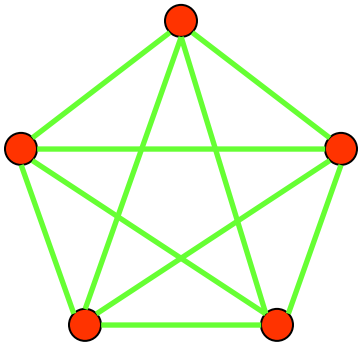
Lanjutan graf

Operasi pada Graf

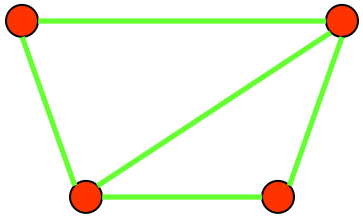
Definisi: Suatu **subgraf** dari graf G = (V, E) adalah graf H = (W, F) dimana $W \subseteq V$ dan $F \subseteq E$.

Catatan: Tentu saja H adalah graf yang valid, sehingga kita tidak bisa membuang titik ujung dari garis hubung yang tersisa saat kita membentuk H.

Contoh:



K_5

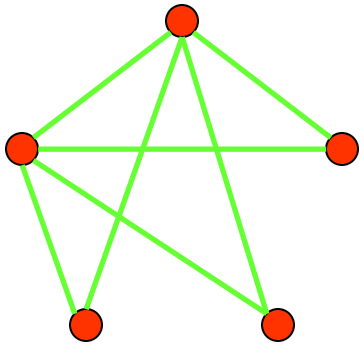


subgraf dari K_5

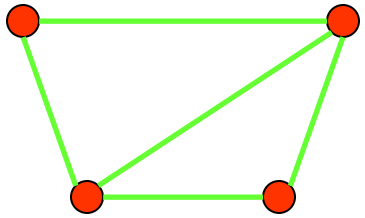
Operasi pada Graf

Definisi: Gabungan dari dua graf sederhana $G_1 = (V_1, E_1)$ dan $G_2 = (V_2, E_2)$ adalah graf sederhana dengan himpunan simpul $V_1 \cup V_2$ dan himpunan garis hubung $E_1 \cup E_2$.

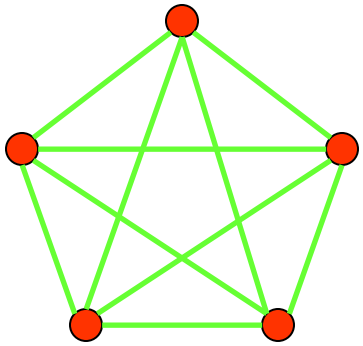
Gabungan dari G_1 dan G_2 dituliskan sebagai $\mathbf{G_1 \cup G_2}$.



G_1



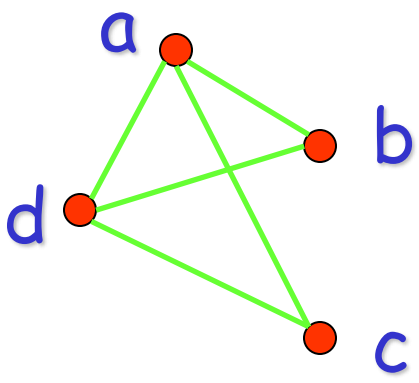
G_2

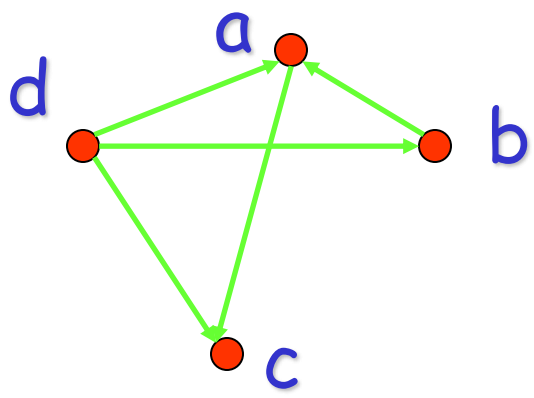


$G_1 \cup G_2 = K_5$

Adjacency Matrix

Representasi Graf





Simpul	Simpul tetangga
a	b, c, d
b	a, d
c	a, d
d	a, b, c

Simpul awal	Simpul akhir
a	c
b	a
c	
d	a, b, c

Representasi Graf

Definisi: Misalkan G = (V, E) sebuah graf sederhana dengan |V| = n. Anggap simpul pada G disusun dengan urutan seperti $v_1, v_2, \ldots, v_n$.

Matriks ketetanggaan (Adjacency matrix) A (atau A_G) dari graf sederhana G, yang berkaitan dengan simpul-simpul , adalah sebuah n×n matriks boolean (zero-one) dengan elemen ke (i, j) adalah 1 jika v_i dan v_j bertetangga, dan selainnya 0.

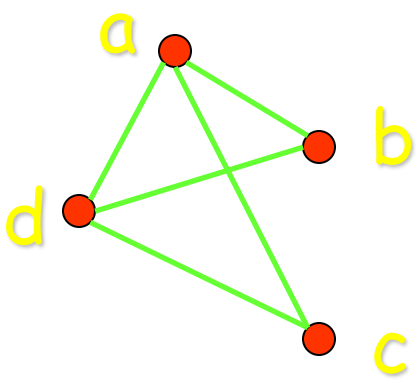
Dengan kata lain, untuk sebuah adjacency matrix A = $[a_{ij}]$,

$a_{ij} = 1$ jika $\{v_i, v_j\}$ adalah sebuah garis hubung (edge) dari G,

$a_{ij} = 0$ selainnya.

Representasi Graf : Adjacency matrix

Contoh: Tentukan adjacency matrix A_G untuk graf berikut berdasarkan urutan simpul a, b, c, d !



Solusi:

$$A_G = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Catatan : adjacency matrix dari graf tidak berarah selalu simetris..

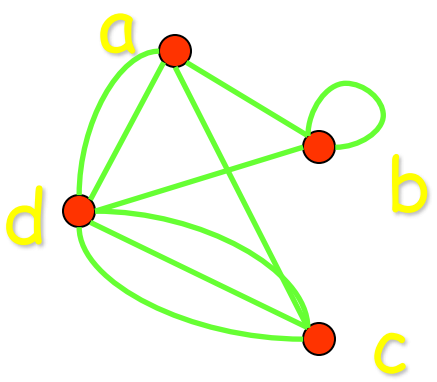
Representasi Graf : Adjacency matrix

Untuk representasi graf dengan garis hubung ganda / berlapis banyak (multiple edge), matriks boolean tidak bisa dipakai, sebagai gantinya dipergunakan matriks bilangan cacah

Elemen ke (i, j) dari matriks tersebut sama dengan jumlah garis hubung yang terdapat pada kedua simpul $\{v_i, v_j\}$.

Representasi Graf : Adjacency matrix

Contoh: Tentukan adjacency matrix A_G untuk graf berikut yang berdasarkan urutan simpul a, b, c, d !



Solusi:

$$A_G = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 3 \\ 2 & 1 & 3 & 0 \end{bmatrix}$$

Catatan : adjacency matrix dari graf tidak berarah selalu simetris

Incidence Matrix

Representasi Graf : Incidence matrix

Definisi: Misalkan G = (V, E) sebuah graf tak berarah dengan |V| = n. Anggap simpul dan garis hubung pada G disusun dengan urutan seperti $v_1, v_2, \ldots, v_n$ dan $e_1, e_2, \ldots, e_m$.

Incidence matrix dari G yang berkaitan dengan simpul dan garis hubung adalah matriks boolean $n \times m$ dengan elemen ke (i, j) =1 jika garis e_j terhubung dng simpul v_i, dan selain itu = 0.

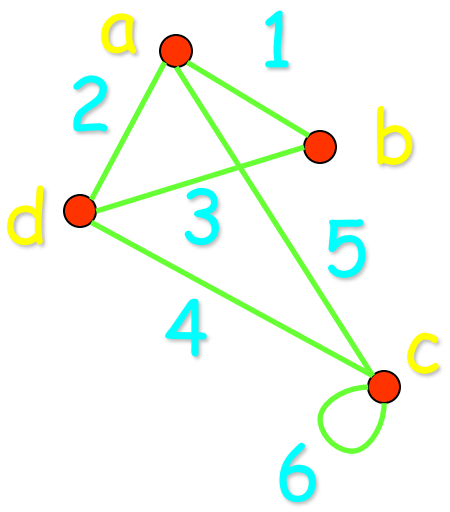
Dengan kata lain , untuk sebuah *incidence matrix* M = $[m_{ij}]$,

$m_{ij} = 1$ jika garis e_j terhubung dengan v_i

$m_{ij} = 0$ selainnya.

Representasi Graf : Incidence matrix

Contoh: Tentukan *incidence matrix* M untuk graf berikut berdasarkan urutan simpul a, b, c, d dan garis 1, 2, 3, 4, 5, 6 !



Solusi :

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Catatan : *incidence matrix* dari graf tidak berarah perkolomnya berisi 2 buah nilai 1 jika garis menghubungkan dua buah simpul dan berisi 1 buah untuk loop.

Graf yang Isomorfis

Definisi: graf sederhana $G_1 = (V_1, E_1)$ dan $G_2 = (V_2, E_2)$ disebut **isomorfis jika** ada sebuah fungsi bijektif (fungsi satu-ke-satu dan onto) dari V_1 ke V_2 dengan sifat bahwa a bertetangga dengan b pada G_1 jika dan hanya jika f(a) bertetangga dengan f(b) pada G_2, untuk seluruh a dan b di V_1.

Fungsi f seperti itu disebut **isomorfisme**.

Dengan kata lain , G_1 dan G_2 adalah isomorfis jika simpul-simpulnya dapat diurutkan dengan suatu cara sedemikian rupa sehingga adjacency matrix M_{G_1} dan M_{G_2} adalah identik.

Graf yang Isomorfis

Dari sudut pandang visual, G_1 dan G_2 adalah isomorfis jika mereka dapat disusun dengan cara sedemikian rupa sehingga tampilannya identik (tentu tanpa merubah ketetanggaan).

Menentukan dua buah graf **tidak isomorfis** lebih mudah dibandingkan dengan menentukan apakah dua buah graf isomorfis.

Untuk dua buah graf sederhana dengan masing-masing simpulnya n buah, maka ada **n! kemungkinan isomorfisme** yang harus diperiksa guna menunjukkan apakah kedua graf isomorfis.

Graf yang Isomorfis

Untuk tujuan ini kita dapat memeriksa **invarian**, yaitu, sifat kepemilikan yang harus dimiliki oleh dua buah graf sederhana yang isomorfis. Mereka harus

- memiliki jumlah simpul yang sama, dan
- jumlah garis yang sama , dan
- derajat yang sama dari simpul-simpulnya.

Perhatikan bahwa dua graf yang salah satu dari invarian di atas berbeda pasti menyebabkan kedua graf tersebut tidak isomorfis, tetapi jika seluruhnya sesuai, belum tentu graf tsb isomorfis.

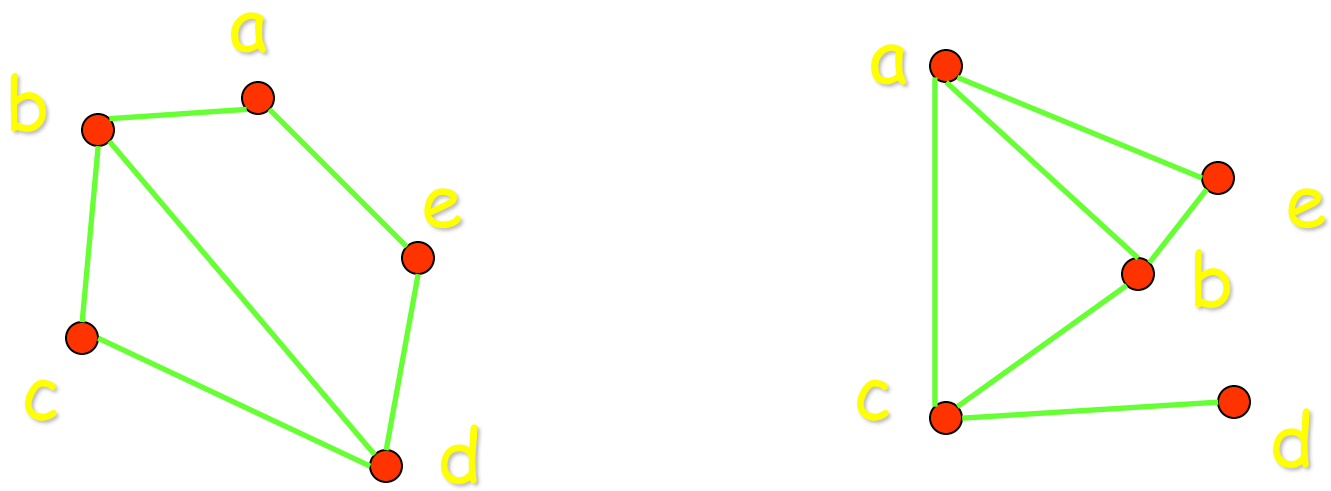
Graf yang Isomorfis

Contoh I: Apakah kedua graf berikut isomorfis ?

Solusi : Ya, keduanya isomorfis karena dapat disusun sehingga terlihat identik. Dapat diamati, jika pada graf sebelah kanan kita gerakkan simpul b kesebelah kiri garis {a, c}. Maka fungsi isomorfis f dari graf kiri ke kanan adalah : f(a) = e, f(b) = a, f(c) = b, f(d) = c, f(e) = d.

Graf yang Isomorfis

Contoh II: Bagaimana dng kedua buah graf berikut ?



Solusi: tidak, mereka tidak isomorfis, karena derajat dari simpul-simpulnya berlainan. Simpul d pada graf dikanan berderajat satu, tetapi tidak ada satupun simpul pada graf dikiri yang berderajat 1.

konektivitas

Definisi: Sebuah **lintasan** (path) dengan panjang n dari (*node*) u ke v, dimana n adalah bilangan bulat positif, dalam sebuah graf **tidak berarah** adalah sebuah urutan garis hubung $e_1, e_2, \ldots, e_n$ dari graf sedemikian hingga $f(e_1) = \{x_0, x_1\}$, $f(e_2) = \{x_1, x_2\}$, $f(e_n) = \{x_{n-1}, x_n\}$, dgn $x_0 = u$ dan $x_n = v$.

Jika graf tsb berupa graf sederhana, kita menotasikan lintasan tersebut dengan **urutan/deretan simpul** $x_0, x_1, \ldots, x_n$, karena secara unik dapat menentukan lintasan.

Lintasan adalah sebuah **sirkit** jika dimulai dan diakhiri pada simpul yang sama, yaitu jika u = v.

Definisi (contd.): Lintasan atau sirkit disebut ***pass through*** atau ***traverse*** $x_1, x_2, \ldots, x_{n-1}$.

Lintasan atau sirkit disebut **sederhana** jika tidak mengandung garis yang sama lebih dari sekali.

konektivitas

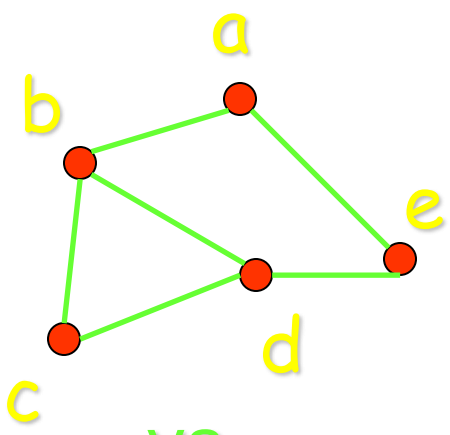
Definisi: graf tak berarah disebut **terhubung (conected)** jika ada **lintasan** diantara setiap pasangan dari simpul yang berbeda dalam graf

Sebagai contoh, setiap dua komputer dalam jaringan dapat berkomunikasi jika dan hanya jika graf dari jaringan tersebut terhubung

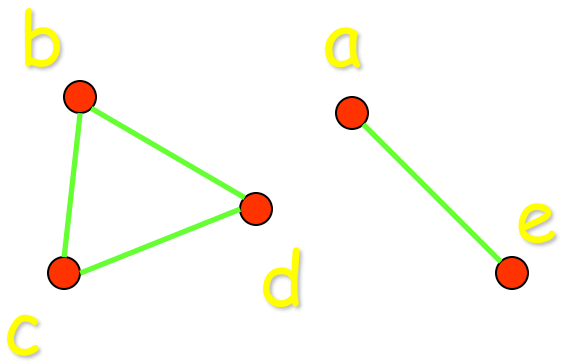
Catatan : sebuah graf yang mengandung hanya satu simpul selalu terhubung, karena ia tidak berisi suatu pasangan simpul yang berbeda

konektivitas

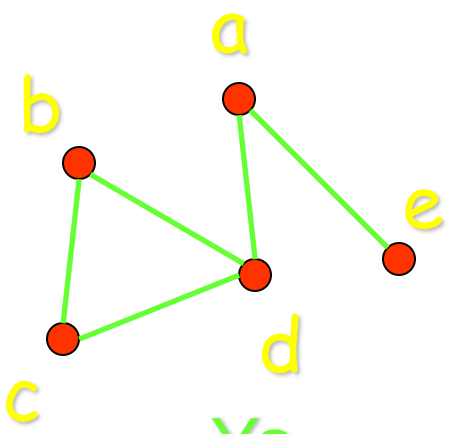
contoh: manakah dari graf berikut yang terhubung?



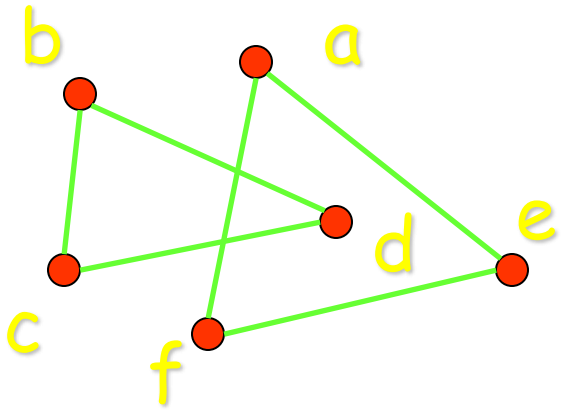
ya.



tidak.



Ya.



tidak.

konektivitas

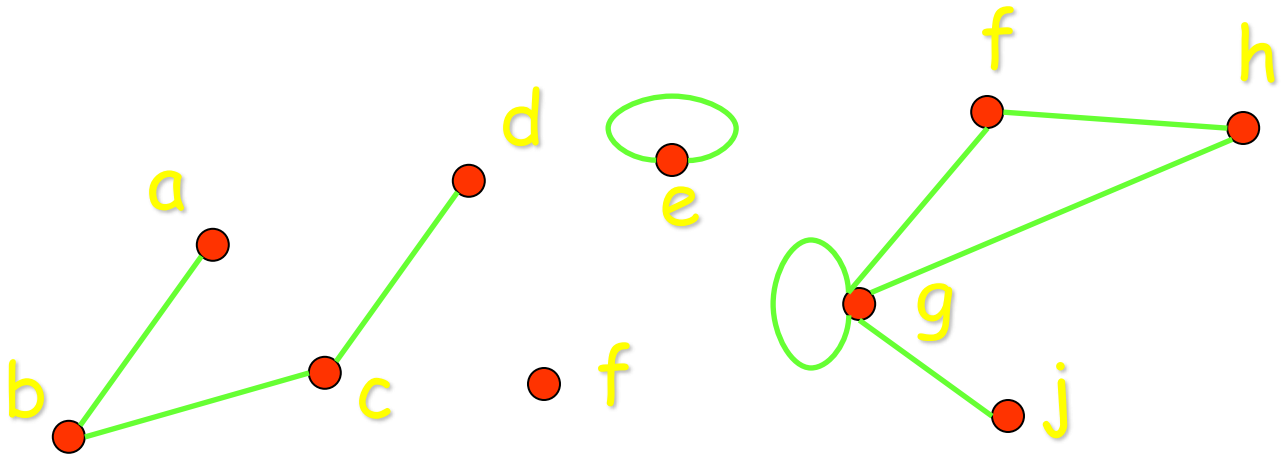
Teorema: Selalu ada lintasan **sederhana** antara setiap pasangan simpul yang berbeda dari suatu graf-tak-berarah yang terhubung

Definisi: sebuah graf yang tidak terhubung adalah gabungan dari dua atau lebih sub-graf yang terhubung dengan masing-masing pasangan darinya tidak memiliki simpul bersama (disjoint).

Sub-graf- Sub-graf yang terhubung tetapi disjoint tersebut dinamakan **komponen-komponen terhubung** dari graf.

konektivitas

Contoh: tentukan komponen-komponen terhubung dari graf berikut ini ?



Solusi: komponen-komponen terhubung adalah graf dengan simpul-simpul {a, b, c, d}, {e}, {f}, {f, g, h, j}.

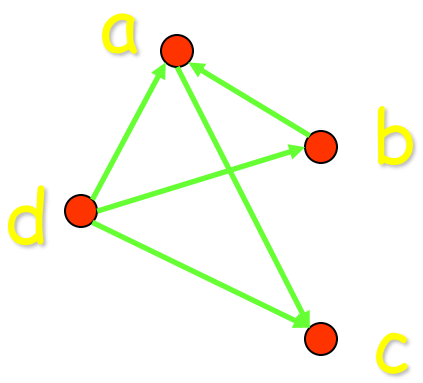
konektivitas

Definisi: sebuah graf berarah disebut **terhubung kuat (strongly connected)** jika ada sebuah lintasan dari a ke b **dan** dari b ke a untuk semua pasangan simpul a, b pada graf.

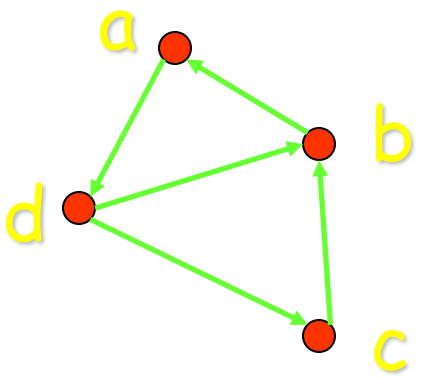
Definisi: sebuah graf berarah disebut **terhubung lemah** jika **hanya ada sebuah lintasan** diantara dua simpul dalam graf tidak berarahnya. (Ada lintasan dari a ke b, tetapi tidak dari b ke a, atau sebaliknya)

konektivitas

Contoh: Apakah graf berarah berikut ini terhubung kuat atau lemah?



Terhubung lemah, karena , sebagai misal, tidak ada lintasan dari b ke d.

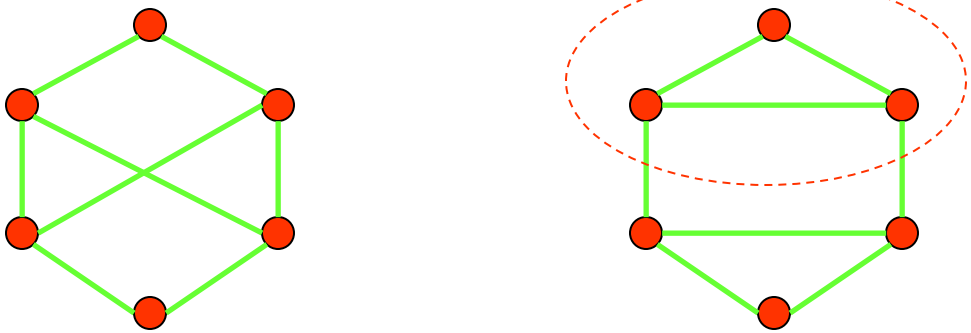


Terhubung kuat, karena ada lintasan diantara seluruh kemungkinan pasangan simpul.

konektivitas

Ide: Jumlah dan ukuran dari komponen dan sirkit terhubung merupakan invarian berkaitan dengan isomorfisme dari graf sederhana

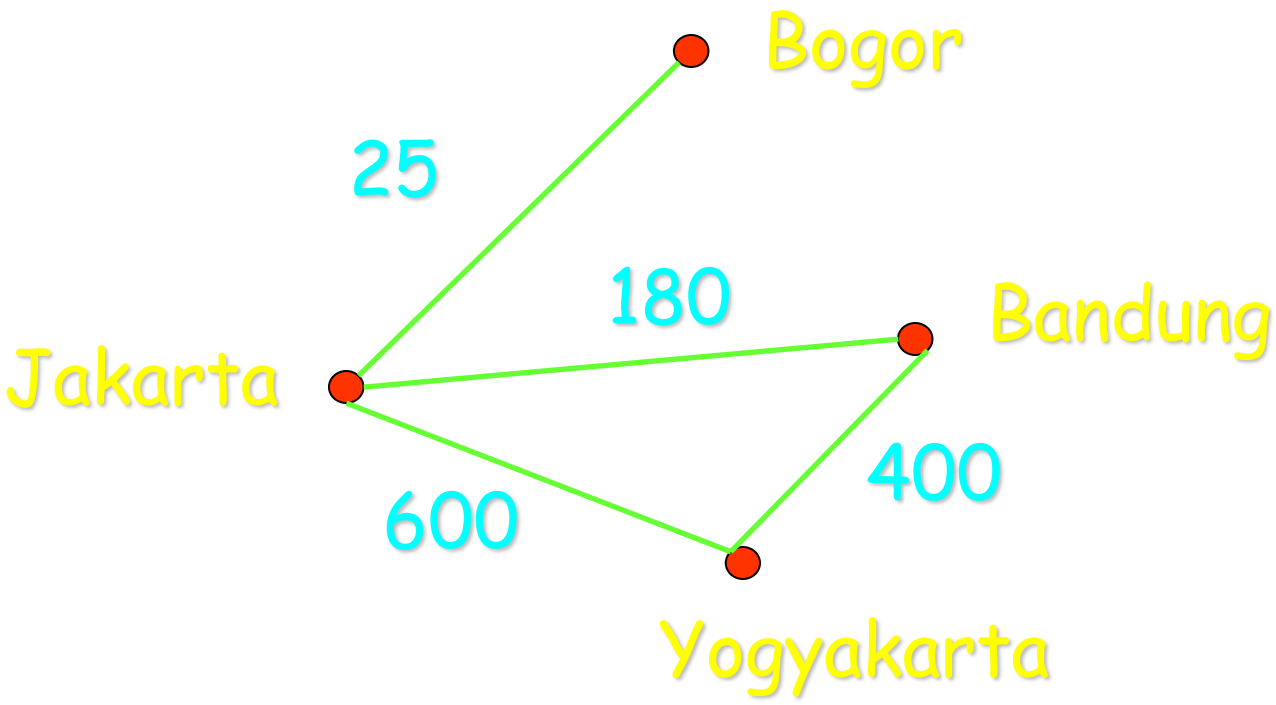
Contoh: apakah dua graf berikut ini isomorfis ?



Solusi: tidak , karena graf sebelah kanan berisi sirkit dengan panjang 3 , sedangkan pada graf sebelah kiri tidak ada.

Masalah-masalah lintasan terpendek

Kita dapat memberikan bobot pada garis dari graf, sebagai contoh untuk merepresentasikan jarak antara kota dalam jaringan jalan kereta :



Masalah-masalah lintasan terpendek

Pembobotan graf tersebut dapat juga dipakai untuk memodelkan jaringan komputer dengan *response time* atau biaya sebagai bobotnya.

Satu pertanyaan yang sangat penting yang dapat kita selidiki pada graf seperti ini adalah :

Yang manakah **lintasan terpendek** antara dua simpul dalam graf, yaitu, lintasan dengan **jumlah bobot minimal** sepanjang jalannya ?

Hal ini misalnya dapat berkaitan dengan koneksi tercepat pada jairngan komputer atau hubungan lintasan kereta terpendek

Algoritma Dijkstra

Algoritma Dijkstra adalah sebuah prosedur iteratif untuk mencari lintasan terpendek antara dua simpul, a dan z, di dalam graf dengan pembobot.

Prosedur ini dilaksanakan dengan cara mencari panjang lintasan terpendek dari sebuah simpul pendahulu dan menambahkan simpul-simpul tersebut ke himpunan simpul S. Algoritma berhenti setelah mencapai simpul z.

Algoritma Dijkstra

procedure Dijkstra(G: graf sederhana yang terhubung dan dengan pembobot, dengan simpul $a = v_0, v_1, \ldots, v_n = z$ dan bobot positif $w(v_i, v_j)$, dimana $w(v_i, v_j) = \infty$ jika $\{v_i, v_j\}$ bukan garis hubung pada G)

for i := 1 **to** n

    $L(v_i) := \infty$

L(a) := 0

S := ∅

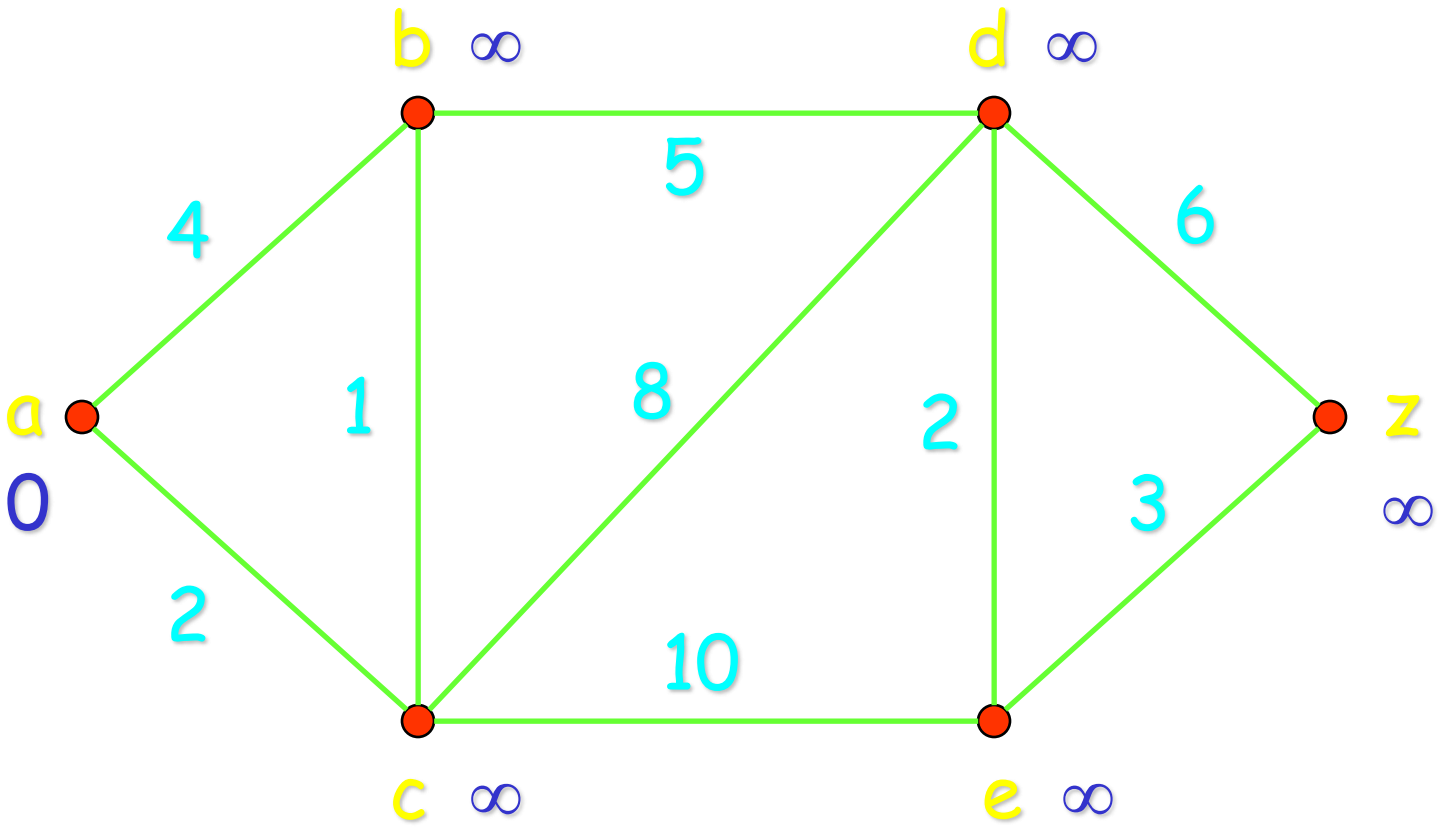
{label-label sekarang diinisialisasi sehingga label dari a adalah nol dan label yang lainnya adalah ∞, dan set S adalah kosong}

Algoritma Dijkstra

while $z \notin S$
begin
    u := simpul tidak pada S dengan minimal L(u)
    S := $S \cup \{u\}$
    for seluruh simpul v tidak pada S
        if L(u) + w(u, v) < L(v) **then** L(v) := L(u) + w(u, v)
    {ini menjumlahkan sebuah simpul ke S dengan label minimal dan memperbaharui label-label simpul yang tidak di S}
end {L(z) = panjang jalur terpendek dari a ke z}

Algoritma Dijkstra

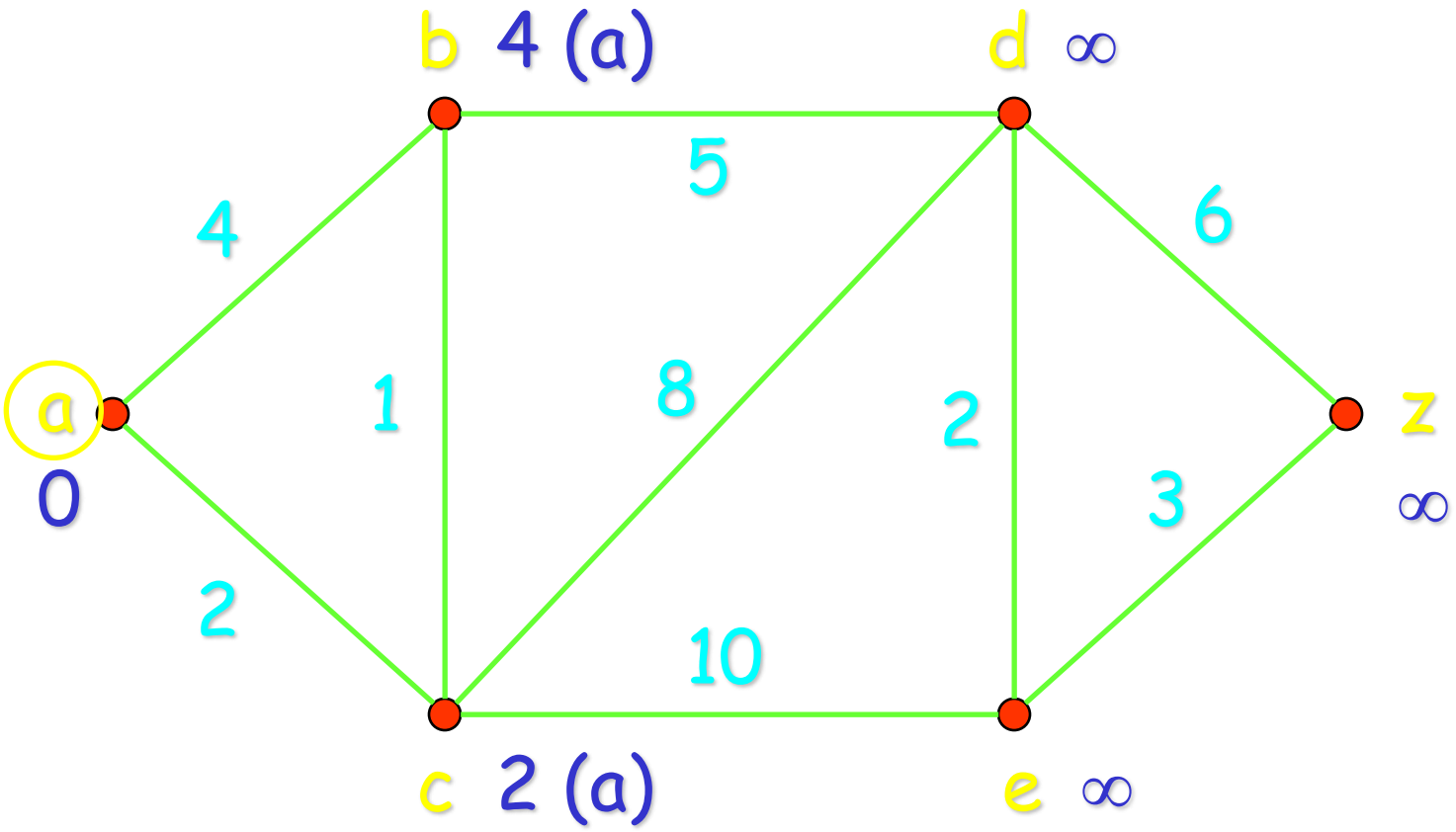
contoh: cari lintasan terpendek dari a ke z



Step 0

Algoritma Dijkstra

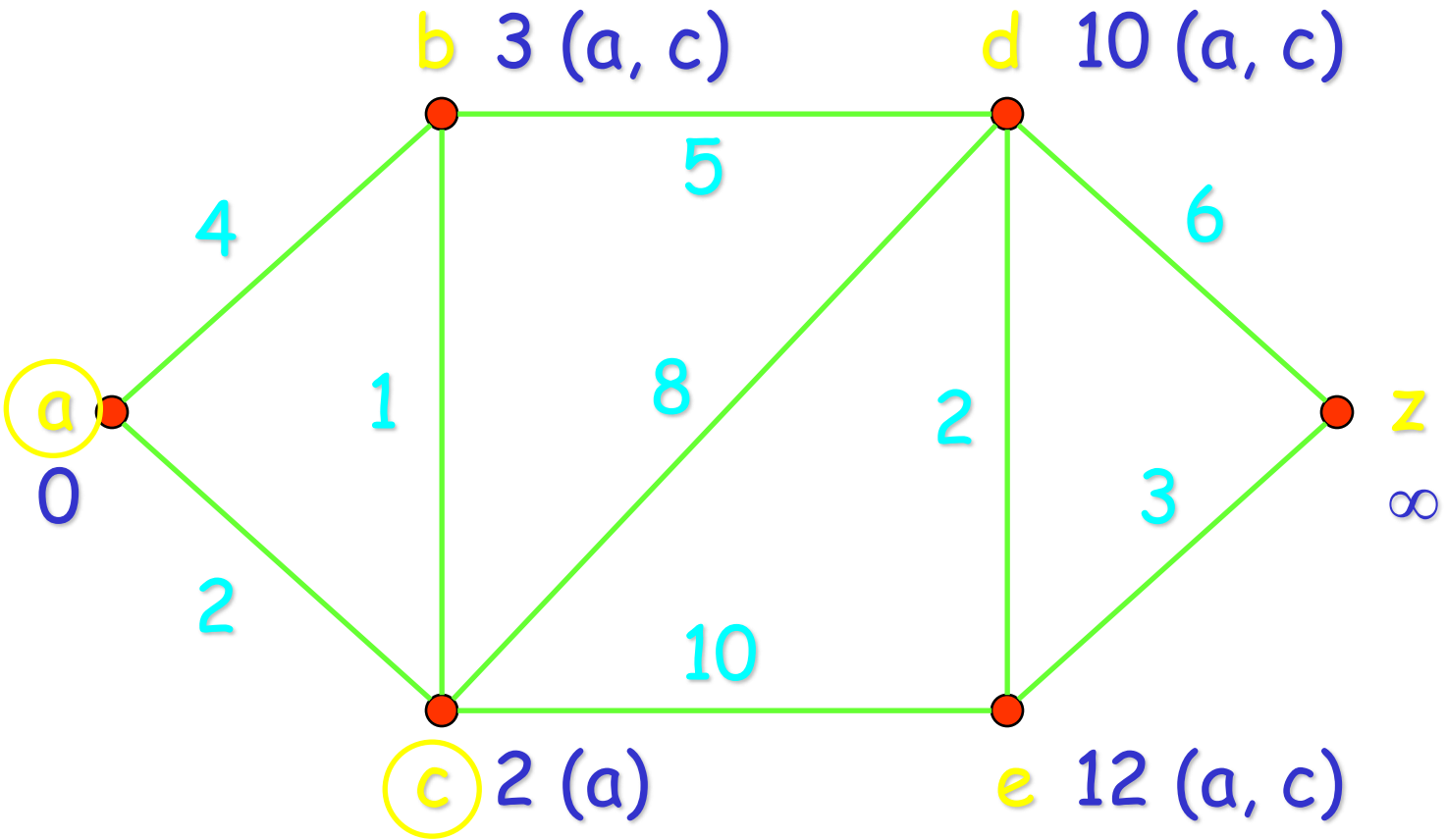
contoh:



Step 1

Algoritma Dijkstra

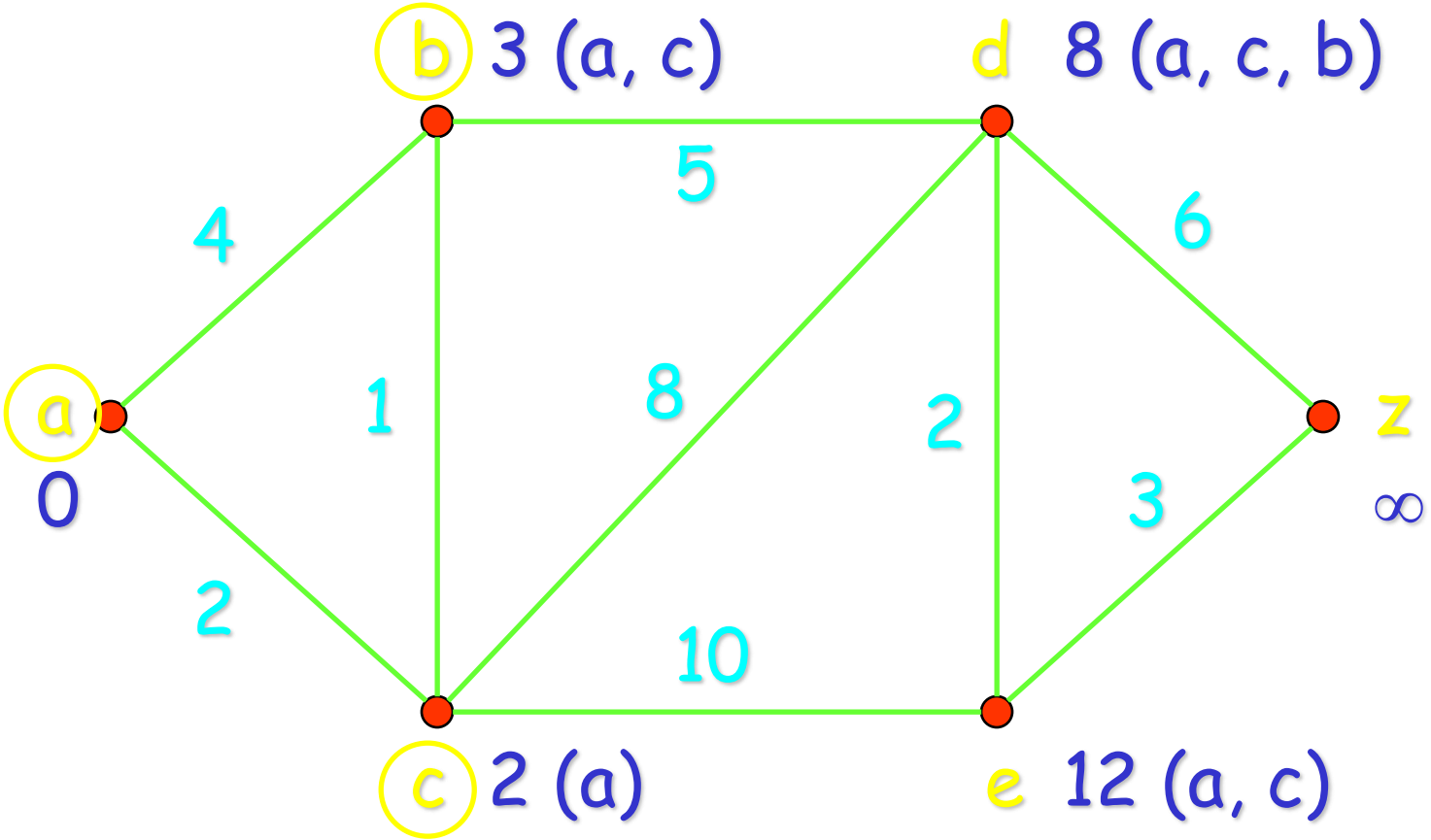
contoh:



Step 2

Algoritma Dijkstra

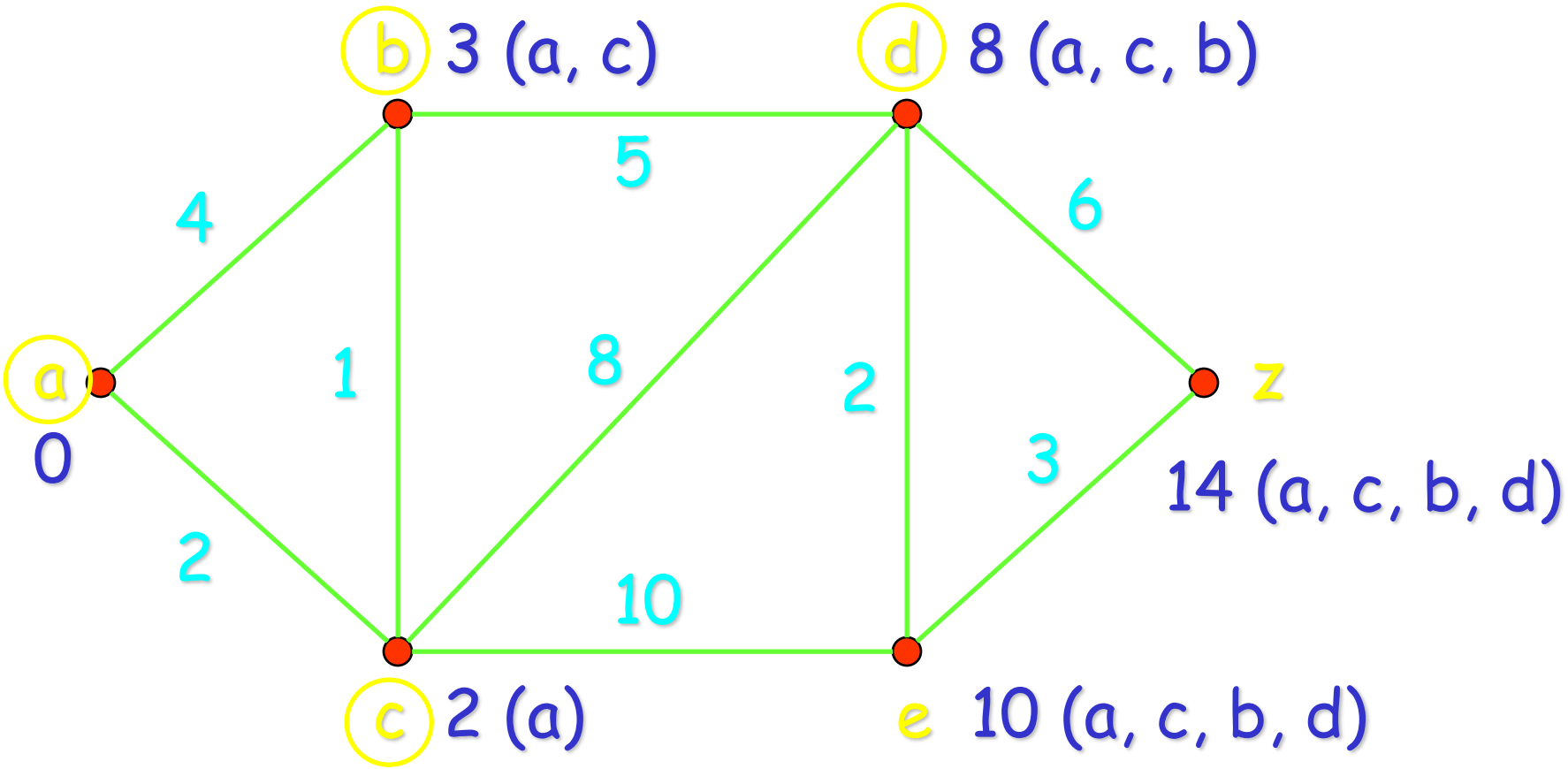
contoh:



Step 3

Algoritma Dijkstra

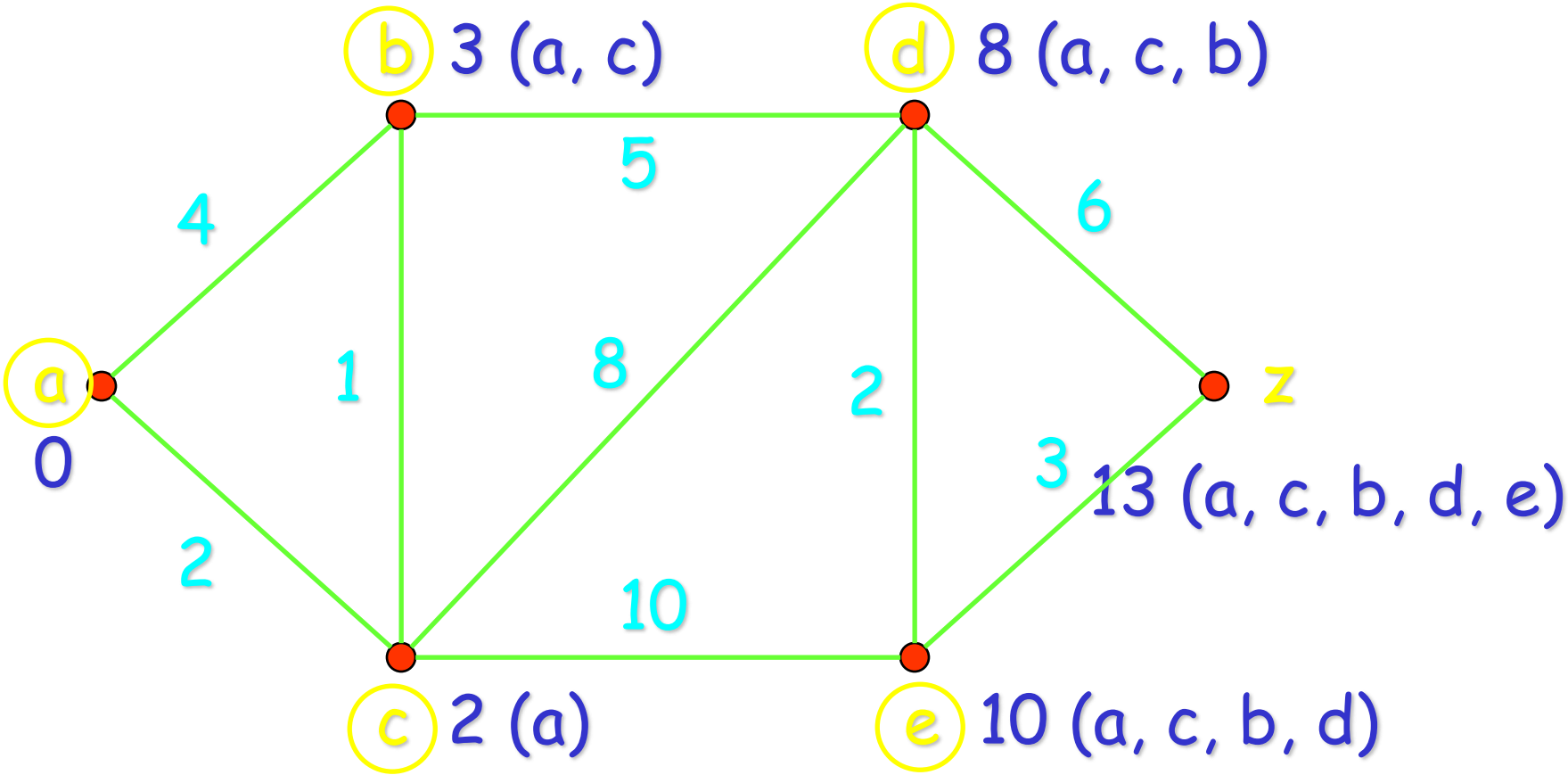
contoh:



Step 4

Algoritma Dijkstra

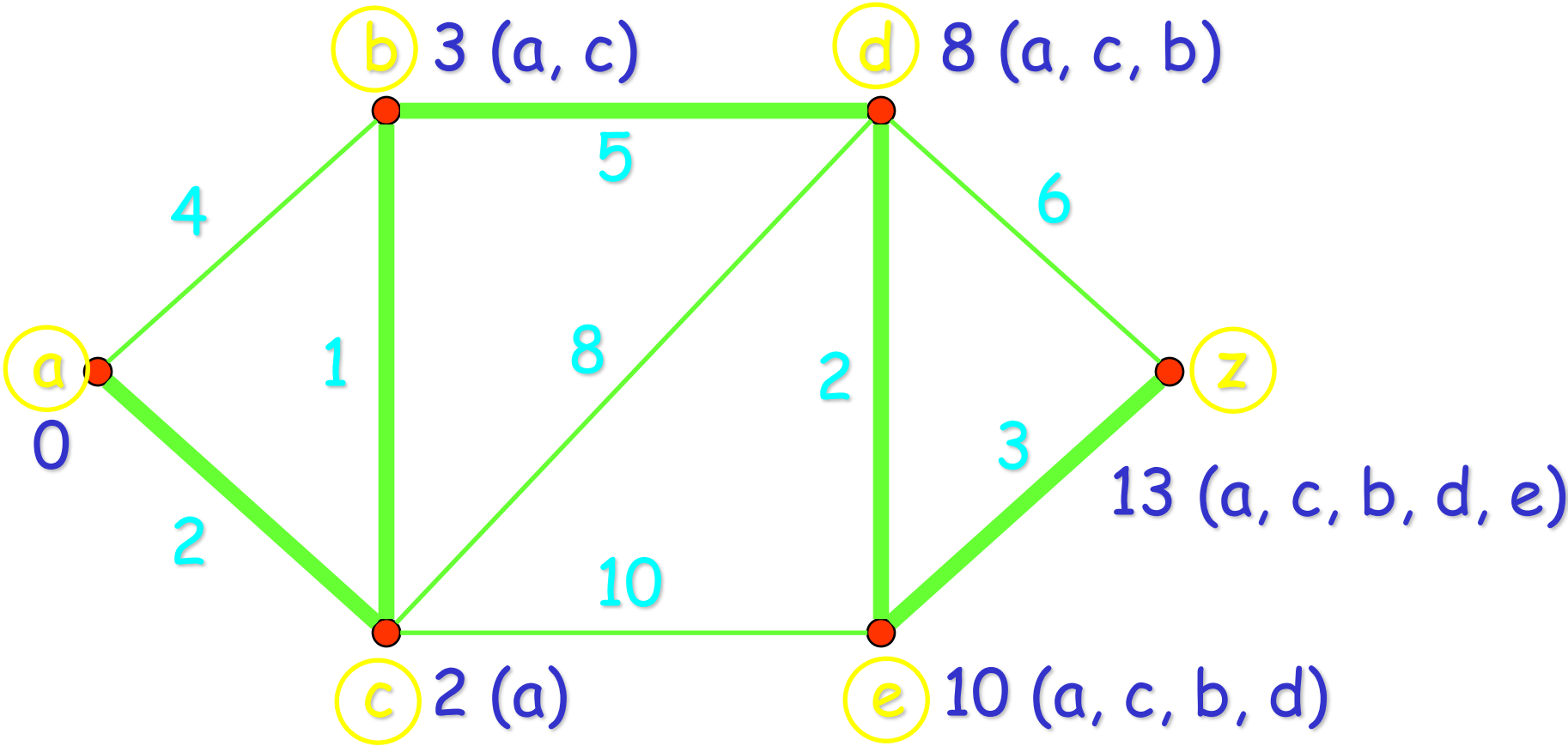
contoh:



Step 5

Algoritma Dijkstra

contoh:



Step 6

Algoritma Dijkstra

Teorema: Algoritma Dijkstra mencari panjang lintasan terpendek antara dua simpul dalam graf yang terhubung, sederhana, tidak berarah, dengan pembobot.

(Lihat Buku)

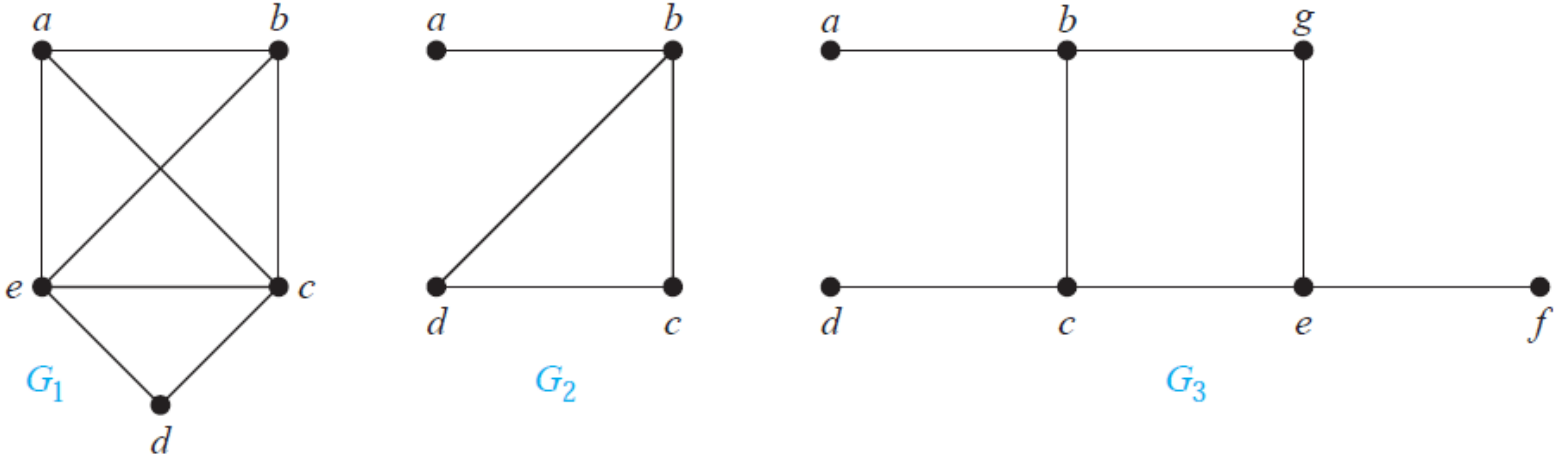
Teorema: algorithm Dijkstra menggunakan $O(n^2)$ operasi (penjumlahan dan perbandingan) untuk mencari panjang lintasan terpendek antara dua simpul dalam graf yang terhubung, sederhana, tidak berarah, dengan pembobot.

TSP

Lintasan dan sirkit Hamilton

- **Definisi**
  - Sebuah lintasan $x_0, x_1, \ldots, x_{n-1}, x_n$ didalam gaf G=(V,E) disebut lintasan Hamilton jika $V=\{x_0, x_1, \ldots, x_{n-1}, x_n\}$ dan $x_i \neq x_j$, untuk $0 \leq i < j \leq n$.
  - Sebuah sirkit $x_0, x_1, \ldots, x_{n-1}, x_n, x_0$ dng (n>1) didalam graf G=(V,E) disebut sirkit Hamilton jika $x_0, x_1, \ldots, x_{n-1}, x_n$ adalah lintasan Hamilton.

Graf manakah yang memiliki sirkit Hamilton ?



- G_1: a,b,c,d,e,a
- G_2: tdk memilikisirkit Hamilton, ttp punya lintasan Hamilton
- G_3: tidak memiliki sirkit maupun lintasan Hamilton

Traveling Salesman Problem

***Traveling salesman problem* (TSP)** adalah salah satu masalah klasik dalam ilmu komputer .

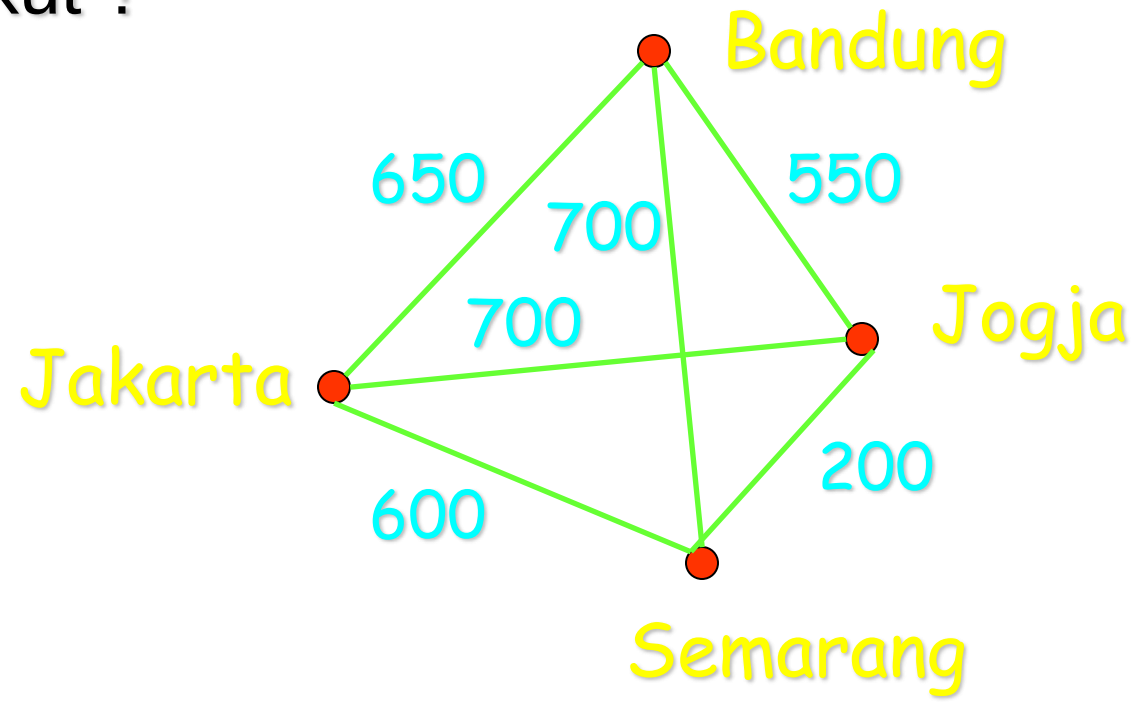
Seorang *salesman* ingin mengunjungi sejumlah kota dan kembali ke titik awal pemberangkatan. Tentunya ia ingin menghemat waktu dan energi, sehingga ia ingin mengetahui lintasan terpendek dalam perjalanannya.

Kita dapat merepresentasikan kota-kota dan jarak nya dengan sebuah graf ber-pembobot, graf lengkap, dan tidak berarah.

Masalah: tenntukan **sirkit dengan bobot total yang minimum dan setiap simpul dikunjungi tepat satu kali**.

Traveling Salesman Problem

Contoh : Lintasan mana yang akan diambil oleh traveling salesman untuk mengunjungi kota-kota berikut ?



Solusi: lintasan terpendek adalah Jogja, Semarang, Jakarta, Bandung, Jogja; yaitu sepanjang 2,000 Km.

Traveling Salesman Problem

Masalah TSP dapat dinyatakan sebagai berikut:

Pertanyaan: Diberikan *n* buah simpul. Berapa banyak cycle C_n yang berbeda, yang dapat kita bentuk dengan menghubungkan simpul-simpul tersebut dengan garis ?

Solusi : pertama kita pilih titik awal. Kemudian kita memiliki (n – 1) pilihan untuk simpul ke dua dalam siklis, (n – 2) untuk ketiga, dst, sehingga ada (n – 1)! Pilihan untuk seluruh siklis.

Akan tetapi, jumlah ini termasuk siklis yang identik yang dibentuk dalam arah kebalikannya. Oleh karena itu jumlah siklis yang benar-benar berbeda adalah

$$C_n = (n-1)!/2.$$

Traveling Salesman Problem

Hingga kini, **belum ada algoritma** yang dapat menyelesaikan masalah traveling salesman problem dengan ***polynomial worst-case time complexity*** .

Ini artinya untuk jumlah simpul yang banyak, penyelesaian *traveling salesman* problem tidak praktis.

Dalam kasus ini, kita dapat menggunakan **algoritma pendekatan** yang efisien dalam menentukan sebuah lintasan, dimana lintasan yang diperoleh mungkin sedikit lebih panjang dibanding lintasan traveling salesman tetapi dengan kompleksitas perhitungan polinomial, mis. dengan jaringan syaraf tiruan.