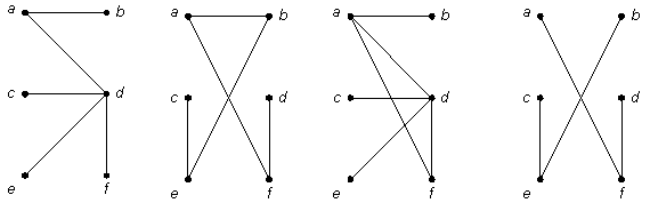
Pohon (*Tree*)

Sesi 12-13

1

Definisi

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

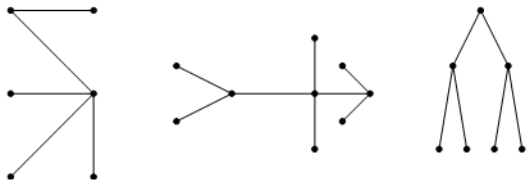


pohon pohon bukan pohon bukan pohon

2

Definisi

- **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

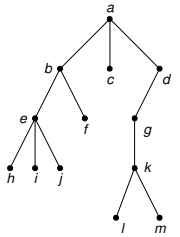
3

Sifat-sifat Pohon

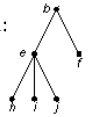
- **Teorema.** Misalkan $G = (V, E)$ adalah graf tak-berarah sederhana dan jumlah simpulnya n. Maka, semua pernyataan di bawah ini adalah ekivalen:
  - G adalah pohon.
  - Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
  - G terhubung dan memiliki $m = n - 1$ buah sisi.
  - G tidak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

4

Terminologi (1)

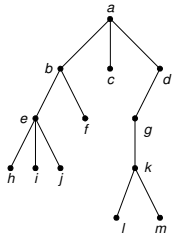


- **Anak(*child* atau *children*) & Orangtua(*parent*)**
  - *b*, *c*, dan *d* adalah anak-anak simpul *a*,
  - *a* adalah orangtua dari anak-anak itu
- **Lintasan (*path*)**
  - Lintasan dari *a* ke *j* adalah *a*, *b*, *e*, *j*.
  - Panjang lintasan dari *a* ke *j* adalah 3.
- **Saudara kandung (*sibling*)**
  - *f* adalah saudara kandung *e*,
  - *g* bukan saudara kandung *e*, karena orangtua mereka berbeda.
- Subgrapgh:



5

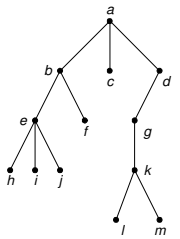
Terminologi (2)



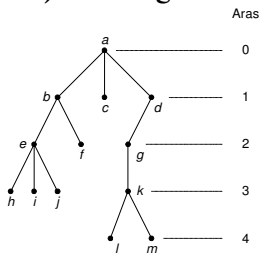
- **Derajat (*degree*)**
  - **Derajat** sebuah simpul adalah jumlah subgraph (atau jumlah anak) pada simpul tersebut.
  - Deg(*a*)= 3, Deg(*b*)= 2, Deg(*d*)= 1 dan Deg(*c*)= 0.
  - Jadi, derajat yang dimaksudkan di sini adalah derajat-keluar.
  - Derajat maksimum dari semua simpul merupakan derajat pohon itu sendiri. Pohon di samping berderajat 3
- **Daun (*leaf*)**
  - Simpul yang berderajat nol (atau tidak mempunyai anak) disebut **daun**. Simpul *h*, *i*, *j*, *f*, *c*, *l*, dan *m* adalah daun.
- **Simpul Dalam (*internal nodes*)**
  - Simpul yang mempunyai anak disebut **simpul dalam**. Simpul *b*, *d*, *e*, *g*, dan *k* adalah simpul dalam.

6

Terminologi (3)



- **Aras (*level*) atau Tingkat**

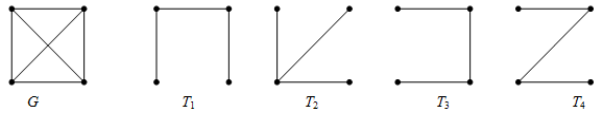


- **Tinggi (*height*) atau Kedalaman (*depth*)**
  - Aras maksimum dari suatu pohon disebut **tinggi** atau **kedalaman** pohon tersebut. Pohon di atas mempunyai tinggi 4

7

Pohon Merentang (*spanning tree*)

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

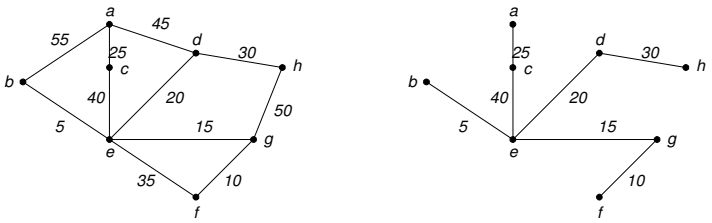


- Setiap graf terhubung mempunyai paling sedikit satu buah pohon merentang
- Graf tak-terhubung dengan *k* komponen mempunyai *k* buah hutan merentang yang disebut hutan merentang (*spanning forest*)

8

Pohon Rentang Minimum

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



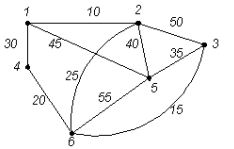
9

Algoritma Prim

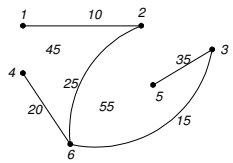
```
procedure Prim(input G : graf, output T : pohon)
{ Membentuk pohon merentang minimum T dari graf terhubung-
berbobot G.
Masukan: graf-berbobot terhubung G = (V, E), dengan |V|= n
Keluaran: pohon rentang minimum T = (V, E')
}
Deklarasi
  i, p, q, u, v : integer

Algoritma
  Cari sisi (p,q) dari E yang berbobot terkecil
  T ← {(p,q)}
  for i←1 to n-2 do
    Pilih sisi (u,v) dari E yang bobotnya terkecil namun
    bersisian dengan simpul di T
    T ← T ∪ {(u,v)}
  endfor
```

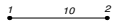
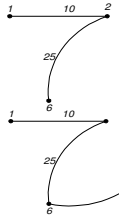
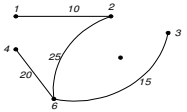
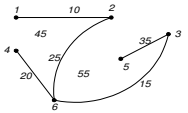
10

Contoh



Minimum Spanning tree yang dihasilkan adalah



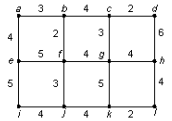
Langkah	Sisi	Bobot	Pohon rentang
1	(1, 2)	10	
2	(2, 6)	25	
3	(3, 6)	15	
4	(4, 6)	20	
5	(3, 5)	35	

Bobot
10 + 25 + 15 + 20 + 35 = 105

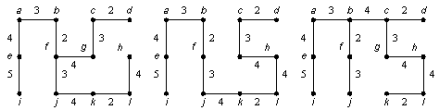
11

Contoh

- Pohon merentang yang dihasilkan tidak selalu unik meskipun bobotnya tetap sama. Ini terjadi jika ada beberapa sisi yang akan dipilih berbobot sama



Tiga buah pohon merentang minimumnya:



Bobotnya sama yaitu = 36

12

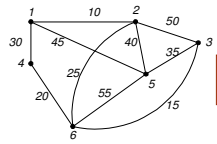
Algoritma Kruskal

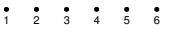
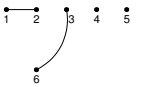
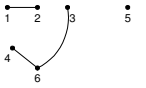
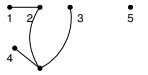
```
procedure Kruskal(input G : graf, output T : pohon)
{ Membentuk pohon merentang minimum T dari graf terhubung -
berbobot G.
Masukan: graf-berbobot terhubung G = (V, E), dengan |V|= n
Keluaran: pohon rentang minimum T = (V, E')
}
Deklarasi
  i, p, q, u, v : integer

Algoritma
  ( Asumsi: sisi-sisi dari graf sudah diurut menaik
    berdasarkan bobotnya - dari bobot kecil ke bobot
    besar)
  T ← {}
  while jumlah sisi T < n-1 do
    Pilih sisi (u,v) dari E yang bobotnya terkecil
    if (u,v) tidak membentuk siklus di T then
       T ← T ∪ {(u,v)}
    endif
  endfor
```


Contoh

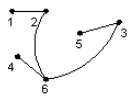
Sisi-sisi diurut menaik:

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

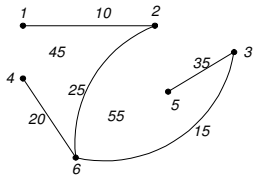


Langkah	Sisi	Bobot	Hutan merentang
0			
1	(1, 2)	10	
2	(3, 6)	15	
3	(4, 6)	20	
4	(2, 6)	25	

Contoh

Langkah	Sisi	Bobot	Hutan merentang
5	(1, 4)	30	ditolak
6	(3, 5)	35	

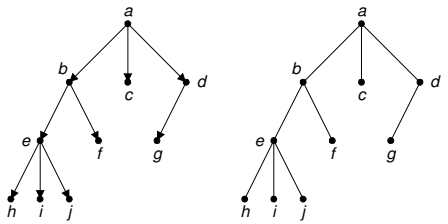
Minimum Spanning tree yang dihasilkan adalah



Bobot
10 + 25 + 15 + 20 + 35 = 105

Pohon berakar (Rooted Tree)

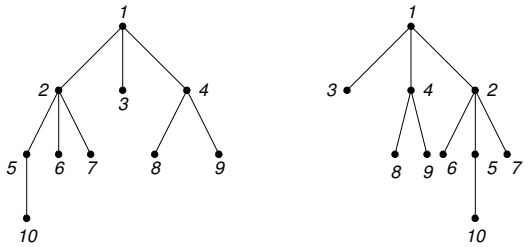
- Definisi : Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah
- Contoh:



- Tanda panah pada tree bisa dihilangkan

Pohon Terurut

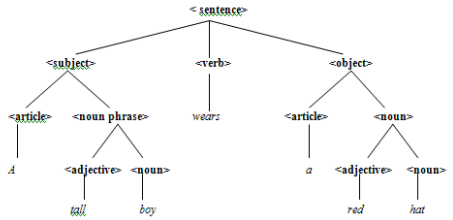
- Pohon berakar yang urutan anak-anaknya penting disebut **pohon terurut** (*ordered tree*)



17

Pohon *m-ary*

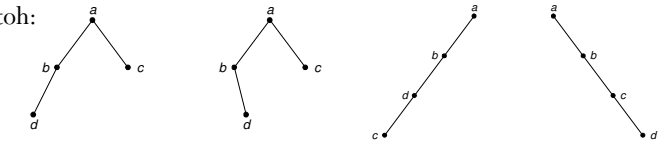
- Pohon berakar yang setiap simpul cabangnya mempunyai paling banyak *m* buah anak disebut **pohon *m-ary***.
- Jika $m = 2$, pohonnnya disebut **pohon biner** (*binary tree.*
- Pohon *m-ary* dikatakan **teratur** atau **penuh** (*full*) jika setiap simpul cabangnya mempunyai tepat *m* anak
- Contoh : Pohon parsing dari kalimat A tall boy wears a red hat



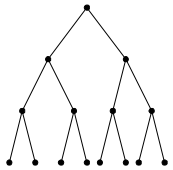
18

Pohon Biner (*Binary Tree*)

- Pohon biner Adalah pohon yang maksimum memiliki 2 cabang
- Contoh:



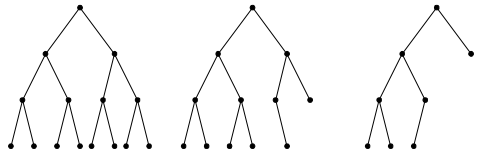
- Pohon Biner penuh adalah pohon biner yang semua cabangnya terisi
- Contoh:



19

Pohon Biner (*Binary Tree*)

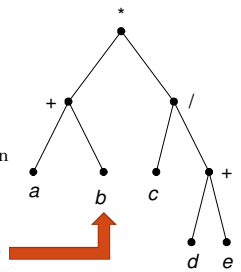
- **Pohon Biner Seimbang**
  - Pada beberapa aplikasi, diinginkan tinggi sub graph kiri dan tinggi sub graph kanan yang seimbang, yaitu berbeda maksimal 1
- Contoh:



20

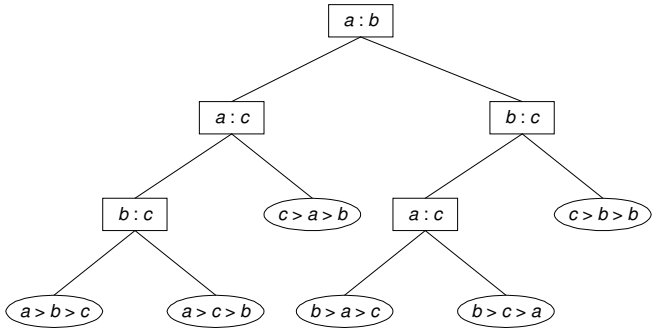
Implementasi Pohon Biner (1)

- Pohon ekspresi,
  - Pembentukan Pohon Ekspresi
    1. Bwt arrray P[n], dimana n adalah jumlah simbol u/ dibentuk pohon ekspresi
    2. Baca simbol secara postfix dan siapkan sebuah stack untuk proses pembentukan pohon ekspresi
    3. FOR i= 1 TO n DO
       a. Jika P[i]=operand
          - Bwt 1 simpul u/ P[i]
          - Push pointer ke Stack
       b. Jika P[i]=operator
          - Bwt pohon dengan P[i] sebagai akar, kemudian pop operand yang ada dlm stack u/ dijadikan sebagai simpul anak
  - Contoh pohon ekspresi dari $(a + b)*(c/(d + e))$



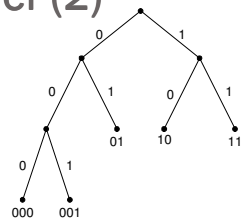
Implementasi Pohon Biner (1)

- Pohon Keputusan
  - Contoh Pohon keputusan untuk mengurutkan 3 buah elemen



Implementasi Pohon Biner (2)

- Kode Awalan
  - Contoh Pohon biner dari kode prefiks { 000, 001, 01, 10, 11}



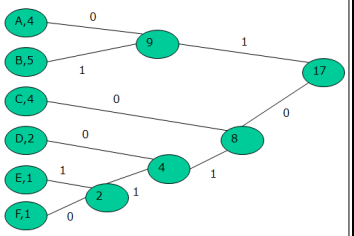
- Kode Huffman (Huffman Code's), dengan pertimbangan:
  - *The more frequently occurring symbols can be allocated with shorter codewords than the less frequently occurring symbols*
  - *The two least frequently occurring symbols will have codewords of the same length, and they differ only in the least significant bit.*

Implementasi Pohon Biner (3)

- Algoritma Huffman Code's dalam *pseudo code*
  1. Hitunglah probabilitas dari setiap simbol yang ada
  2. Pasangkan setiap <simbol,probabilitas> dengan node
  3. Temukan 2 buah node dengan probabilitas terendah kemudian buatkah node parent dengan probabilitas gabungan dari 2 anaknya
  4. Berikan label untuk cabang dari anak ke parent dengan 0 dan 1 (sebaiknya konsisten)
  5. Update node (node anak diabaikan), lalu periksa jumlah node yang ada, bila jumlah node > 1 maka ulangi langkah 3
  6. Untuk menemukan kode setiap simbol, lakukan *traverse* dari root ke leaf, (label branch yang dilalui akan menjadi kode untuk leaf)

Implementasi Pohon Biner (4)

- Contoh Huffman Code's
  - Misalkan data: S= AABAACCCCDDBBBBEF
  - Hitung frekuensi data ≈ probalitas
    - a → 4, b → 5, c → 4, d → 2, e → 1, f → 1
  - Huffman Code's yang dihasilkan



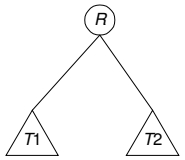
  - Kode untuk setiap simbol:

A → 10 B → 11 C → 00
D → 010 E → 0111 F → 0110

Jadi data S= AABAACCCCDDBBBBEF(17 byte)
akan dikodekan menjadi :
10 10 11 10 10 00 00 00 00 010 010 11 11 11 11 0111 0110 = 40 bit ≈ 5 byte

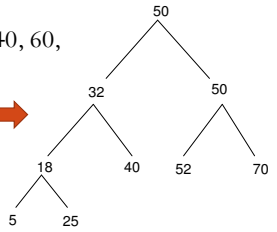
Implementasi Pohon Biner (5)

- Pohon Pencarian Biner (Binary Search Tree)



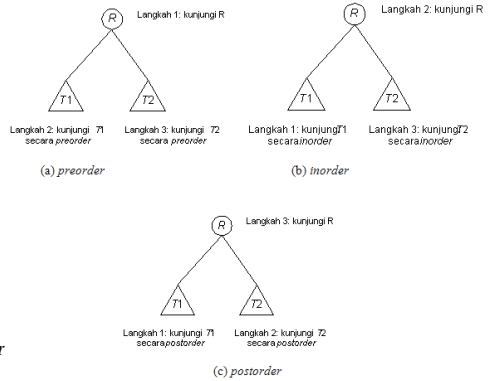
Kunci(*T1*) < Kunci(*R*)
Kunci(*T2*) > Kunci(*R*)

- Contoh:
  - Diketahui nilai-nilai sbb : 50, 32, 18, 40, 60, 52, 5, 25, 70
  - Binary Tree yang dihasilkan



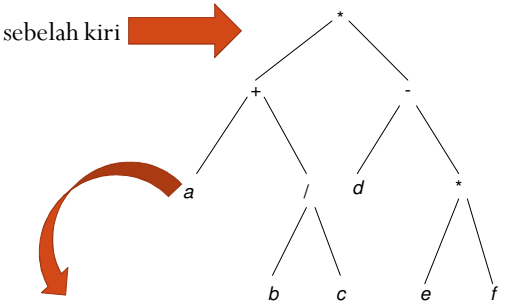
Penelusuran Pohon Biner (1)

- *Preorder* : R, T_1, T_2
  - kunjungi R
  - kunjungi T_1 secara *preorder*
  - kunjungi T_2 secara *preorder*
- *Inorder* : T_1 , R, T_2
  - kunjungi T_1 secara *inorder*
  - kunjungi R
  - kunjungi T_2 secara *inorder*
- *Postorder* : T_1, T_2 , R
  - kunjungi T_1 secara *postorder*
  - kunjungi T_2 secara *postorder*
  - kunjungi R



Penelusuran Pohon Biner (2)

- Diketahui tree di sebelah kiri



- Hasil Penelusurannya adalah
  - *preorder* : * + a / b c - d * e f (*prefix* → akar-kiri-kanan)
  - *inorder* : a + b / c * d - e * f (*infix* → kiri-akar-kanan)
  - *postorder* : a b c / + d e f * - * (*postfix* → kiri-kanan-akar)

Referensi

1. Rinaldi Munir, "Materi Kuliah Matematika Diskrit",Informatika-ITB, Bandung,2003
2. Rinaldi Munir, "Matematika Diskrit",Informatika, Bandung,2001

29