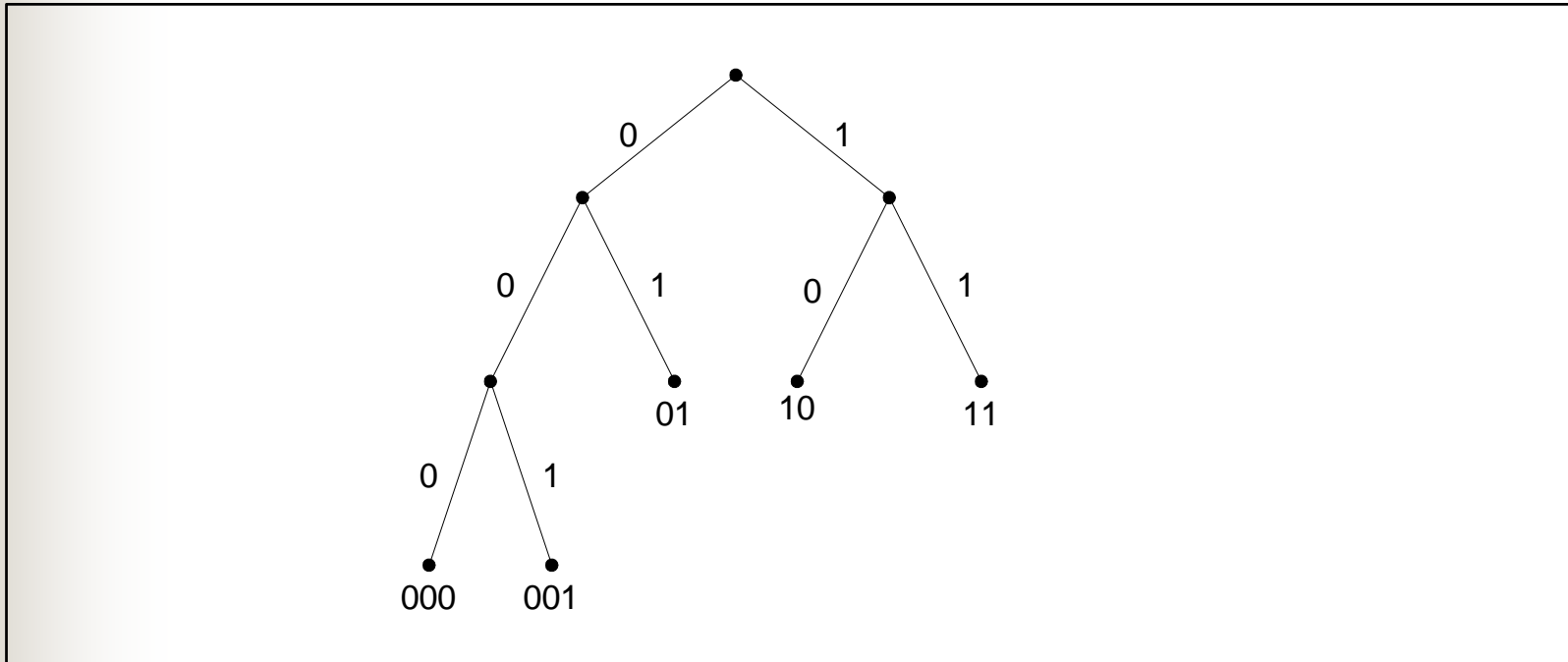
simpul dalam \rightarrow operator

2. Pohon Keputusan



Gambar Pohon keputusan untuk mengurutkan 3 buah elemen

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 '*ABACCCA*':

01000001010000010010000010100000110100000110100010001000001

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


Tabel Tabel kekerapan (frekuensi) dan kode Huffman
untuk *string* *ABACCD*A

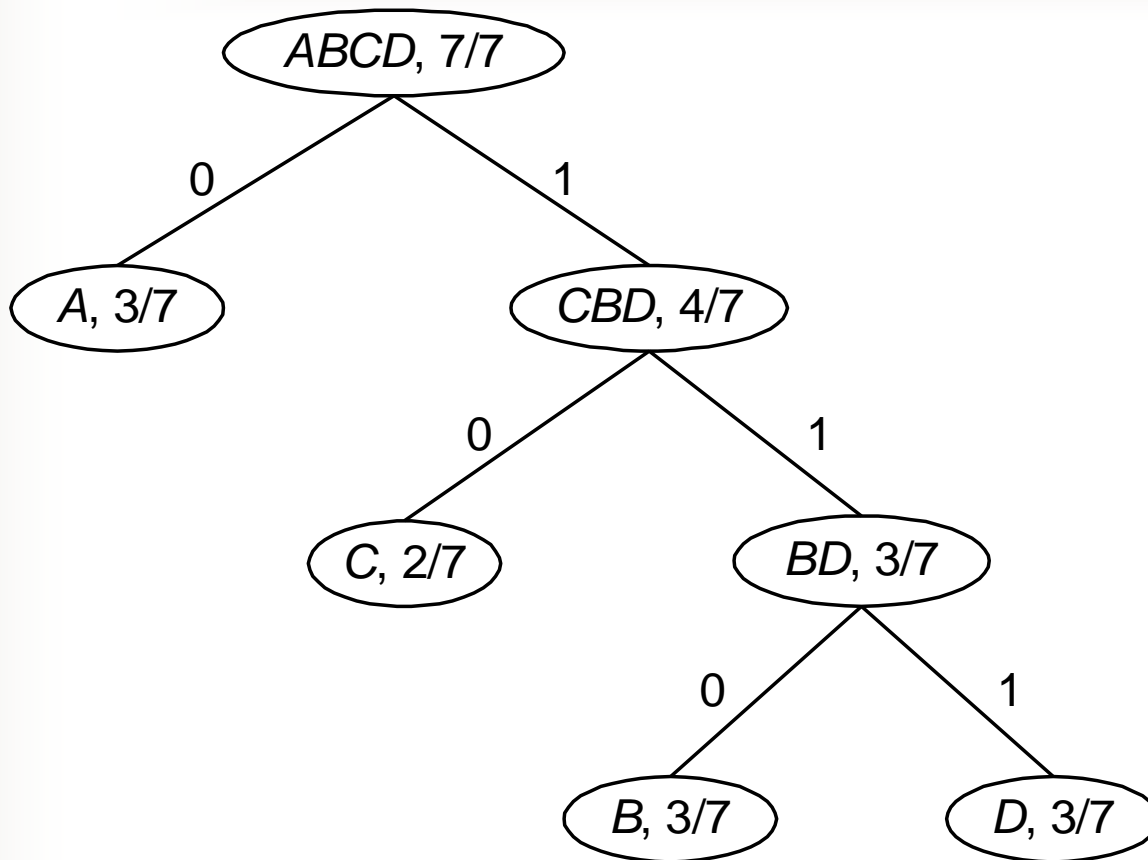
Simbol	Kekerapan	Peluang	Kode Huffman
<i>A</i>	3	$3/7$	0
<i>B</i>	1	$1/7$	110
<i>C</i>	2	$2/7$	10
<i>D</i>	1	$1/7$	111

Dengan kode Huffman, rangkaian bit untuk '*ABACCD*A':

0110010101110

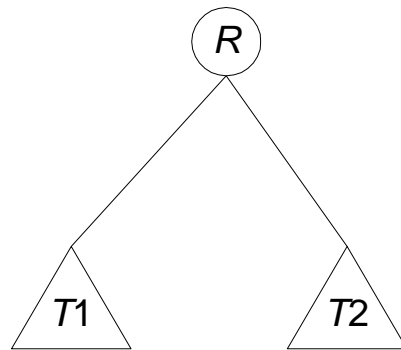
hanya 13 bit!

- 
- Algoritma pembentukan pohon 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.



- $A = 0, C = 10, B = 110, D = 111$

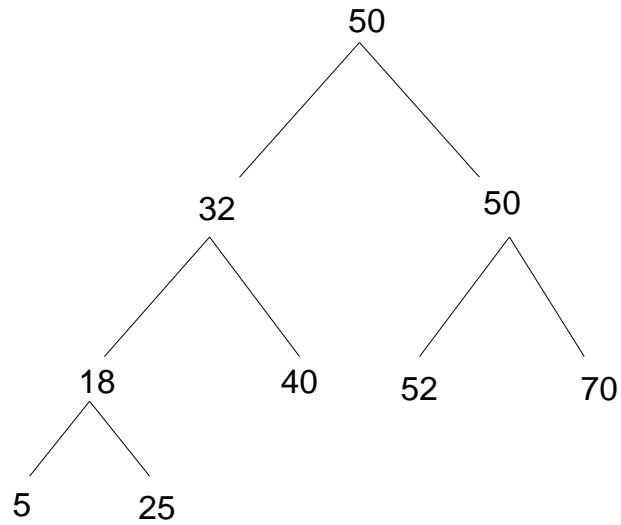
5. Pohon Pencarian Biner



Kunci($T1$) < Kunci(R)

Kunci($T2$) > Kunci(R)

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





Penelusuran (traversal) Pohon Biner

1. *Preorder* : R, T_1, T_2

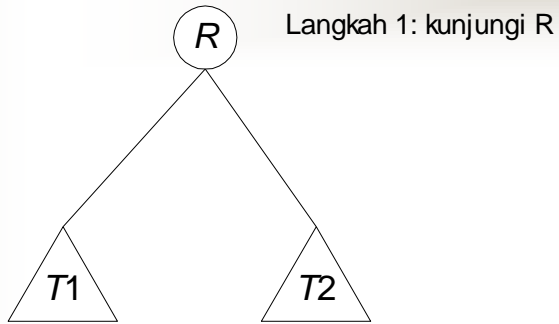
- kunjungi R
- kunjungi T_1 secara *preorder*
- kunjungi T_2 secara *preorder*

2. *Inorder* : T_1, R, T_2

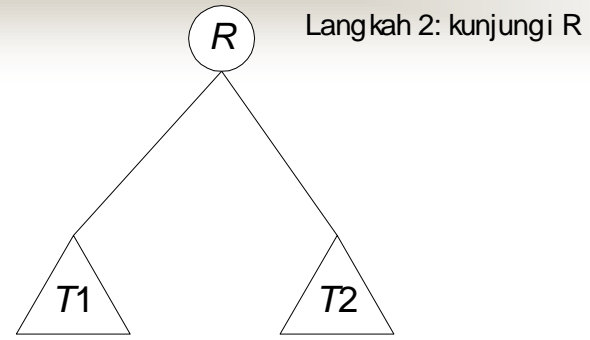
- kunjungi T_1 secara *inorder*
- kunjungi R
- kunjungi T_2 secara *inorder*

3. *Postorder* : T_1, T_2, R

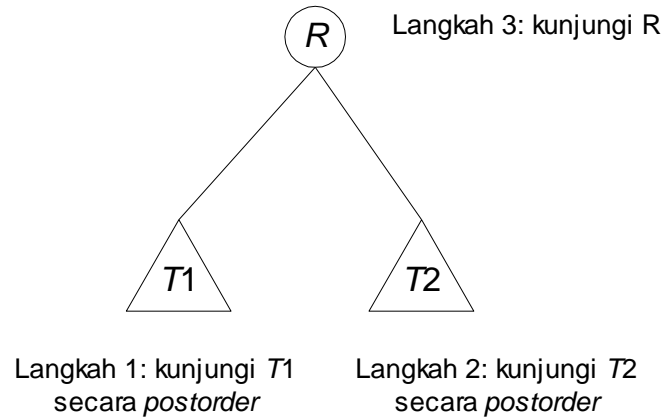
- kunjungi T_1 secara *postorder*
- kunjungi T_2 secara *postorder*
- kunjungi R



(a) preorder



(b) inorder



(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)

