MODUL KULIAH KE → 13

ALGORITMA dan STRUKTUR DATA (3 SKS)

MATERI KULIAH :

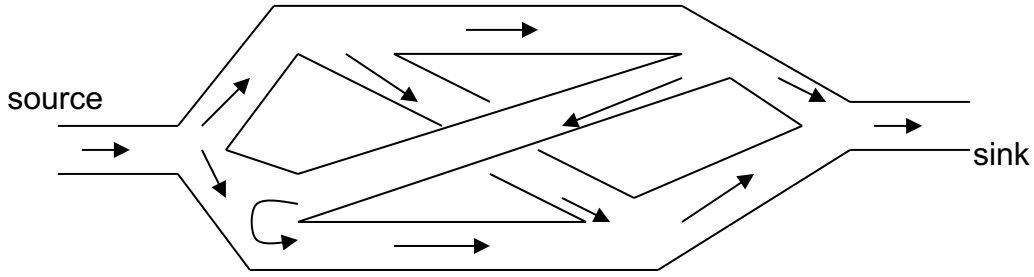
Flow networks, Minimum Spanning Tree, Minimum Length Encoding

POKOK BAHASAN :

1. *Flow Networks*

Oleh : Endang Sri Rahayu, Ir

Kita telah dapat memodelkan peta jalan sebagai *directed graph* dengan tujuan untuk menemukan jalur terpendek dari satu titik ke titik yang lain. Kita juga dapat menafsirkan directed *graph* sebagai *flow network* dan menggunakannya untuk menjawab pertanyaan tentang arus material. Bayangkan material pada sistem berasal dari *source* dan dihasilkan pada *sink.* Source menghasilkan material pada kecepatan yang tetap dan sink menerima material dengan kecepatan yang tetap sama.



Contoh persoalan :

- Cairan mengalir melalui jaringan pipa
- Aliran dari komponen pada "*assembly line*"
- Arus listrik pada jaringan listrik
- Aliran informasi pada jaringan komunikasi

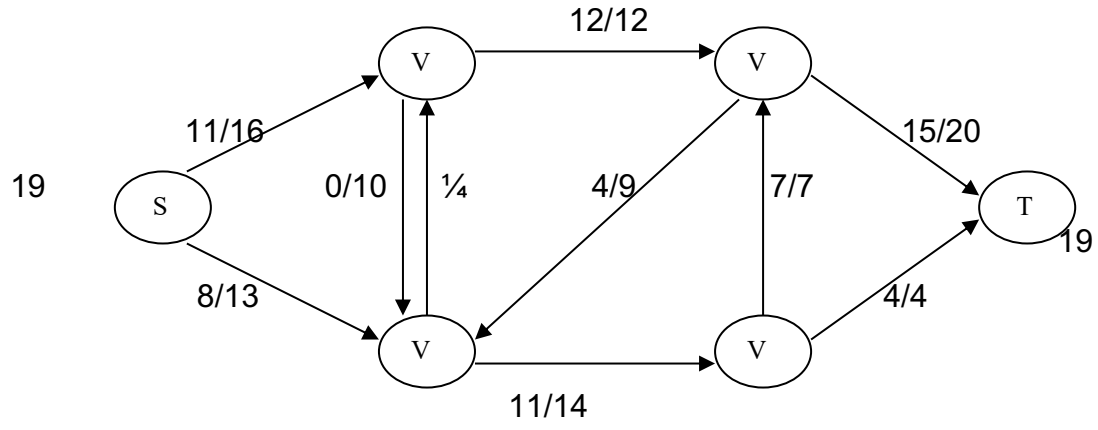
Definisi

- *Flow Network* : *Graph* berarah dengan bobot pada setiap *edge*. Memiliki "*source*" (verteks tanpa *in-edge*) dan "*sink*" (verteks tanpa *out-edge*)
- Kapasitas dari *edge* : Bobot pada suatu *edge* (non-negatif)
- *Flow* : bobot lain pada suatu *edge*

- *Flow* pada suatu *edge* lebih kecil atau sama dengan kapasitas dari *edge*
- *Flow* ke suatu verteks ≡ *flow* dari verteks tsb.
- "*Value*" dari suatu *flow* = *flow* ke *source* atau flow keluar dari sink

Network Flow Problem :

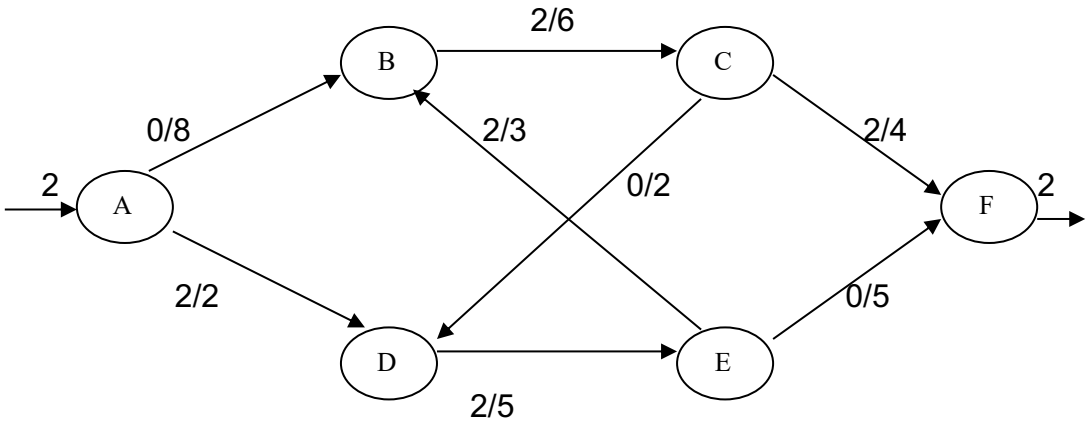
Mencari *value* dari *flow* yang maksimum pada suatu *network* yang diberikan.

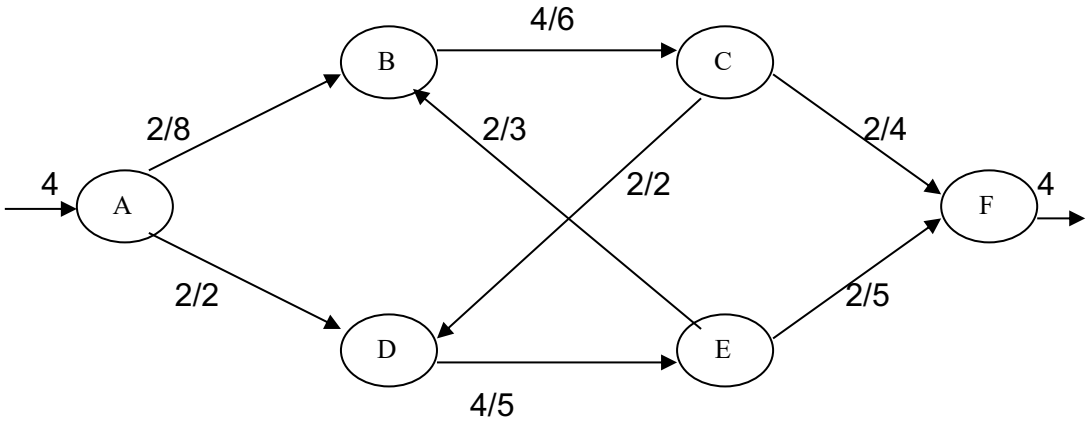


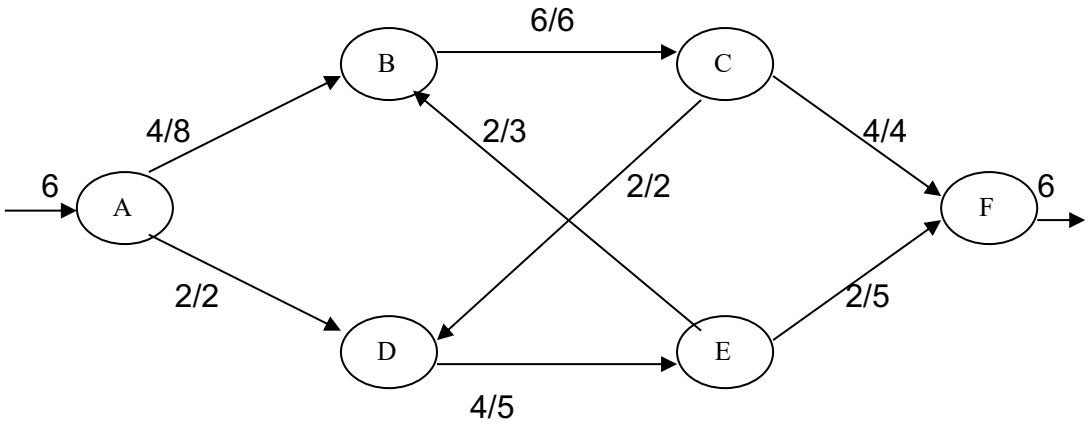
Bagaimana mencari solusinya ?

Metoda Ford-Fulkerson (sekitar tahun 1962)

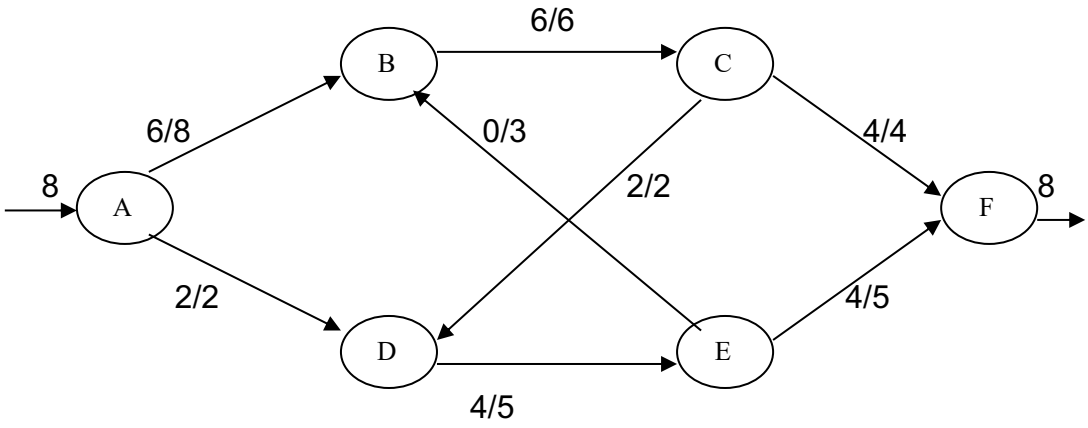
- ✓ Mulai dengan value dari *flow*=0
- ✓ Tambah *value* dari *flow* selama masih mungkin dengan memakai *edge* yang ada (dengan mengingat sisa kapasitas yang ada)







- ✓ Jika *value* dari *flow* belum maksimum, modifikasi dengan menambah *flow* dari *source* ke *sink* dan mengurangi *flow* dari *sink* ke *source*.



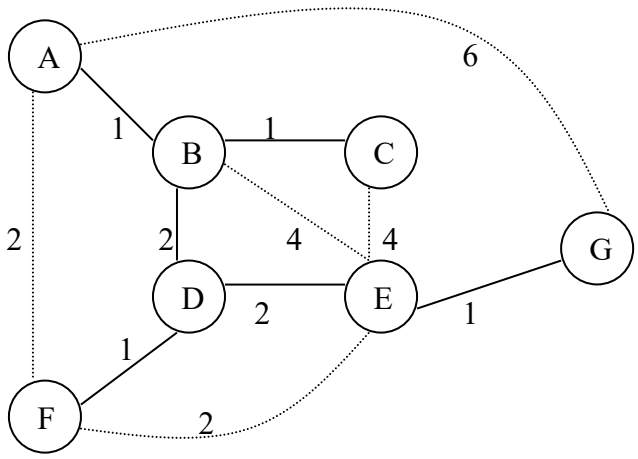
2. *MiNimum Spanning Tree*

- *Minimum Spanning Tree* dari suatu *graph* berbobot adalah kumpulan *edge* yang menghubungkan semua verteks dari *graph* tersebut sedemikan sehingga jumlah bobot dari kumpulan *edge* tersebut paling minimum.
- *Minimum Spanning Tree* merupakan salah satu jenis algoritma pada *graph*

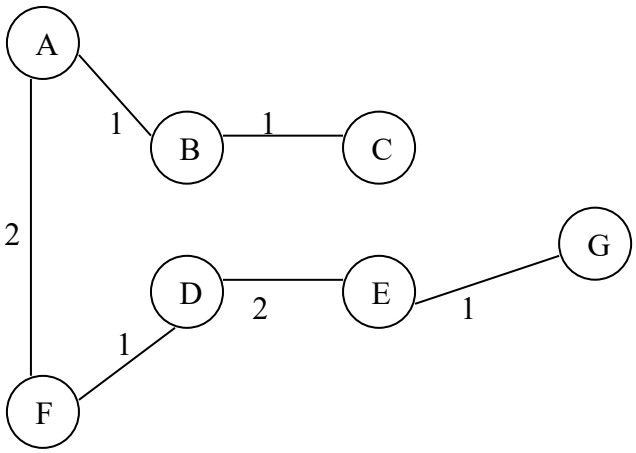
Minimum Spanning Tree Problem

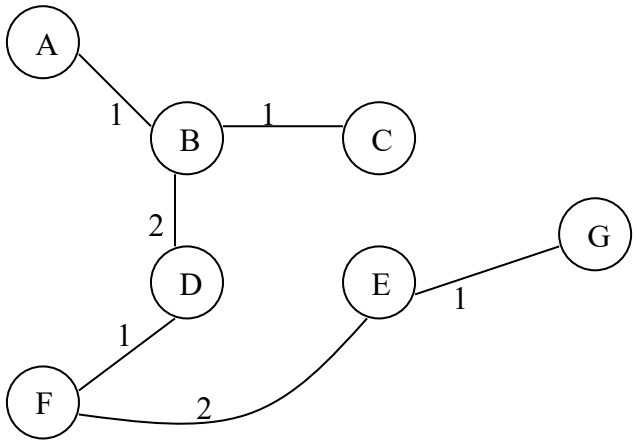
Mencari jarak terpendek untuk menghubungkan semua titik didalam suatu graph.

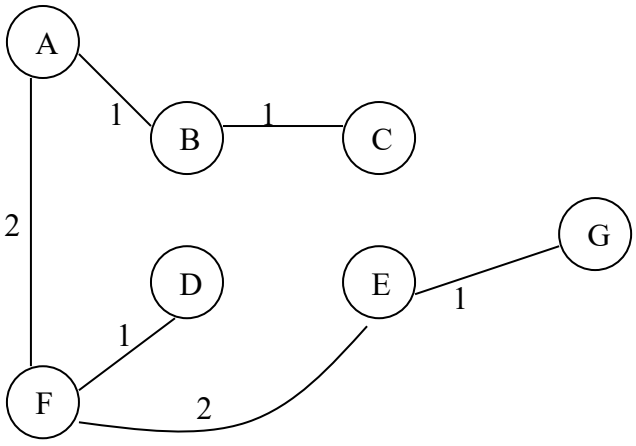
Contoh :



Minimum Spanning Tree tidak unik, contoh *minimum spanning tree* yang lain dari contoh diatas :


A
1
B
1
C
2
G
D
2
E
1
1
F

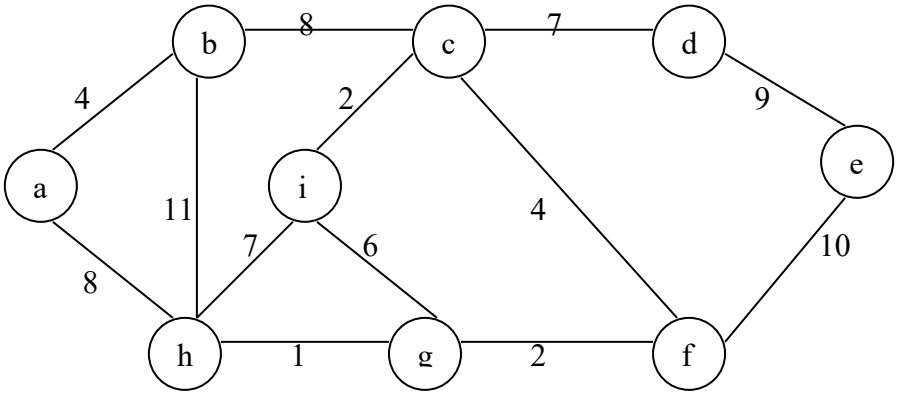

A
1
B
1
C
2
G
D
E
1
1
F
2


A
1
B
1
C
2
G
D
E
1
1
F
2

Euclidean Minimum Spanning Tree

Jika diberikan N buah titik di sebuah bidang , *Euclidean Minimum Spanning Tree* adalah kumpulan garis paling pendek yang menghubungkan N buah titik tersebut.

Contoh Soal :
Cari *minimum spanning tree* pada *graph* berikut ini :
(kunci : 37)



3. *Minimum Length Encoding*

- *Encode* : pengkodean yang tidak memperhatikan kerahasian
- *Encrypt* : sistem pengkodean yang membutuhkan perhatian masalah kerahasian
- Karakter biasanya di*encode* sebagai suatu "*string of bits*"

Fixed Length Character Code

⇒ Mudah mengenali akhir suatu karakter dan awal dari karakter berikutnya.

⇒ "*Space*" dapat menjadi besar

⇒ Kode untuk karakter yang jarang dipakai sama panjangnya dengan kode untuk karakter yang sering dipakai.

Idealnya :

Karakter yang sering muncul memiliki kode yang pendek.

→ Diinginkan meminimalkan panjang dari kode-kode karakter yang ada (untuk menghemat *space*) → diperlukan informasi frekuensi kemunculan dari karakter-karakter yang ada.

Minimum Length Encoding Problem

> Dipunyai suatu kumpulan dari karakter, dengan masing-masing karakter mempunyai probabilitas untuk muncul dalam suatu teks.
> Dicari cara bagaimana membuat suatu kode sehingga *encoding* dan *decoding* setiap karakter dapat dilakukan dengan pasti dan kode tersebut menghasilkan panjang yang minimal.

Ilustrasi :

⇒ Harus dihindari masalah ketidakpastian dalam mengenali kode dari suatu karakter.

⇒ Misalnya bila dikodekan A=00 dan B=000, maka string berbentuk 000000 dapat diartikan AAA atau BB.

⇒ Dalam hal ini dikatakan kode dari A merupakan *prefix* dari kode dari B, atau cukup dikatakan A merupakan prefix dari B.

Beberapa Istilah :

x, y adalah suatu bit string atau karakter string

- x adalah ***prefix*** dari y ←→ terdapat string z sehingga xz = y
- x adalah ***suffix*** dari y ←→ terdapat string z sehingga y = zx

Suatu kode (sistem pengkodean) dikatakan mempunyai *prefix property* (*prefix code*) ←→ tidak benar bahwa kode untuk suatu karakter merupakan *prefix* dari kode untuk karakter yang lain.

Pada contoh sebelumnya, untuk A=00 dan B=000 dikatakan sistem pengkodean ini tidak memiliki ***prefix property.***

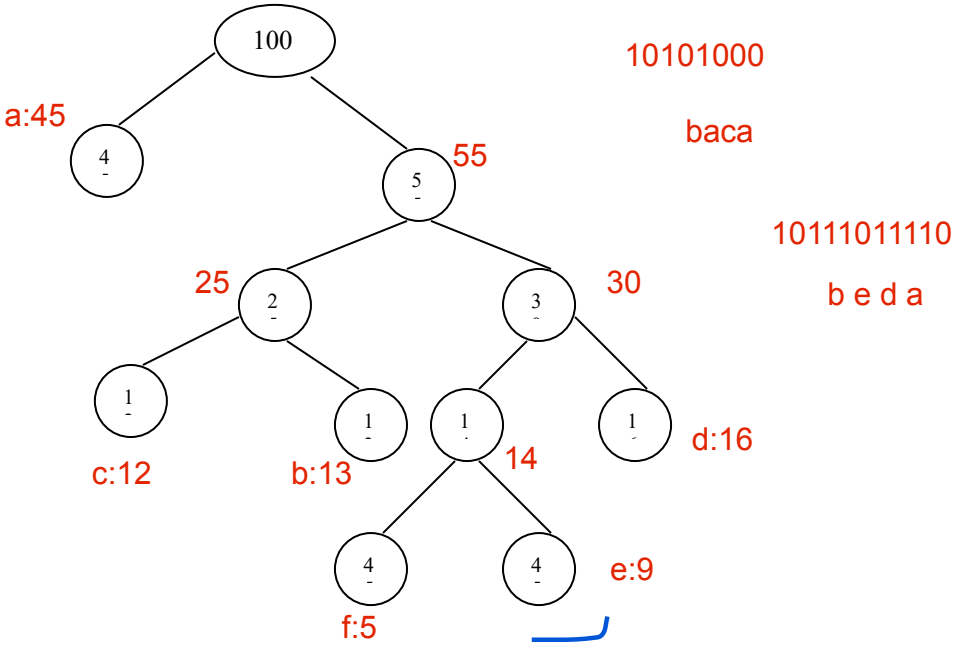
Contoh lain :

Dipunyai suatu berkas data yang terdiri dari 100.000 karakter dengan deskripsi sebagai berikut :

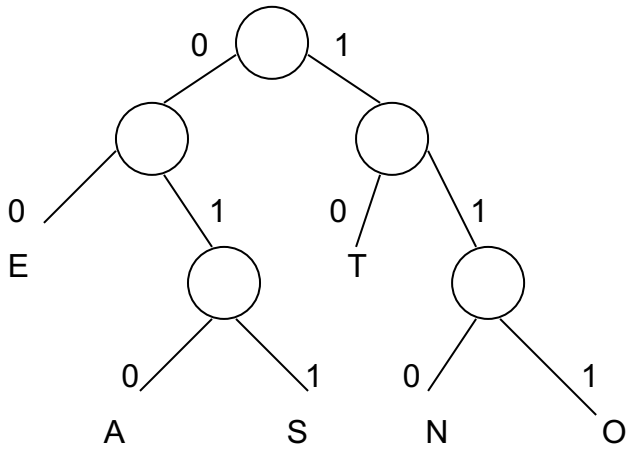
Karakter	a	b	c	d	e	f
Frekuensi (ribuan)	45	13	12	16	9	5
Fixed-length code	000	001	010	011	100	101
Variable-length code	0	101	100	111	1101	1100
	1	3	3	3	4	4

Dengan 3 bit diperlukan 300.000 bit sedangkan dengan *variable length code* diperlukan 224.000 bit.

Dengan *variable length code* diperoleh 25% penghematan



- ✓ Suatu sistem pengkodean yang memiliki *prefix property* (*prefix code*) mempunyai representasi yang natural berupa tree biner (*binary tree*)
- ✓ Setiap node dikaitkan dengan suatu bit string
- ✓ Root dikaitkan dengan *empty* string



Pengkodean

E = 00	S = 011
N = 110	A = 010
T = 10	O = 111

Algoritma untuk membaca kembali (*Decoding*)

1) Mulai pada *root*
2) 0 berarti pergi ke anak kiri
3) 1 berarti pergi ke anak kanan
4) Jika mencapai *leaf,* cetak karakter pada *leaf* dan kembali ke *root*

Observasi

Karakter yang sering muncul ada di level atas (dengan demikian mempunyai kode yang pendek)

Dengan pendekatan ***divide & conquer*** akan diperiksa karakter mana yang harus diletakkan di subtree kiri atau di subtree kanan → pendekatan ini tidak praktis.

Pendekatan yang lebih baik → membentuk *tree* dari bawah ke atas, mulai dari karakter dengan frekuensi yang rendah

Catatan

Divide and Qonquer : ✓

Untuk menyelesaikan suatu persoalan, bagi masalah tersebut menjadi masalah-masalah yang lebih kecil yang dapat diselesaikan secara independen.

Kesimpulan *Minimum Length Encoding*

Jika dipunyai n karakter, yaitu $\{C_i\}$ dengan i = 1, 2, 3, …, n, masing-masing karakter mempunyai probabilitas p_i dan panjang kode dari masing-masing

karakter adalah s_i maka ekspektasi panjang total dari kode seluruh karakter tersebut adalah :

$$\sum p_i s_i$$

→ jumlah ini yang akan dibuat minimal !

REFERENSI

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, **Introduction to Algoritms**, Mc Graw–Hill, 1990
- Dr W. Nugroho, **Bahan Kuliah Desain Analisis Algoritma**, Fasilkom, Universitas Indonesia, 2001