Bahan Kuliah IF2211 Strategi Algoritma

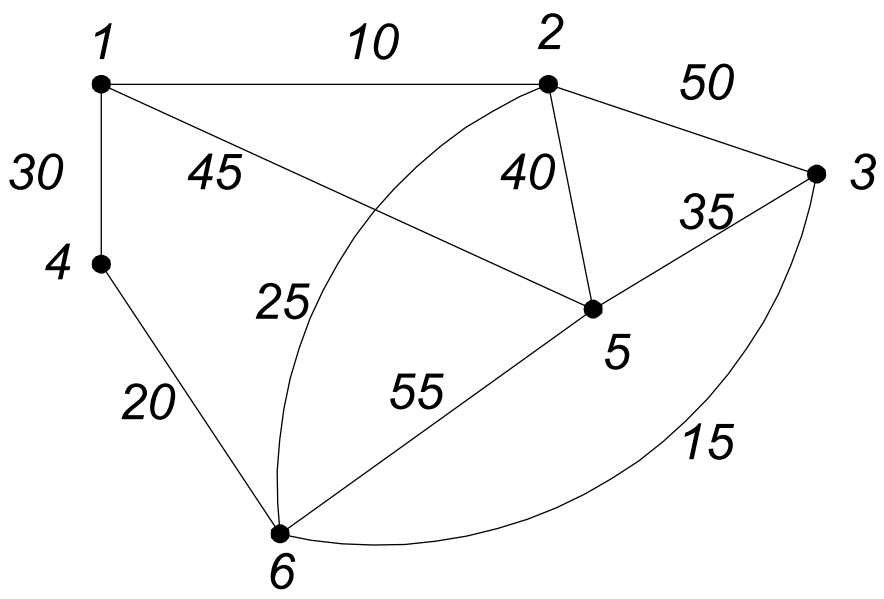
Algoritma *Greedy*

(Bagian 2)

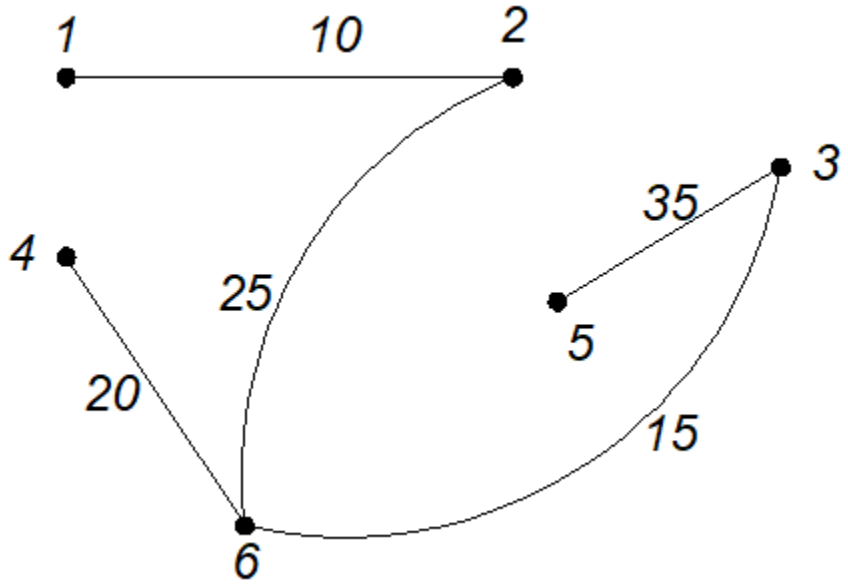
Oleh: Rinaldi M

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika ITB
2026

7. Pohon Merentang Minimum



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



(b) Pohon merentang minimum

(a) Algoritma Prim

- Misalkan sisi-sisi pohon merentang minimum yang dibangun dinyatakan di dalam himpunan *T*.

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

> "Pada setiap langkah, pilih sisi *e* = (*u*, *v*) 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.

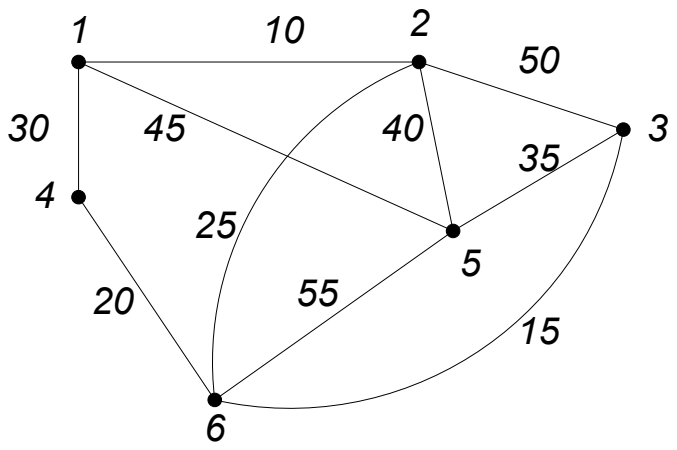
procedure *Prim*(**input** *G* : *graf*, **output** *T* : *pohon*)
{ Membentuk pohon merentang minimum T dari graf berbobot G.
Masukan: graf-berbobot terhubung G = (V, E), dengan $|V| = n$
Luaran: pohon rentang minimum T = (V, E') }

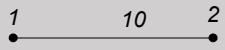
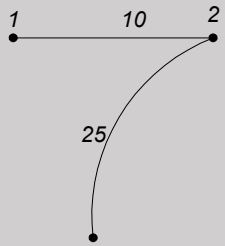
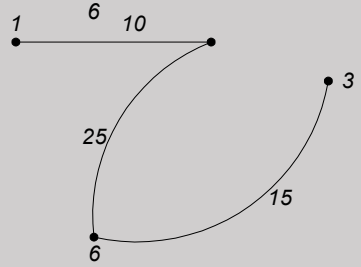
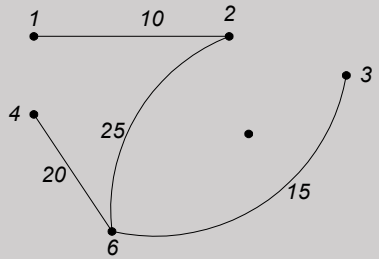
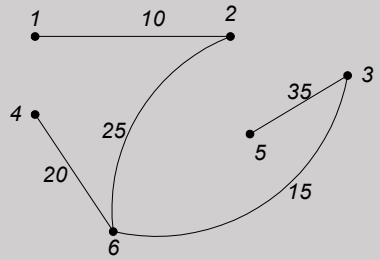
Kamus
i, *p*, *q*, *u*, *v* : **integer**

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.

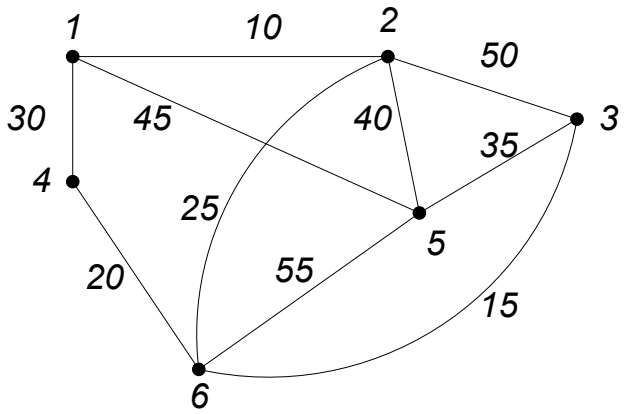
procedure *Kruskal*(**input** *G* : *graf*, **output** *T* : *pohon*)
{ Membentuk pohon merentang minimum T dari graf berbobot G.
Masukan: graf-berbobot terhubung G = (V, E), dengan $|V| = n$
Luaran: pohon rentang minimum T = (V, E') }

Kamus
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

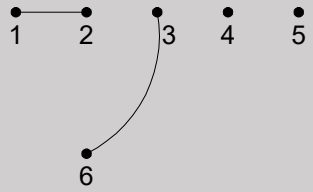
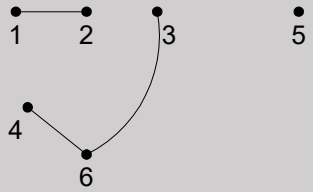
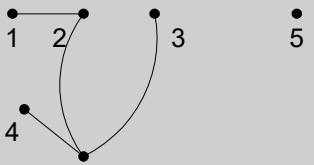
Kompleksitas algoritma: $O(|E| \log |E|)$

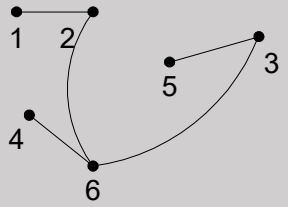
Contoh 13:



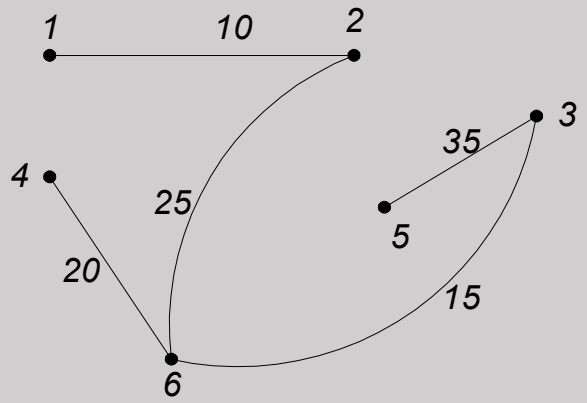
Sisi-sisi diurut menaik:

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*).

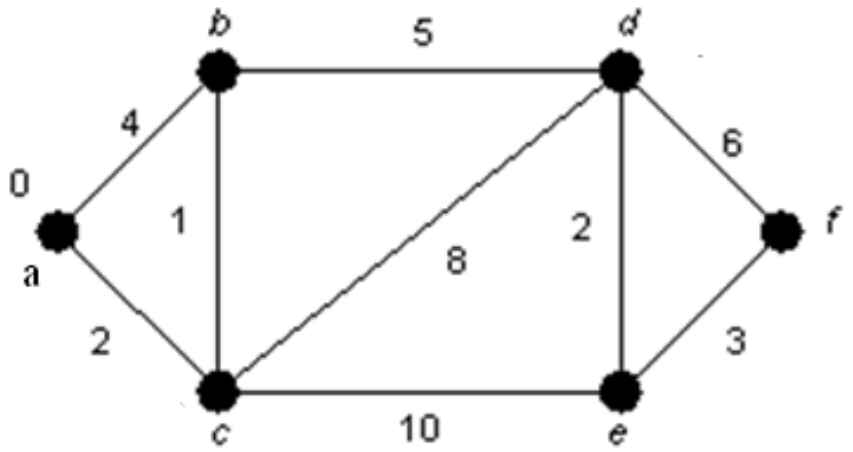
d) Lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).

→ 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?

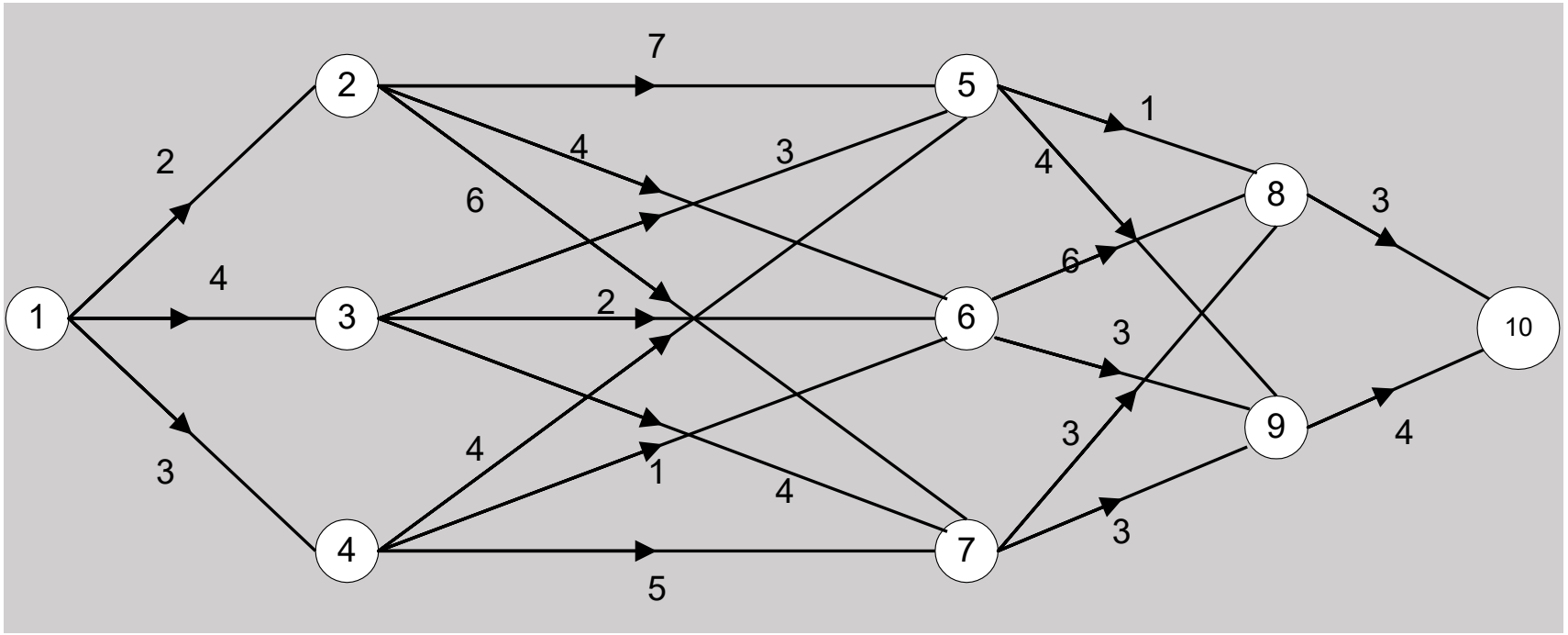
Penyelesaian dengan Algoritma *Brute Force*

- 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

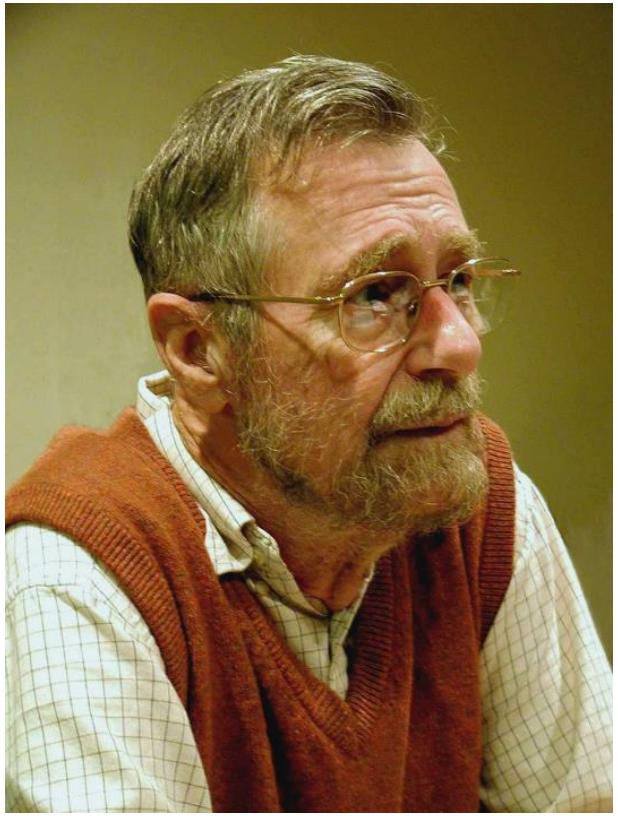
Algoritma Dijkstra

- 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)

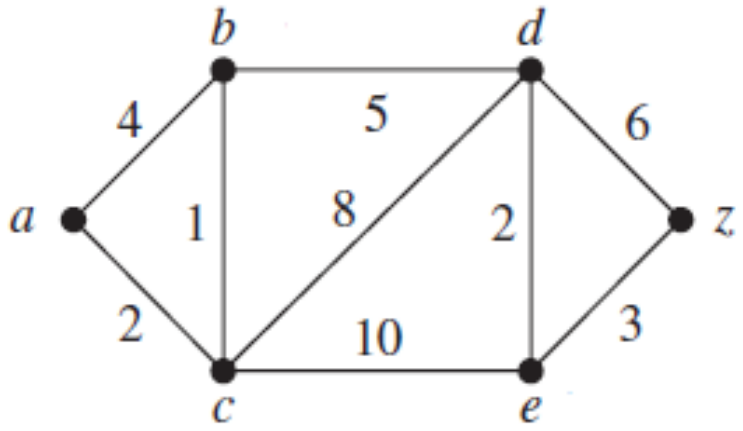
- Edsger Wybe Dijkstra was one of the most influential members of computing science's founding generation. Among the domains in which his scientific contributions are fundamental are
- algorithm design
- programming languages
- program design
- operating systems
- distributed processing

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 }

Kamus:
    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) ← ∞
    endfor
    L(a) ← 0       { jarak dari a ke a adalah 0 }
    S ← { }
    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    { jarak dari a ke u ditambah bobot sisi dari u ke v lebih kecil dari jarak a ke v }
                L(v) ← L(u) + G(u, v)          { jarak dari a ke v yang baru diganti dengan L(u) + G(u, v) }
            endif
       endfor
    enfor
```

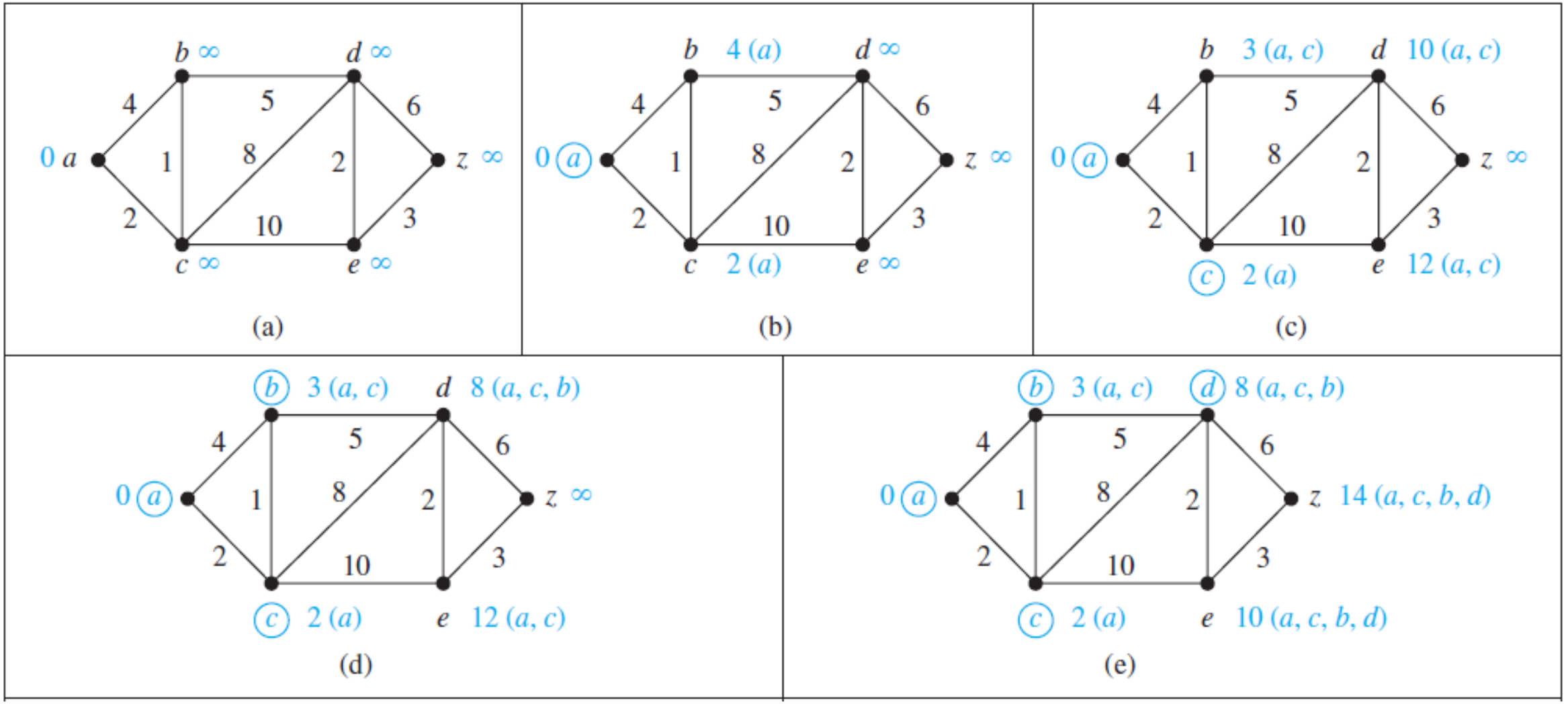
Contoh 14: Diberikan graf G di bawah ini. Carilah lintasan terpendek dari simpula a ke semua simpul lainnya.

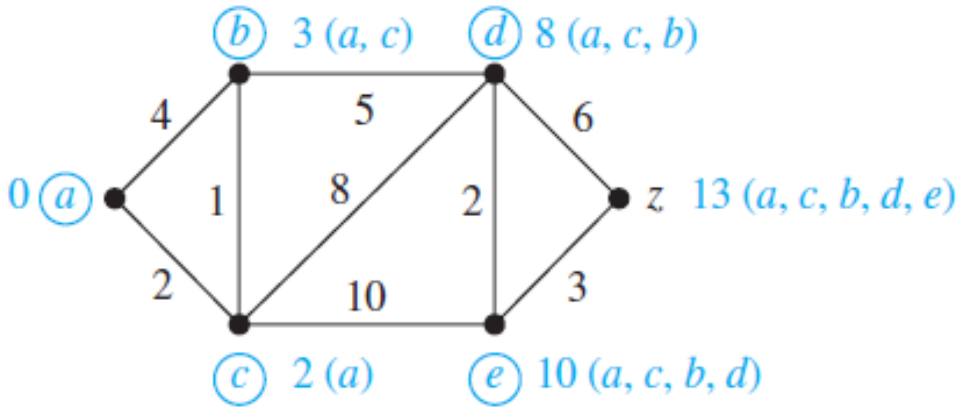


(Sumber: Rosen, *Discrete Mathematics and Its Application*, 7th Edition)

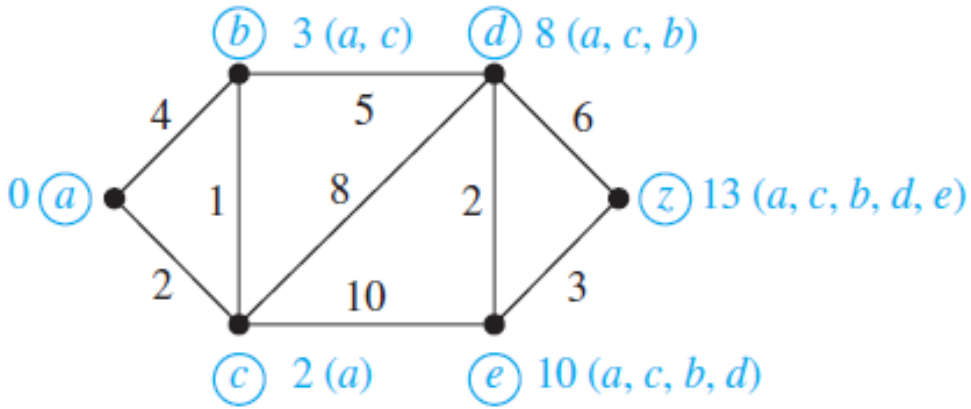
Penyelesaian:

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





(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
→ membutuhkan paling banyak *n* – 1 perbandingan: *O*(*n*)

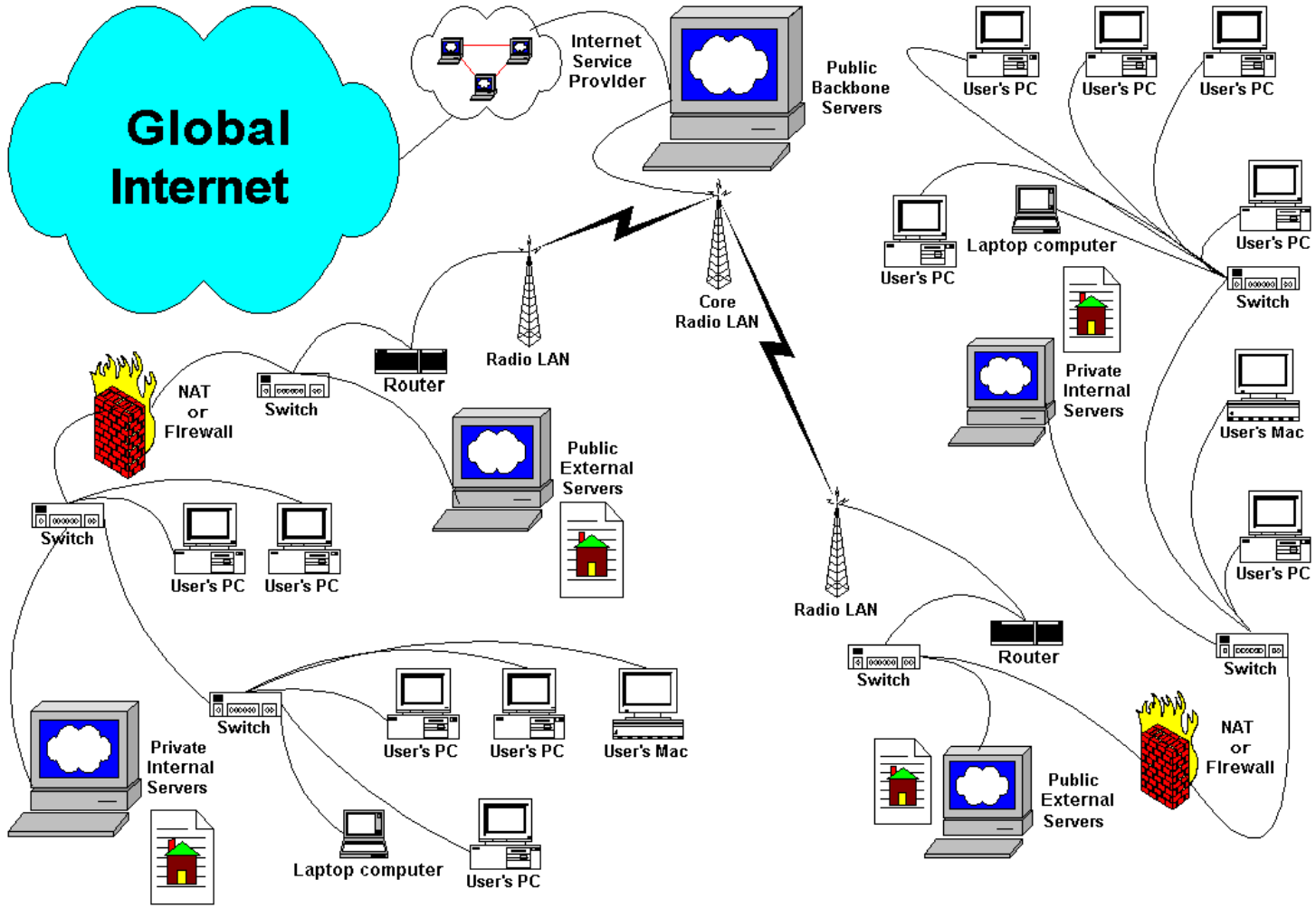
(ii) Memperbarui (*update*) jarak yang baru dari *a* ke *v*:
→ membutuhkan paling banyak *n* – 1 perbandingan dan n – 1 penjumlahan: *O*(*n*)

(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

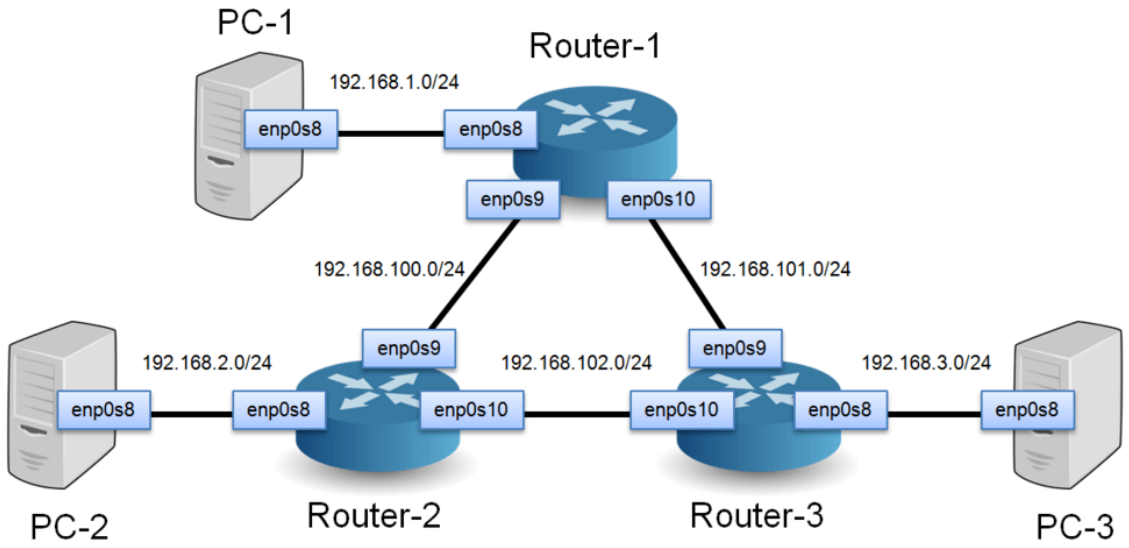


- Pesan yang dikirim dari satu komputer ke komputer lainnya umumnya dipecah menjadi sejumlah paket (*packet*) data yang berukuran lebih kecil.

- 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.

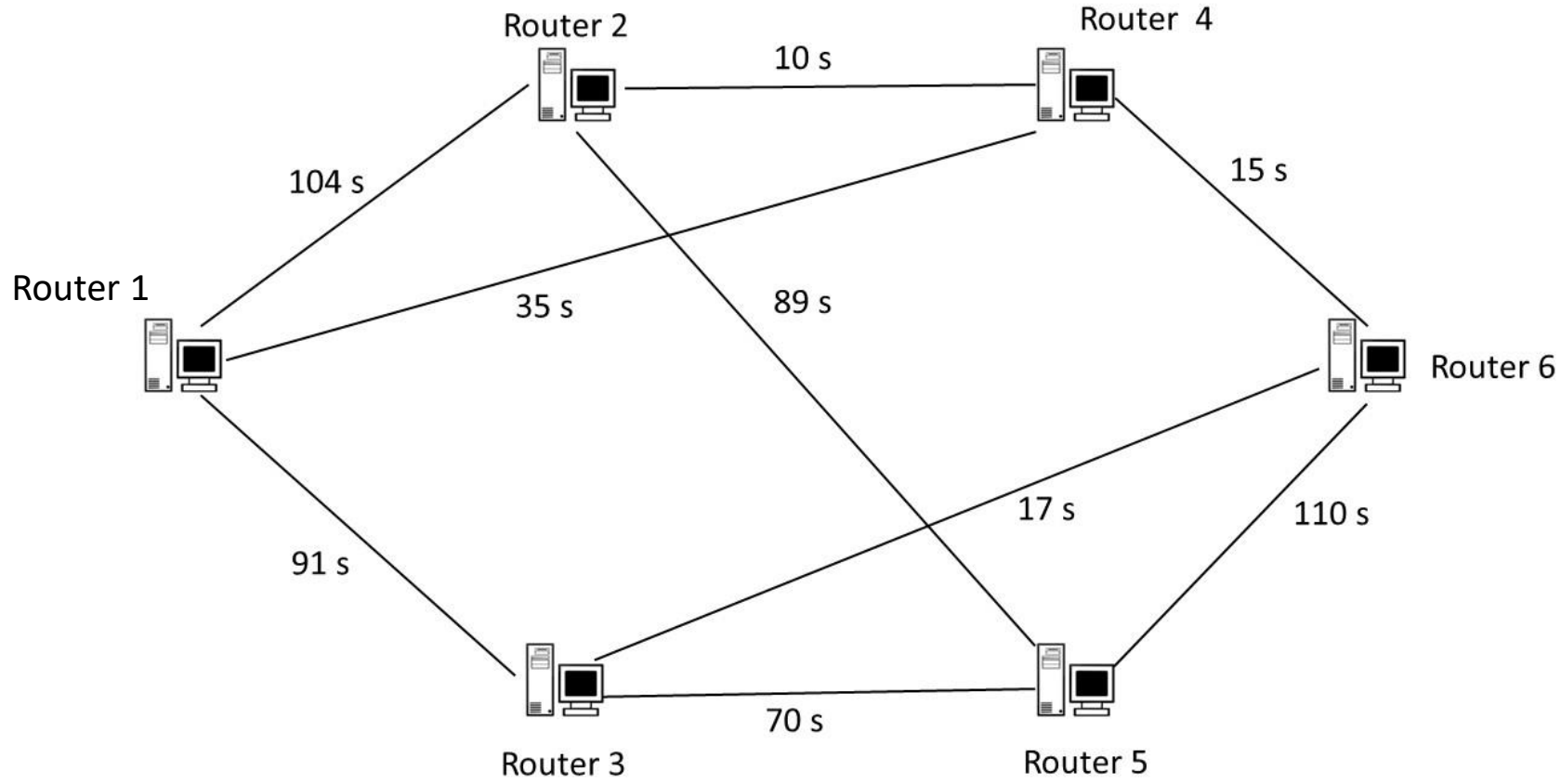
- Yang dimaksud dengan **perutean** (*routing*) adalah menentukan lintasan yang dilalui oleh paket dari komputer pengirim (asal) ke komputer penerima (tujuan).

- *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.

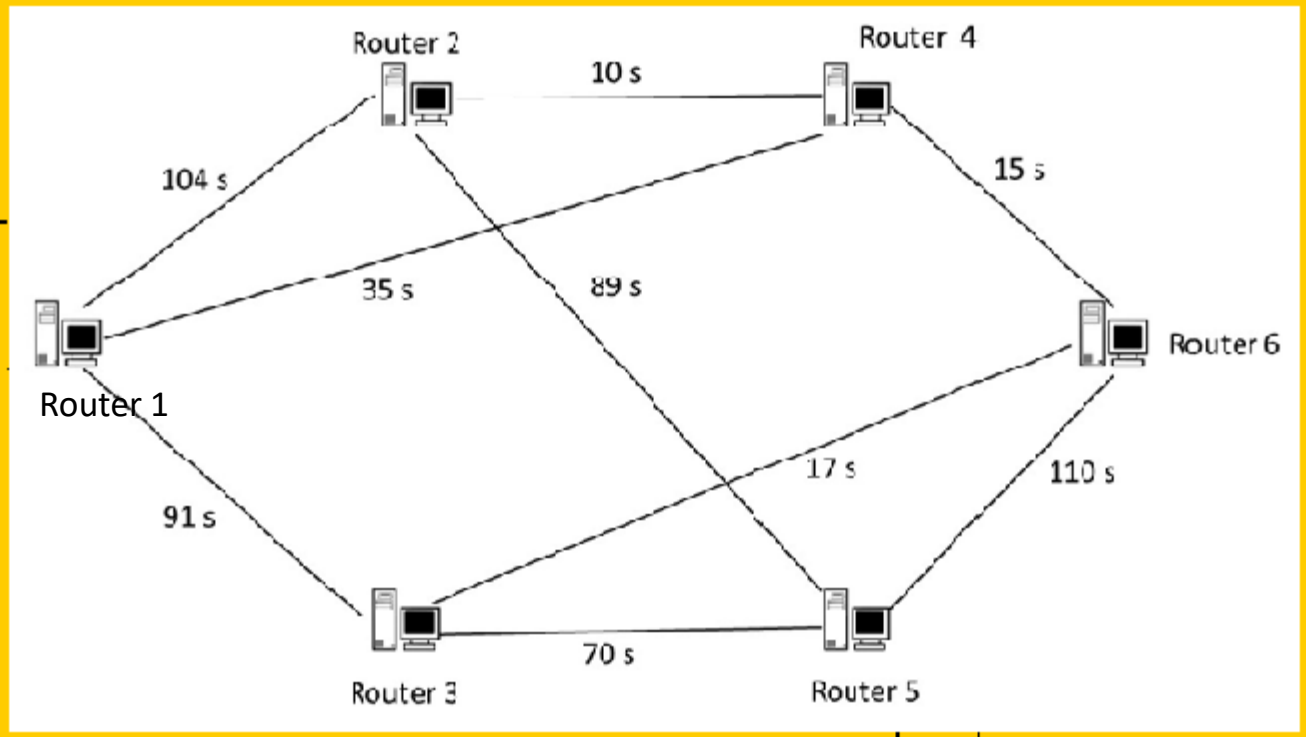
Lintasan terpendek (berdasarkan *delay time*):

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

Soal Latihan Algoritma Greedy

(Diambil dari soal-soal UTS)

IF2211 Strategi Algoritma

2025

Soal Latihan 1

Misalkan terdapat 9 buah job dengan waktu pengerjaan adalah 3, 5, 6, 10, 11, 14, 15, 18, dan 20 menit. Anda memiliki tiga buah prosesor untuk menjalankan job-job tersebut. Bagaimana cara pengaturan job pada setiap prosesor sehingga waktu penyelesaian semua job adalah minimal? Waktu penyelesaian job dihitung dari prosesor yang terlama menyelesaikan job. Misalkan cara pengaturan job adalah sbb:

Waktu penyelesaian job = 18 + 11 + 6 = 35 menit.

Jawaban:

Persoalan ini adalah *multiprocessor scheduling* untuk meminimalkan *makespan* (waktu penyelesaian maksimum).

Diketahui waktu job: 3, 5, 6, 10, 11, 14, 15, 18, dan 20 menit

Jumlah prosesor = 3

Total waktu seluruh job:

$$3 + 5 + 6 + 10 + 11 + 14 + 15 + 18 + 20 = 102 \text{ menit}$$

Batas bawah (*lower bound*) *makespan*:

$$\frac{102}{3} = 34$$

Artinya waktu minimum penyelesaian job tidak mungkin kurang dari 34 menit.

- Jika dijalankan pada prosesor dengan job waktu terpendek lebih dahulu, hasilnya sbb:

Strategi ini bukan solusi yang bagus karena waktu penyelesaiannya menjadi 6 + 14 + 20 = 40 menit .

Algoritma *greedy* yang lebih baik adalah LPT (*Longest Processing Time First*), yaitu pilih job dengan waktu terpanjang terlebih dahulu:

Algoritma *greedy*:

1. Urutkan job dari waktu terbesar ke terkecil.
2. Ambil job satu per satu.
3. Setiap job dimasukkan ke prosesor yang saat itu memiliki total waktu paling kecil.

Hasil pengurutan job: 20,18,15,14,11,10,6,5,3

Misalkan P1, P2, P3 = prosesor

Awalnya:

P1 = 0, P2 = 0, P3 = 0

Hasil pengururan job: 20,18,15,14,11,10,6,5,3

Job 20 → P1 (P1 = 20)

Job 18 → P2 (P2 = 18)

Job 15 → P3 (P3 = 15)

Job 14 → ke prosesor paling kecil (P3) (P3 = 15 + 14 = 29)

Job 11 → ke prosesor paling kecil (P2) (P2 = 18 + 11 = 29)

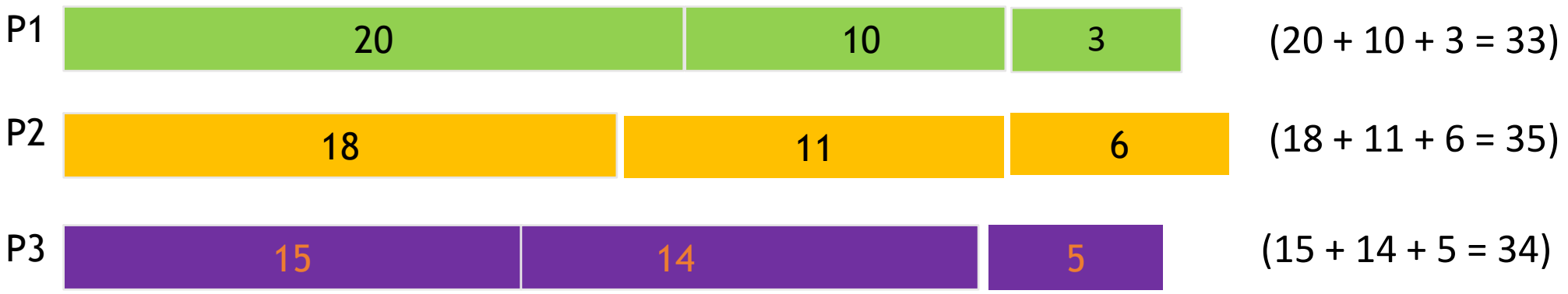
Job 10 → ke prosesor paling kecil (P1) (P1 = 20 + 10 = 30)

Job 6 → ke prosesor paling kecil (P2 atau P3, keduanya 29, ambil salah satu, misal P2):

(P2 = 29 + 6 = 35)

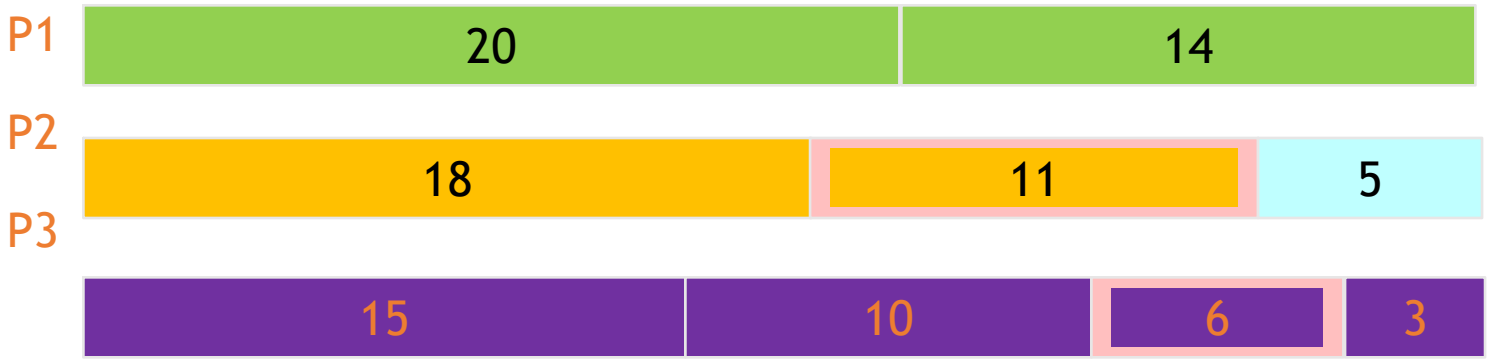
Job 5 → ke prosesor paling kecil (P3 = 29) (P3 = 29 + 5 = 34)

Job 3 → ke prosesor paling kecil (P1 = 30) (P1 = 30 + 3 = 33)



Makespan = 35 menit

- Solusi yang optimal adalah sebagai berikut:



Makespan menjadi 20 + 14 = 34 menit, optimal.

Algoritma LPT menghasilkan 35, tetapi solusi optimal adalah 34.

Artinya:

- LPT adalah heuristik greedy
- Tidak selalu optimal
- Tetapi sering mendekati optimal

Soal Latihan 2

Terdapat n buah mata kuliah yang akan dijadwalkan pada sejumlah ruang kuliah. Setiap mata kuliah i memiliki waktu mulai s_i dan waktu selesai f_i. Bagaimana menjadwalkan semua kuliah pada ruang-ruang kuliah sehingga jumlah ruang kuliah yang dipakai seminimal mungkin? Tidak boleh ada dua atau lebih kuliah yang bentrok waktunya (beririsan waktunya) menggunakan ruang kuliah yang sama. Jelaskan strategi greedy-nya seperti apa dan berapa kompleksitas waktunya. Ilustrasikan jawaban anda dengan contoh berikut: (Nilai: 15)

n = 10, (si , fi) = [(5, 6), (4, 7), (2, 5), (1, 4), (3, 7), (8, 10), (7, 8), (1, 3), (6, 9), (5, 8)]

Jawaban:
Strategi *greedy*-nya adalah:

(i) Urutkan mata kuliah dalam urutan menaik berdasarkan waktu mulainya (s).

(ii) Mulai dengan ruang kuliah ke-k = 1.

(iii)Assign kuliah-kuliah ke dalam ruang kuliah k yang waktu mulainya lebih besar atau sama dengan waktu selesai kuliah yang telah dipilih sebelumnya.

(iv)Jika ada kuliah yang tersisa, tambahkan ruang kuliah baru (k = k+ 1), lalu ulangi langkah (iii) sampai seluruh mata kuliah sudah di-*assign* ke ruang-ruang kuliah.

Contoh: n = 10, (s_i , f_i) = [(5, 6), (4, 7), (2, 5), (1, 4), (3, 7), (8, 10), (7, 8), (1, 3), (6, 9), (5, 8)]

Diurutkan: [(1, 3), (1, 4), (2, 5), (3, 7), (4, 7), (5, 6), (5, 8), (6, 9), (7, 8), (8, 10)]

Ruang kuliah k = 1: [(1, 3), (3, 7), (7, 8), (8, 10)]

Ruang kuliah k = 2: [(1, 4), (4, 7)]

Ruang kuliah k = 3: [(2, 5), (5, 6), (6, 9)]

Ruang kuliah k = 4: [(5, 8)]

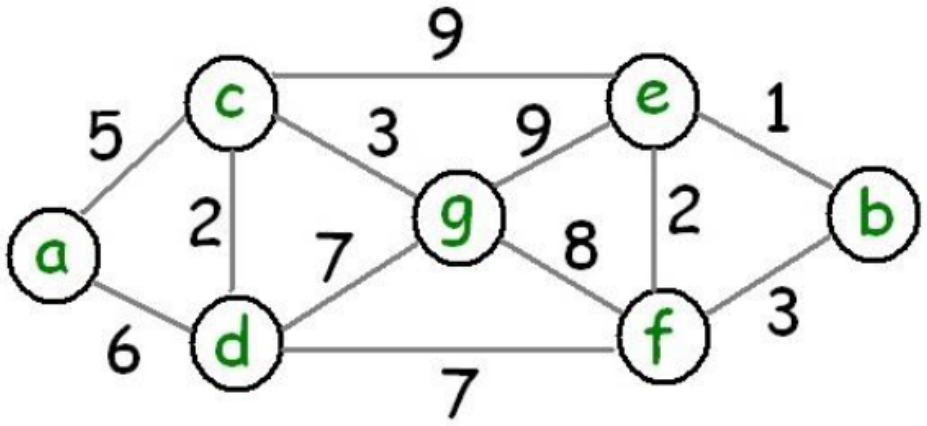
Jadi, dibutuhkan 4 ruang kuliah saja

Meng-*assign* setiap kuliah ke ruang kuliah cukup dilakukan dalam waktu O(n) saja. Kompleksitas algoritma adalah $O(n^2)$ atau O(n log n) jika waktu pengurutan diperhitungkan.

Jika waktu pengurutan tidak diperhitungkan, maka kompleksitasnya adalah O(n).

Soal Latihan 3

Diberikan graf berikut ini, kita akan menentukan lintasan terpendek dari simpul a ke semua simpul lainnya.



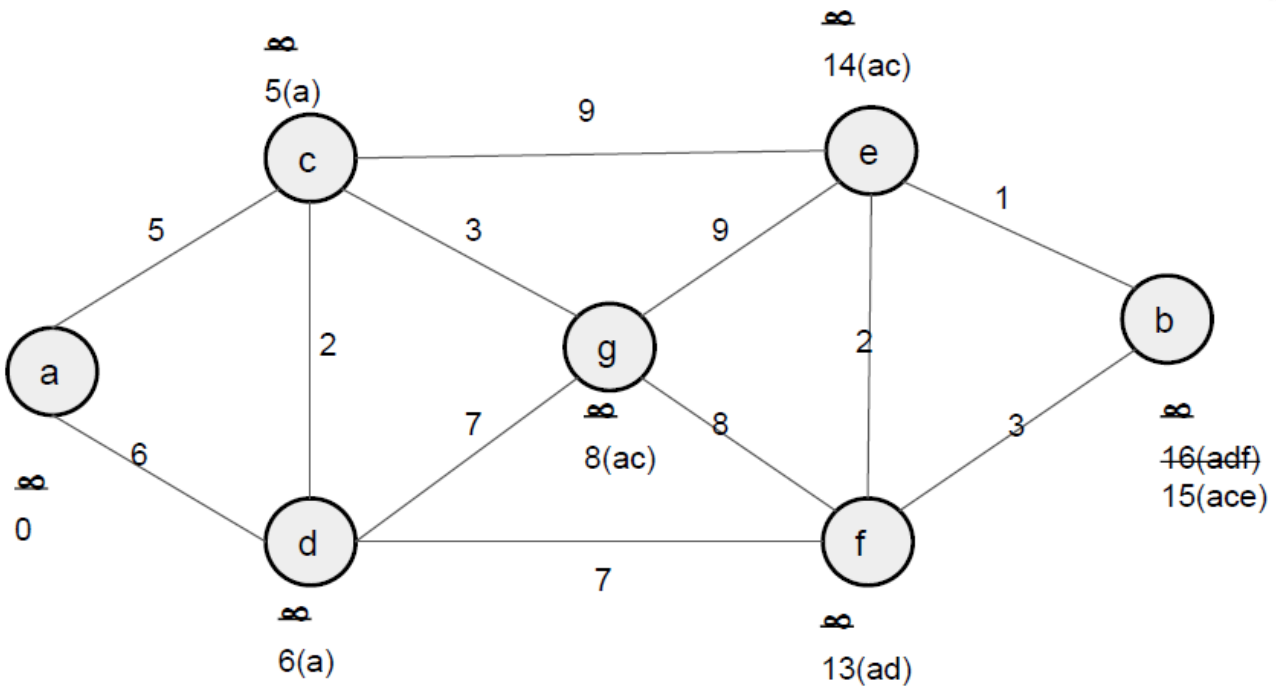
(a) Gunakanlah algoritma Kruskal untuk menentukan lintasan terpendek tersebut. Sebelum mengerjakan, tuliskanlah strategi greedy yang digunakan Kruskal. **(Nilai 7.5)**

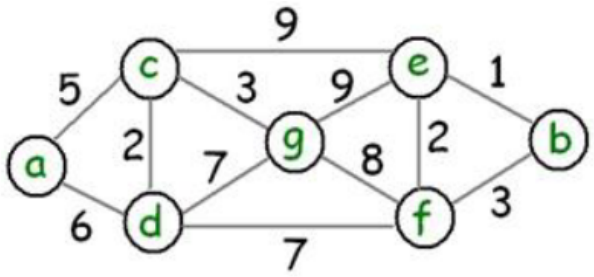
(b) Gunakanlah algoritma Dijkstra untuk menentukan lintasan terpendek tersebut. Sebelum mengerjakan, tuliskanlah strategi greedy yang digunakan Dijkstra. **(Nilai 10)**

(c) Berikanlah kesimpulan dari hasil (a) dan (b). **(Nilai 2.5)**

Jawaban:

Solusi Soal a: Dijkstra (Nilai 10)





Strategi *greedy*

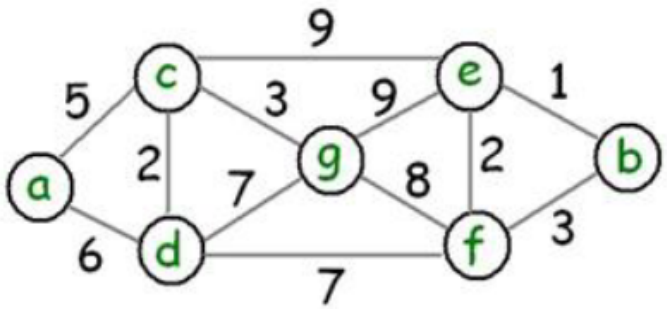
Pada setiap langkah, pilih simpul yang belum terpilih dan memiliki panjang lintasan terpendek dari simpul awal, lalu update simpul lain yang belum terpilih dengan L(u) + G(u,v) jika L(u)>L(u) + G(u,v).

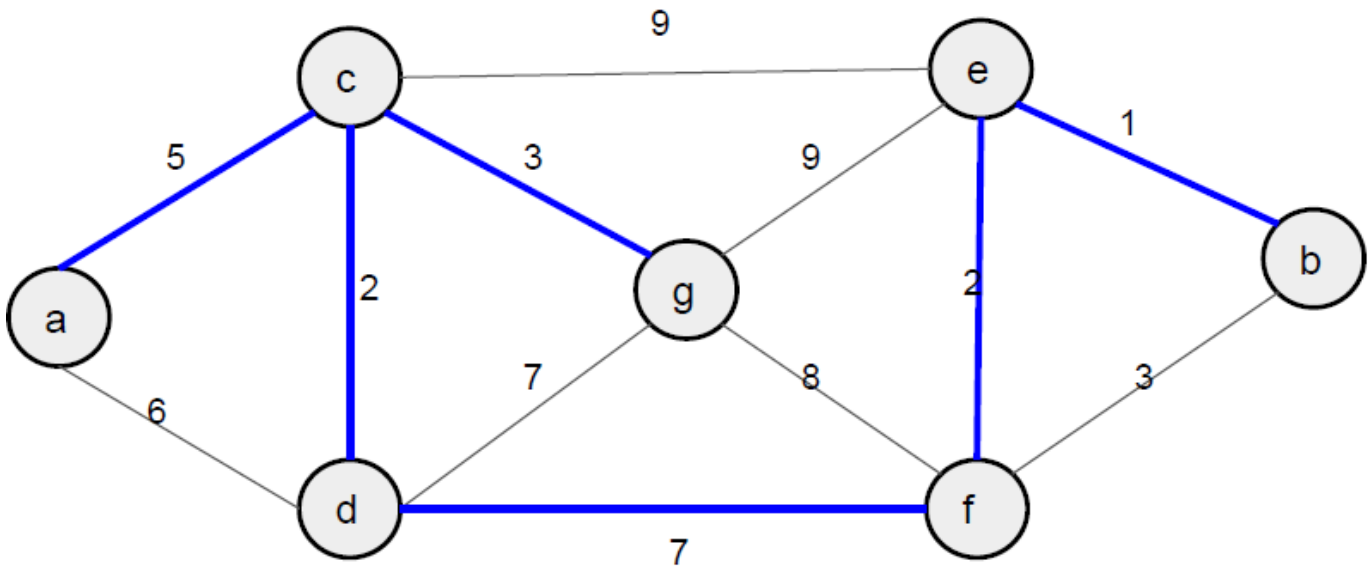
(versi slide kuliah): Pada setiap langkah, ambil sisi yang berbobot minimum yang menghubungkan sebuah simpul yang sudah terpilih dengan sebuah simpul lain yang belum terpilih.

Lintasan terpendek Dijkstra:

a-b: a-c-e-b = 15; a-c: a-c = 5; a-d: a-d = 6
a-e: a-c-e = 14; a-f: a-d-f = 13; a-g: a-c-g = 8

Solusi b: Kruskal (nilai 7.5)





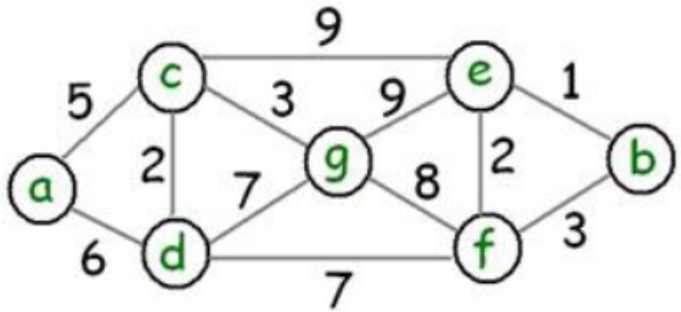
Strategi *greedy*:

Pada setiap langkah, pilih **sisi** *e* dari graf *G* yang mempunyai bobot minimum tetapi *e* tidak membentuk sirkuit di *T*.

Lintasan terpendek:

a-b: a-c-d-f-e-b = 17
a-c: a-c = 5
a-d: a-c-d = 7
A-e: a-c-d-f-e = 16
A-f: a-c-d-f = 14
A-g: a-c-g = 8

Solusi c (nilai 2.5)



Lintasan terpendek Dijkstra:
a-b: a-c-e-b = 15; a-c: a-c = 5; a-d: a-d = 6
a-e: a-c-e = 14; a-f: a-d-f = 13; a-g: a-c-g = 8

Lintasan terpendek Kruskal:
a-b: a-c-d-f-e-b = 17; a-c: a-c = 5; a-d: a-c-d = 7
a-e: a-c-d-f-e = 16; a-f: a-c-d-f = 14; a-g: a-c-g = 8

Kesimpulan:
Dijkstra menjamin memberikan lintasan terpendek ke semua simpul; sedangkan Kruskal hanya memberikan pohon merentang minimum yang tidak memberikan lintasan terpendek ke semua simpul.

Bersambung ke bagian 3