Bahan Kuliah IF2211 Strategi Algoritma

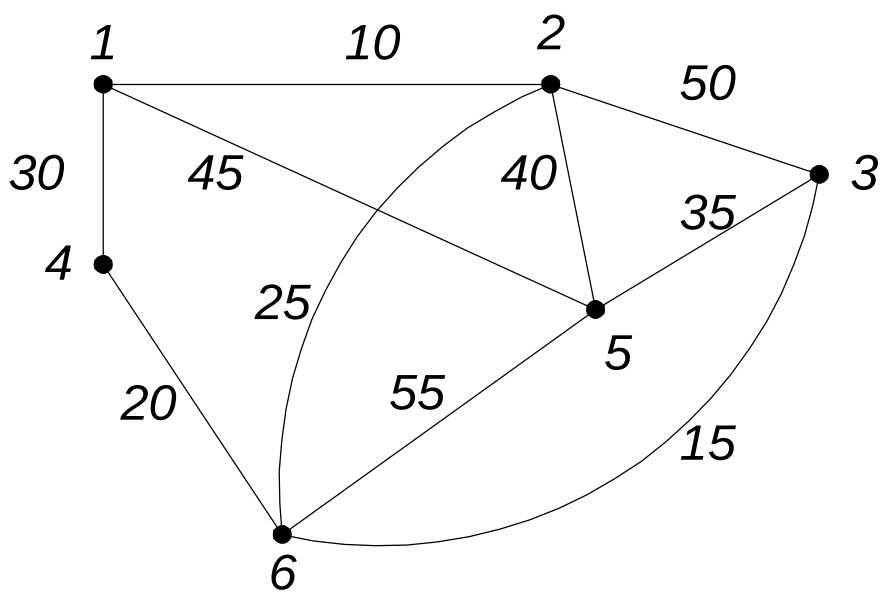
Algoritma *Greedy*

(Bagian 2)

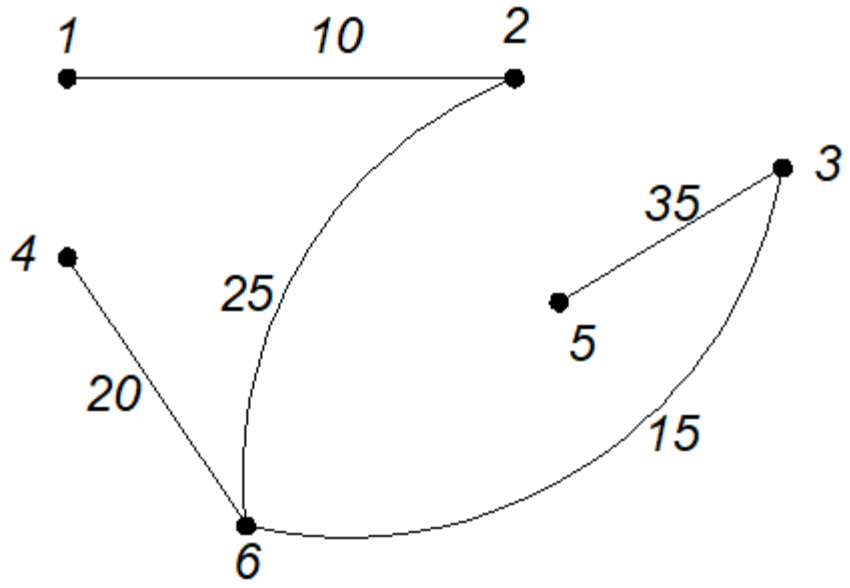
Oleh: Rinaldi Munir

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika ITB
2021

7. Pohon Merentang Minimum



(a) Graf *G = (V, E)*



(b) Pohon merentang minimum

(a) Algoritma Prim

- Misalkan pohon merentang minimum yang dibangun adalah himpunan *T*.

- Strategi *greedy* yang digunakan di dalam Algoritma Prim:

> "Pada setiap langkah, pilih sisi $e = (v_1, v_2)$ dari graf $G(V, E)$ yang memiliki bobot terkecil dan bersisian (*incidency*) dengan simpul-simpul di *T* tetapi *e* tidak membentuk sirkuit di *T*. Masukkan *e* ke dalam *T*. "

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.

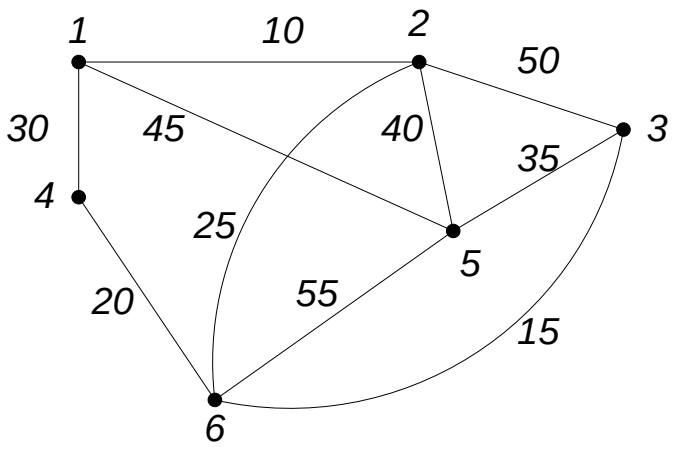
procedure *Prim*(**input** *G* : *graf*, **output** *T* : *pohon*)
{ Membentuk pohon merentang minimum T dari graf berbobot G.
Masukan: graf-berbobot terhubung G = (V, E), dengan $|V| = n$
Luaran: pohon rentang minimum T = (V, E') }

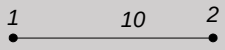
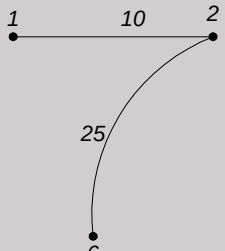
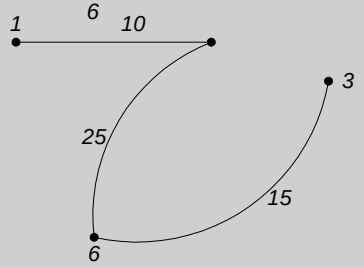
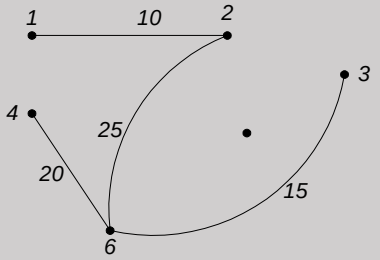
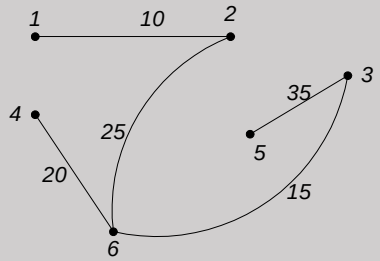
Deklarasi
i, *p*, *q*, *u*, *v* : **integer**

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

Komplesitas algoritma: $O(n^2)$

Contoh 12:



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	

(b) Algoritma Kruskal

- Urutkan terlebih dahulu sisi-sisi di dalam graf berdasarkan bobotnya dari kecil ke besar

- Strategi *greedy* yang digunakan:

  "Pada setiap langkah, pilih sisi $e = (v_1, v_2)$ dari graf $G = (V, E)$ yang memiliki bobot minimum. Jika e tidak membentuk sirkuit di T, maka masukkan e ke dalam T"

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.

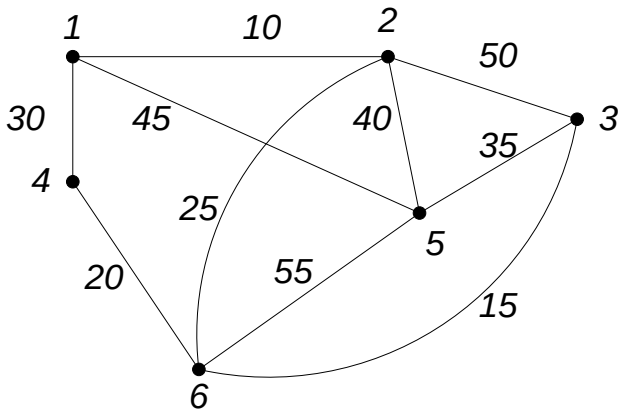
procedure *Kruskal*(**input** *G* : *graf*, **output** *T* : *pohon*)
{ Membentuk pohon merentang minimum T dari graf berbobot G.
Masukan: graf-berbobot terhubung G = (V, E), dengan $|V| = n$
Luaran: pohon rentang minimum T = (V, E') }

Deklarasi
i, *u*, *v* : integer

Algoritma
{ Asumsi: sisi-sisi dari graf sudah diurut menaik berdasarkan bobotnya dari kecil ke besar}
$T \leftarrow \{\}$
while jumlah sisi di dalam $T < n - 1$ **do**
Pilih sisi (*u*, *v*) dari *E* yang bobotnya terkecil
if (*u*, v) tidak membentuk sirkuit di *T* **then**
$T \leftarrow T \cup \{(u, v)\}$
endif
endfor

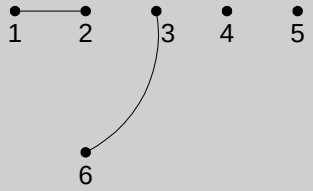
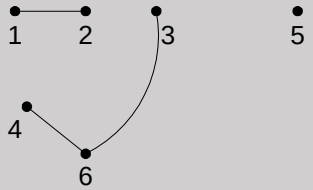
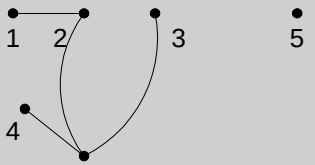
Kompleksitas algoritma: $O(|E| \log |E|)$

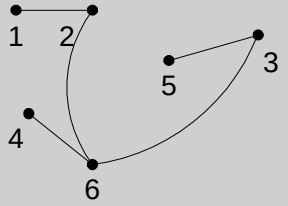
Contoh 13:



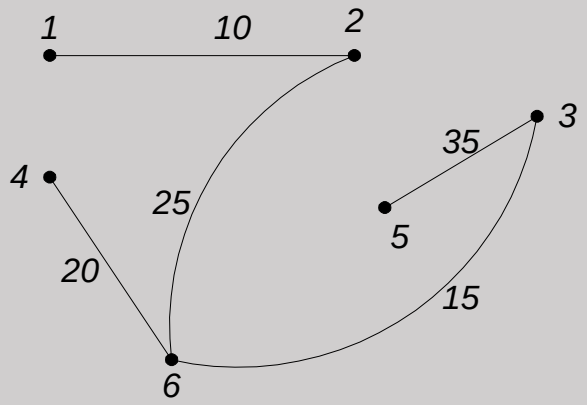
Sisi-sisi diurut menaik:

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	

5	(1, 4)	30	ditolak
6	(3, 5)	35	

Pohon merentang minimum yang dihasilkan:



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

Theorem 4.6 Kruskal's algorithm generates a minimum-cost spanning tree for every connected undirected graph G.

Proof: Let G be any undirected connected graph. Let t be the spanning tree for G generated by Kruskal's algorithm. Let t' be a minimum-cost spanning tree for G. We show that both t and t' have the same cost.

Let $E(t)$ and $E(t')$ respectively be the edges in t and t'. If n is the number of vertices in G, then both t and t' have $n-1$ edges. If $E(t) = E(t')$, then t is clearly of minimum cost. If $E(t) \neq E(t')$, then let q be a minimum-cost edge such that $q \in E(t)$ and $q \notin E(t')$. Clearly, such a q must exist. The inclusion of q into t' creates a unique cycle (Exercise 5). Let $q, e_1, e_2, \ldots, e_k$ be this unique cycle. At least one of the e_i's, $1 \leq i \leq k$, is not in $E(t)$ as otherwise t would also contain the cycle $q, e_1, e_2, \ldots, e_k$. Let e_j be an edge on this cycle such that $e_j \notin E(t)$. If e_j is of lower cost than q, then Kruskal's algorithm will consider e_j before q and include e_j into t. To see this, note that all edges in $E(t)$ of cost less than the cost of q are also in $E(t')$ and do not form a cycle with e_j. So $cost(e_j) \geq cost(q)$.

Now, reconsider the graph with edge set $E(t') \cup \{q\}$. Removal of any edge on the cycle $q, e_1, e_2, \ldots, e_k$ will leave behind a tree t'' (Exercise 5). In particular, if we delete the edge e_j, then the resulting tree t'' will have a cost no more than the cost of t' (as $cost(e_j) \geq cost(e)$). Hence, t'' is also a minimum-cost tree.

By repeatedly using the transformation described above, tree t' can be transformed into the spanning tree t without any increase in cost. Hence, t is a minimum-cost spanning tree. □

(Sumber: Ellis Horrowitz & Sartaj Sahni, *Computer Algorithms*, 1998)

8. Lintasan Terpendek (*Shortest Path)*

Beberapa macam persoalan lintasan terpendek:

a) Lintasan terpendek antara dua buah simpul tertentu (*a pair shortest path*).

b) Lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*).

c) Lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*single-source shortest path*).

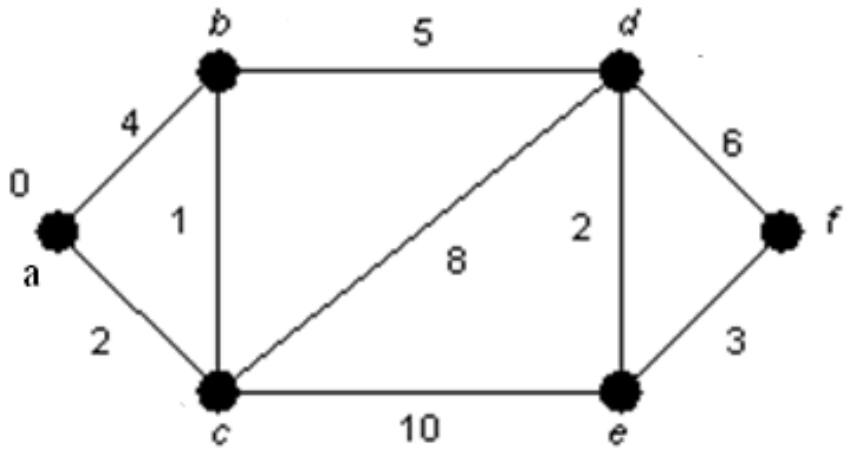
d) Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).

→ Yang akan dibahas adalah persoalan c)

Persoalan lintasan terpendek:

Diberikan graf berbobot $G = (V, E)$. Tentukan lintasan terpendek dari sebuah simpul asal a ke setiap simpul lainnya di G.

Asumsikan semua sisi di dalam graf berbobot positif.



Berapa jarak terpendek berikut lintasannya dari:

a ke b?

a ke c?

a ke d?

a ke e?

a ke f?

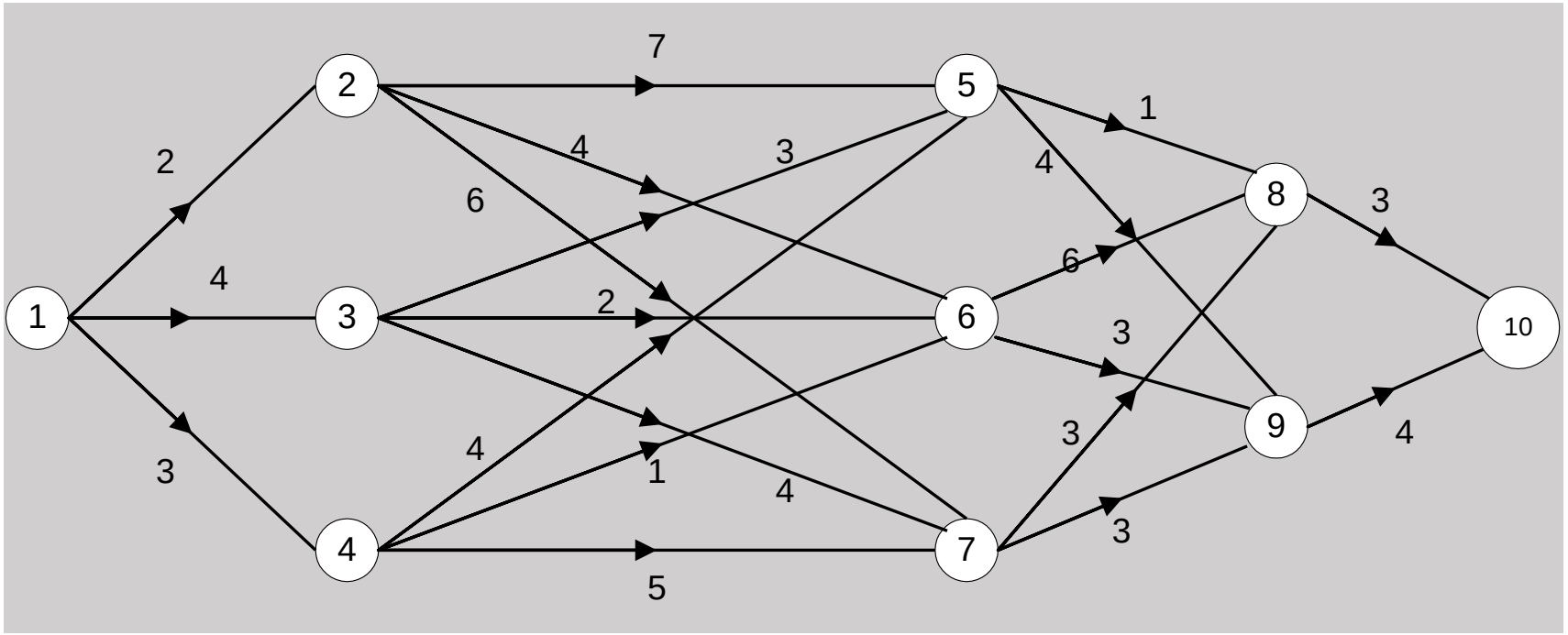
Penyelesaian dengan Algoritma *Brute Force*

- Misalkan ingin menentukan jarak terpendek dari *a* ke *b*
- Enumerasi semua lintasan yang mungkin dibentuk dari *a* ke *b*, hitung total bobotnya
- Lintasan yang memiliki bobot terkecil adalah lintasan terpendek dari *a* ke *b*
- Ulangi cara yang sama untuk jarak terpendek dari *a* ke *c*, dari *a* ke *d*, dan seterusnya.

Penyelesaian dengan Algoritma Greedy

- Misalkan ingin menentukan jarak terpendek dari *a* ke *b*
- Strategi *greedy*: pada setiap langkah, pilih sisi (*u*, *v*) dengan bobot terkecil
- Ulangi cara yang sama untuk jarak terpendek dari *a* ke *c*, dari *a* ke *d*, dan seterusnya.

- Namun, strategi *greedy* di atas tidak selalu menjamin solusi optimal
- Contoh: Lintasan terpendek dari 1 ke 10 pada graf di bawah ini!



Greedy: 1 – 2 – 6 – 9 – 10 dengan bobot = 2 + 4 + 3 + 4 = 13 → Tidak optimal

Solusi optimal: 1 – 3 – 5 – 8 – 10 dengan bobot = 4 + 3 + 1 + 2 = 11

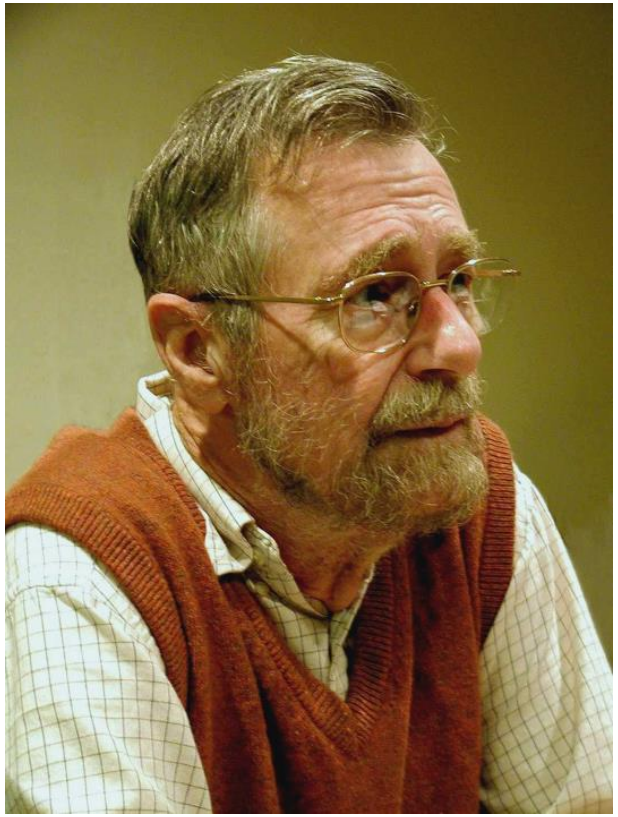
Algoritma Dijkstra

- Merupakan algoritma yang optimal untuk menentukan lintasan terpendek.

- Lintasan terpendek dibangun langkah per langkah. Pada langkah pertama bangun lintasan terpendek pertama, pada langkah kedua bangun lintasan terpendek kedua, demikian seterusnya.

- Strategi *greedy* yang digunakan:

  "Pada setiap langkah, pilih lintasan berbobot minimum yang menghubungkan simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih. Lintasan dari simpul asal ke simpul yang baru haruslah merupakan lintasan yang terpendek diantara semua lintasannya ke simpul-simpul yang belum terpilih."



Edsger W. Dijkstra (1930–2002)

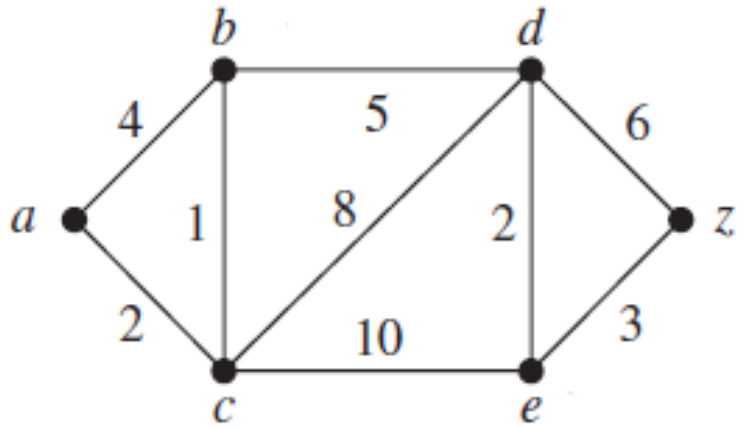
- Edsger Wybe Dijkstra was one of the most influential members of computing science's founding generation. Among the domains in which his scientific contributions are fundamental are
- algorithm design
- programming languages
- program design
- operating systems
- distributed processing

In addition, Dijkstra was intensely interested in teaching, and in the relationships between academic computing science and the software industry. During his forty-plus years as a computing scientist, which included positions in both academia and industry, Dijkstra's contributions brought him many prizes and awards, including computing science's highest honor, the ACM Turing Award.
Sumber: http://www.cs.utexas.edu/users/EWD/

procedure *Dijkstra* (**input** *G*: *weighted_graph*, **input** *a*: *intial_vertex*, **output** *L* : **array** [1..*n*] of **real**)
{ Mencari lintasan terpendek dari simpul a ke semua simpul lain di dalam graf berbobot G.
Masukan: graf-berbobot yang terhubung, G = (V, E) dengan $|V| = n$
Luaran: L[1..n], L[i] berisi panjang terpendek dari simpul a ke simpul v_i *}*

Deklarasi:
i : **integer**
u, *v* : *vertex*
S **: set of** *vertex* *{ himpunan solusi untuk mencatat simpul-simpul yang sudah dipilih di dalam tur }*
Algoritma
for *i*←1 **to** *n*
$L(v_i) \leftarrow \infty$
endfor
$L(a) \leftarrow 0$ *{ jarak dari a ke a adalah 0 }*
$S \leftarrow \{\ \}$
for $k \leftarrow 1$ to n **do**
$u \leftarrow$ pilih simpul yang belum terdapat di dalam S dan memiliki $L(u)$ minimum
$S \leftarrow S \cup \{u\}$ *{ masukkan u ke dalam S }*
for semua simpul *v* yang tidak terdapat di dalam *S*
{ update jarak yang baru dari a ke v }
if $L(u) + G(u, v) < L(v)$ **then** *{ jarak dari a ke u ditambah bobot sisi dari u ke v lebih kecil dari jarak a ke v }*
$L(v) \leftarrow L(u) + G(u, v)$ *{ jarak dari a ke v yang baru diganti dengan L(u) + G(u, v) }*
endif
endfor
enfor

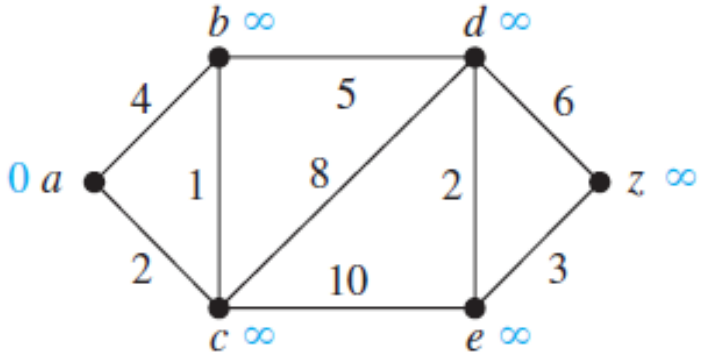
Contoh 14: Diberikan graf G di bawah ini. Carilah lintasan terpendek dari simpula a ke semua simpul lainnya.



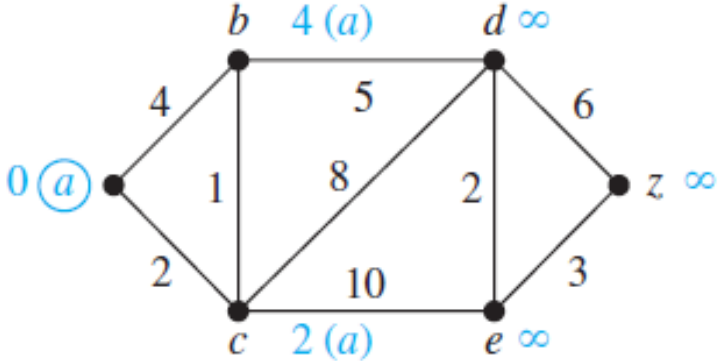
(Sumber: Rosen, *Discrete Mathematics and Its Application*, 7th Edition)

Penyelesaian:

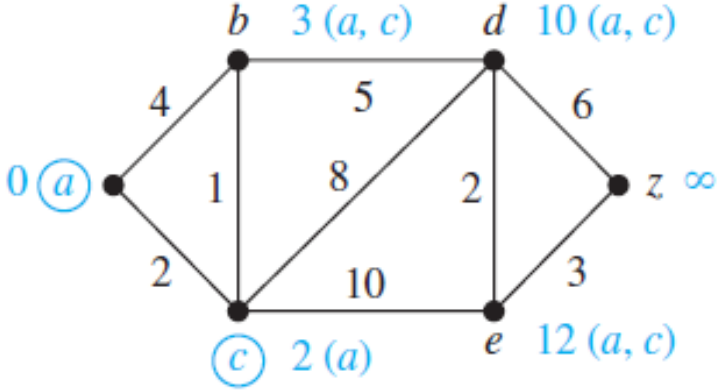
(Sumber: Rosen, *Discrete Mathematics and Its Application*, 7th Edition)



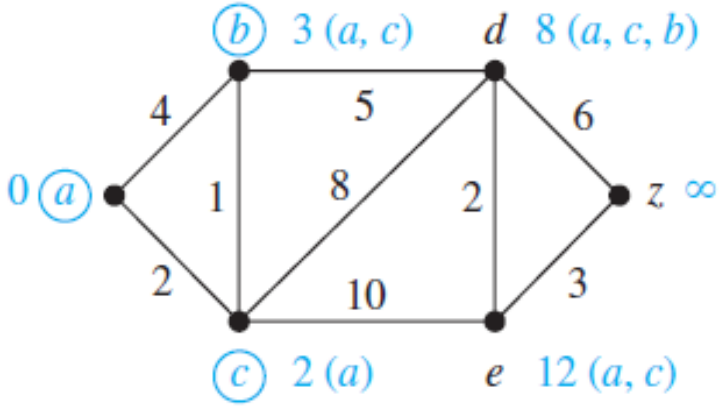
(a)



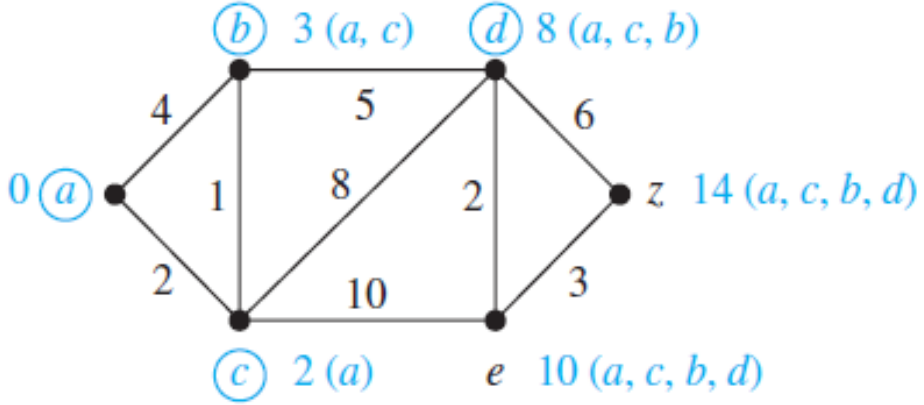
(b)



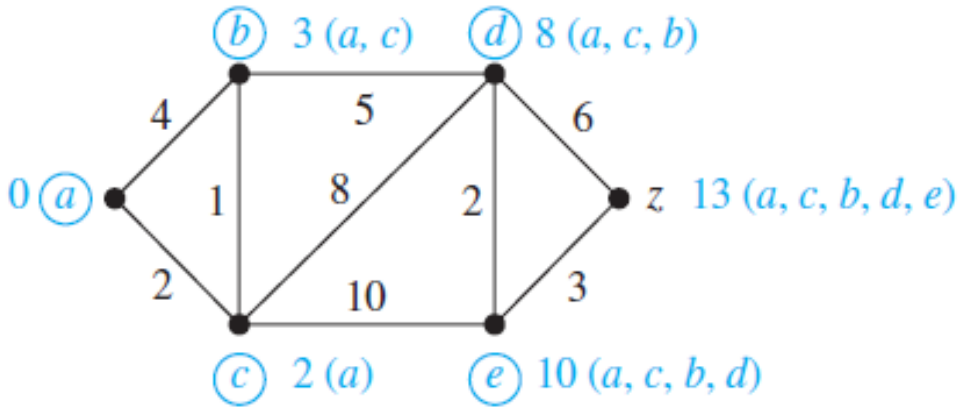
(c)



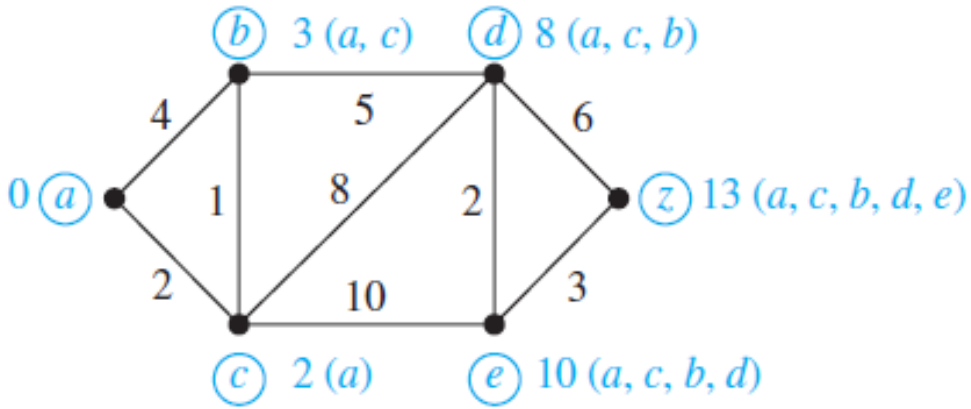
(d)



(e)



(f)



(g)

Jadi, lintasan terpendek (dimulai dari lintasan yang bobot terkecil) dari:

a ke *c* adalah *a*, *c* dengan bobot = 2

a ke *b* adalah *a*, *c*, *b* dengan bobot = 3

a ke *d* adalah *a*, *c*, *b*, *d* dengan bobot = 8

a ke *e* adalah *a*, *c*, *b*, *d*, *e* dengan bobot = 10

a ke *z* adalah *a*, *c*, *b*, *d*, *e*, *z* dengan bobot = 13

- Kompleksitas Algoritma Dijkstra ditentukan oleh kalang (*loop*) berikut:

```
for k ← 1 to n    do
    u ← pilih simpul yang belum terdapat di dalam S dan memiliki L(u) minimum
    S ← S ∪ {u}        { masukkan u ke dalam S }
    for semua simpul v yang tidak terdapat di dalam S
        { update jarak yang baru dari a ke v }
        if L(u) + G(u, v) < L(v) then
            L(v) ← L(u) + G(u, v)
        endif
    endfor
endfor
```

(i) Memilih simpul *u* yang bukan di dalam *S* dan memiliki *L*(*u*) minimum
$\rightarrow$ membutuhkan paling banyak $n - 1$ perbandingan: $O(n)$

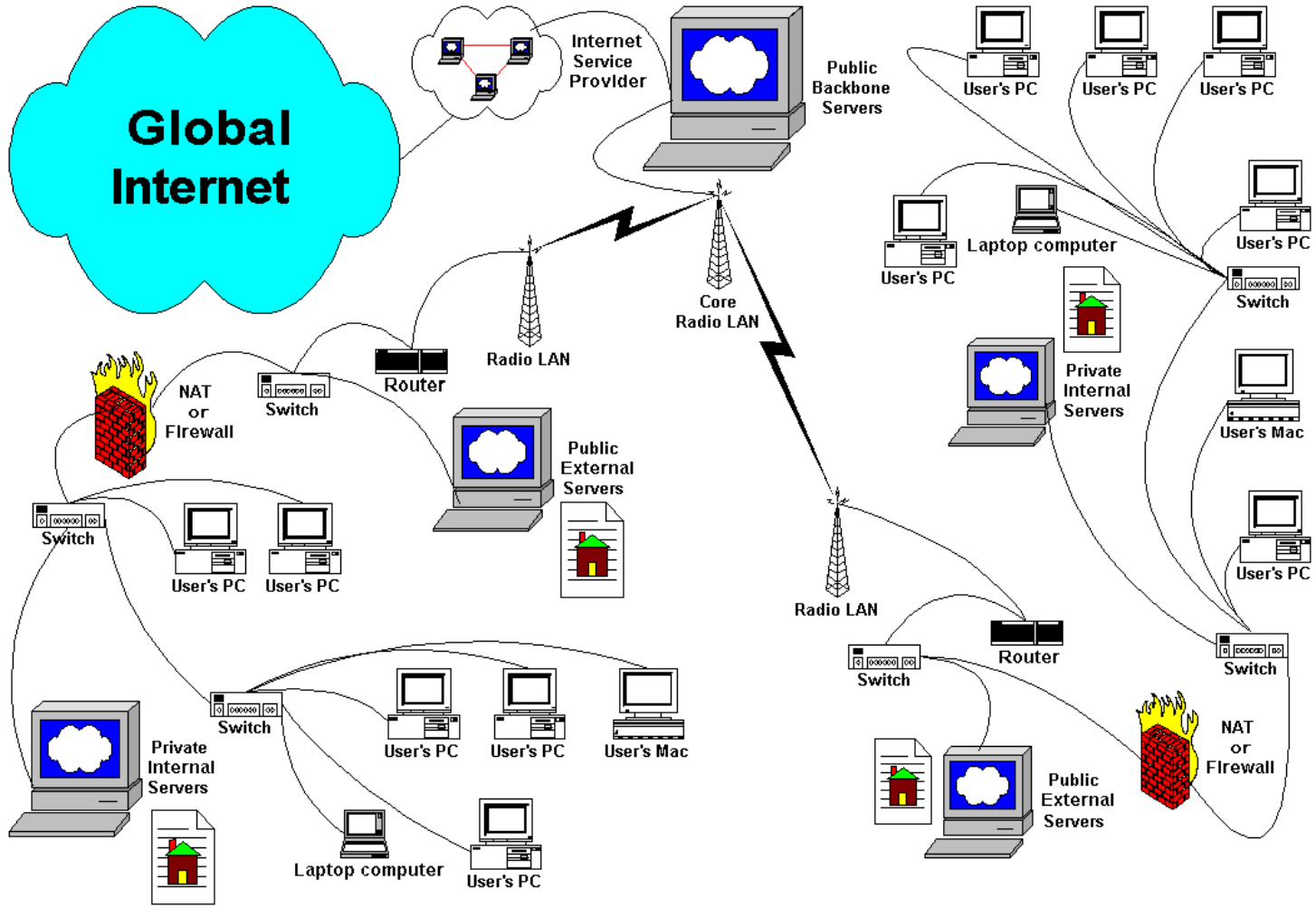
(ii) Memperbarui (*update*) jarak yang baru dari *a* ke *v*:
$\rightarrow$ membutuhkan paling banyak $n - 1$ perbandingan dan n – 1 penjumlahan: $O(n)$

(ii) Pengulangan **for** *k* dari 1 sampai *n* dilakukan sebanyak *n* kali

Kompleksitas waktu algoritma Dijkstra: $T(n) = n \{ O(n) + O(n) \} = O(n^2)$

Aplikasi algoritma Dijkstra:

→*Routing* pada jaringan komputer

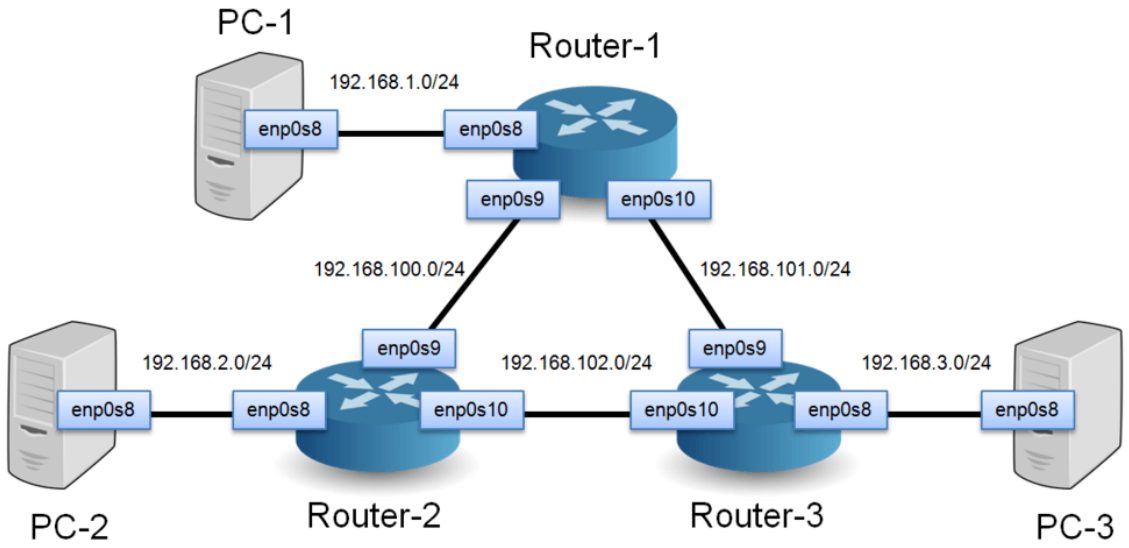


- Pesan yang dikirim dari satu komputer ke komputer lainnya umumnya dipecah menjadi sejumlah paket (*packet*) data yang berukuran lebih kecil.

- Untuk menyampaikan paket data dari dari satu komputer ke komputer lainnya, sistem jaringan komputer harus dapat melakukan pemilihan rute yang tepat agar paket dapat sampai ke komputer tujuan dalam waktu yang cepat.

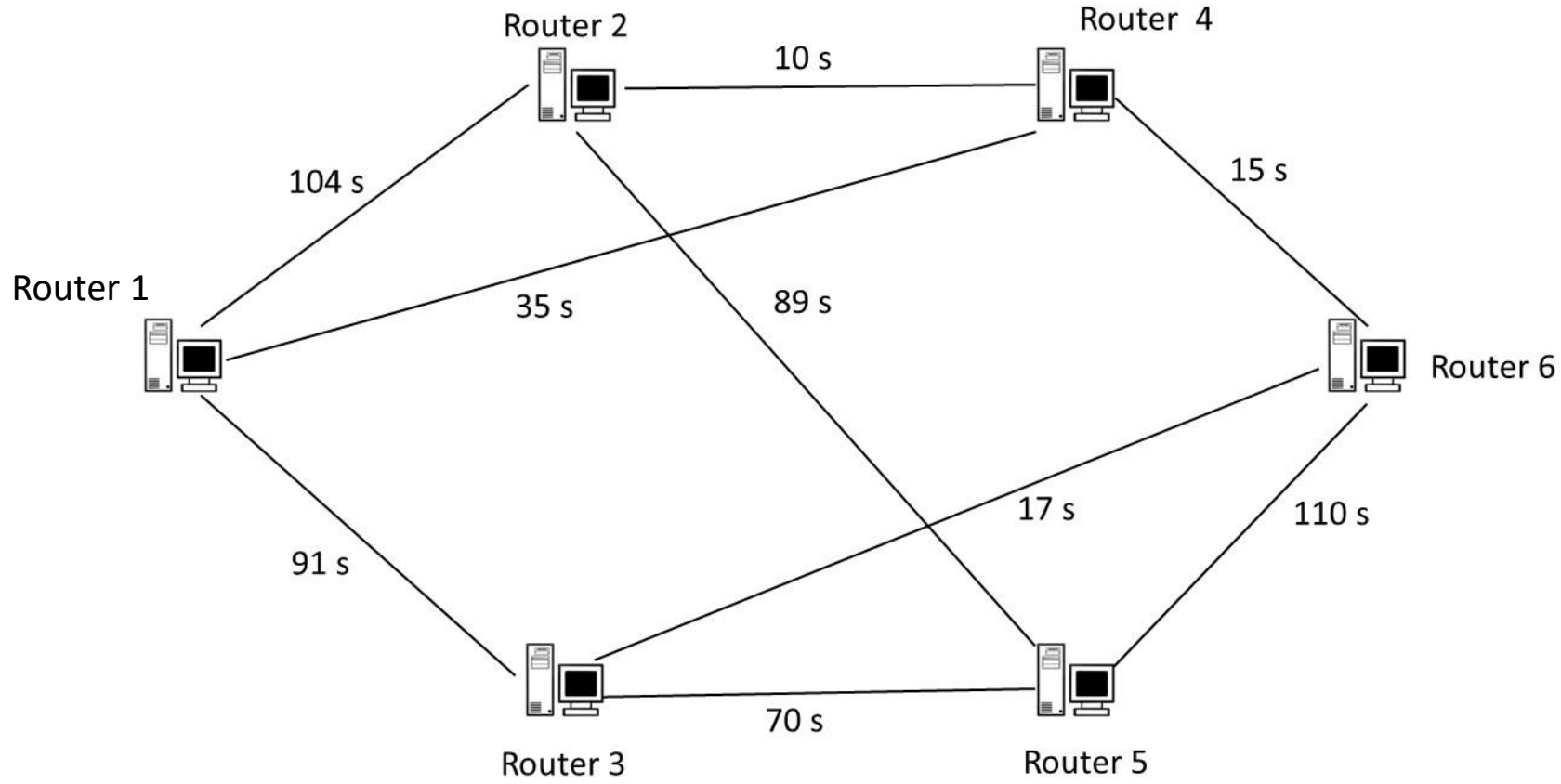
- Yang dimaksud dengan **perutean** (*routing*) adalah menentukan lintasan yang dilalui oleh paket dari komputer pengirim (asal) ke komputer penerima (tujuan).

- *Router* adalah komputer yang didedikasikan untuk mengarahkan pesan dari suatu simpul ke simpul lainnya.



- Setiap *router* memelihara sebuah tabel yang disebut tabel rute (*routing table*). Tabel rute berisi alamat komputer asal, alamat komputer tujuan, dan simpul antara (via) yang dilalui.

Contoh sebuah jaringan router:



Mencari lintasan terpendek dari *router* asal ke *router* tujuan dapat diartikan sebagai menentukan lintasan terpendek dari simpul asal ke simpul tujuan di dalam jaringan komputer.

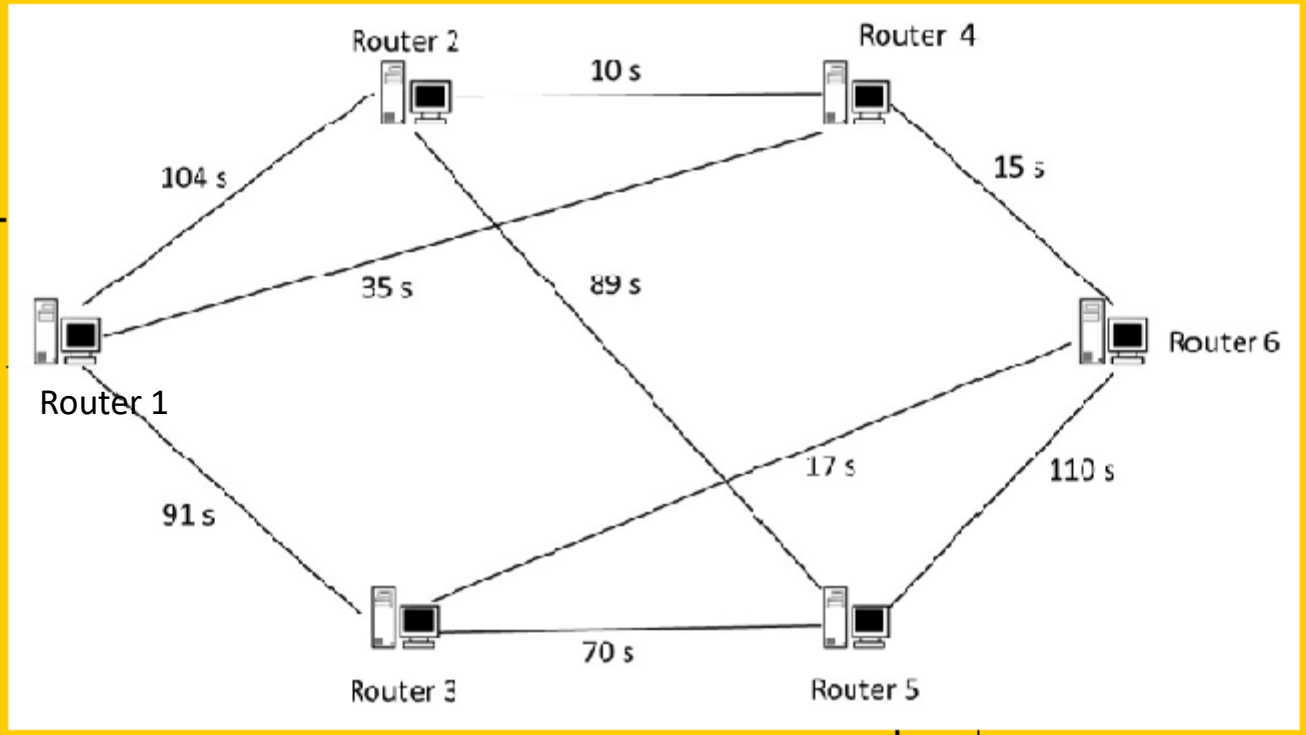
Lintasan terpendek (berdasarkan *delay time*):

Router Asal	*Router* Tujuan	Lintasan Terpendek
1	1	-
	2	1, 4, 2
	3	1, 4, 6, 3
	4	1, 4
	5	1, 4, 2, 5
	6	1, 4, 6
2	1	2, 4, 1
	2	-
	3	2, 4, 6, 3
	4	2, 4
	5	2, 5
	6	2, 4, 6
3	1	3, 6, 4, 1
	2	3, 6, 4, 2
	3	-
	4	3, 6, 4
	5	3, 5
	6	3, 6
4	1	4, 1
	2	4, 2
	3	4, 6, 2
	4	4, 6, 3
	5	4, 2, 5
	6	4, 6

Asal	Tujuan	Via
2	1	4
2	2	-
2	3	4
2	4	2
2	5	2
2	6	4

Asal	Tujuan	Via
4	1	4
4	2	4
4	3	6
4	4	-
4	5	2
4	6	4

Asal	Tujuan	Via
1	1	-
1	2	4
1	3	4
1	4	4
1	5	4
1	6	4



Asal	Tujuan	Via
6	1	4
6	2	4
6	3	6
6	4	6
6	5	3
6	6	-

Asal	Tujuan	Via
3	1	6
3	2	6
3	3	-
3	4	6
3	5	3
3	6	3

Asal	Tujuan	Via
5	1	2
5	2	5
5	3	5
5	4	2
5	5	-
5	6	3

Bersambung ke bagian 3