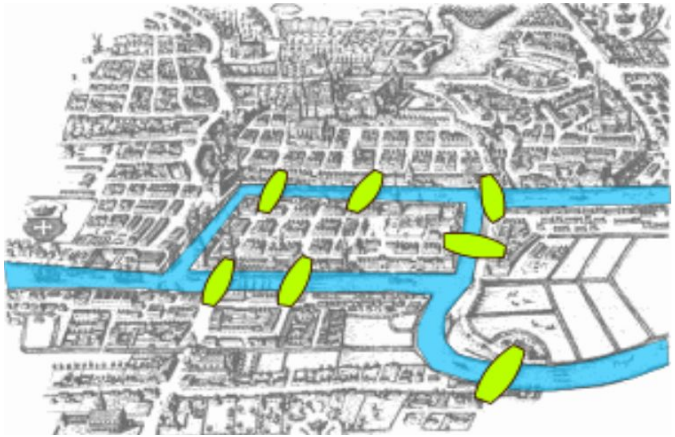
Graf/Çizge (Graph) Veri Yapısı

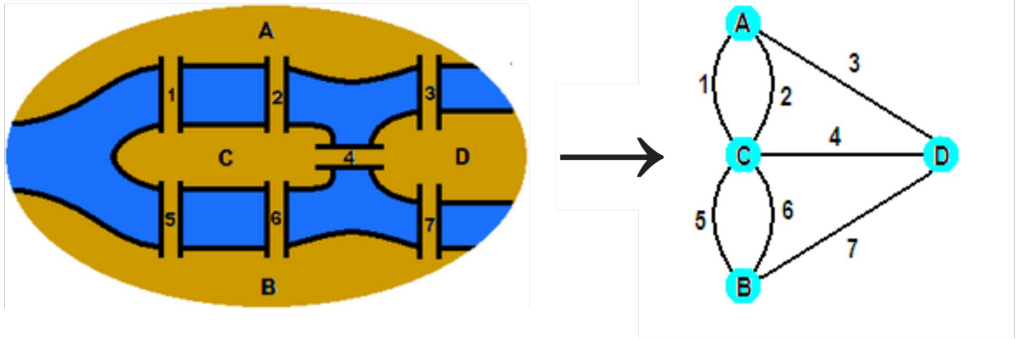
Dr. Hakan TEMİZ

Çizge

Graf teorisi İsviçreli matematikçi Leonhard Euler tarafından, ünlü Konigsberg probleminin çözümünde ortaya atılmıştır.

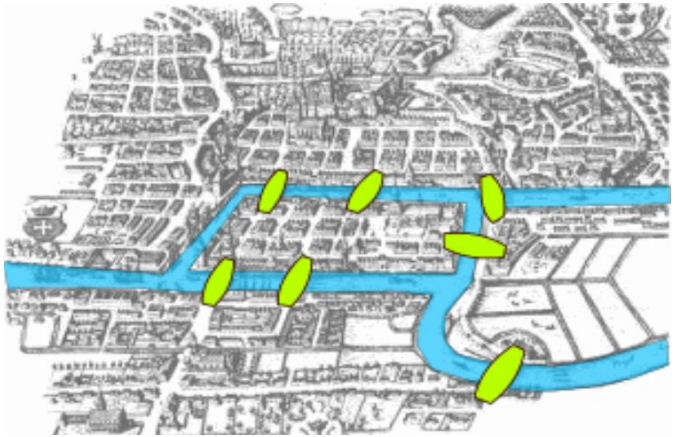
Königsberg kentinde, Pregel nehri şehri dört bölüme ayırmaktadır ve nehir üzerinde bu bölgeleri birleştiren yedi köprü ve iki adacık bulunmaktadır. Acaba, bütün köprülerden sadece ve sadece bir defa geçmek koşulu ile bir yürüyüş yapılabilir mi?

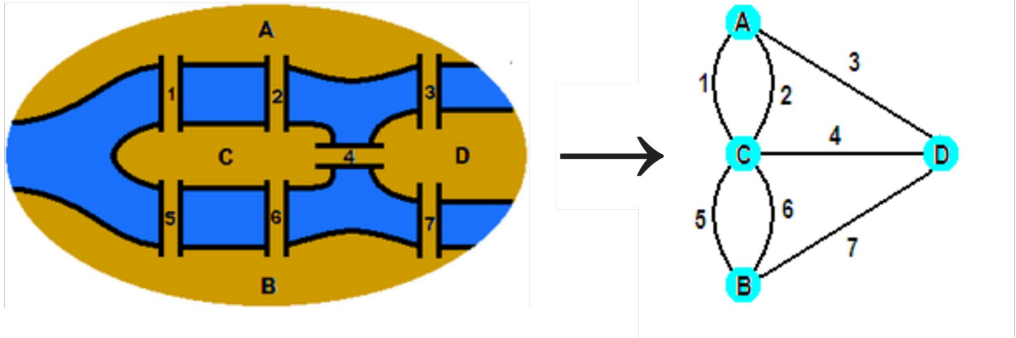




Çizge

Problemin basitleştirilmesi adına, kara parçaları harflerle; köprüler ise sayılarla işaretlenmiştir. Daha basit bir temsil için, gereksiz bileşenler atıldıktan sonra, kara parçaları noktalar, köprüler ise bu noktaları birleştiren çizgiler şeklinde gösterilmiştir (sağ). Bu yeni gösterim üzerinde, problem, yedi düğümün (nokta) her birini, çizgiler üzerinden ilerleyerek, **yalnızca bir kez** kullanarak dolaşmaya dönüşür. Bu gösterim biçimi, çizgelerin temelini oluşturmaktadır.





Çizge

- Graflar, temelde, nesneleri ve aralarındaki ilişkiyi temsil etmek amacıyla kullanılır. Bir çizge, boş olmayan bir **V** kümesi ve her bir elemanı V kümesindeki elemanlardan sırasız çiftleri barındıran bir **E** kümesi ile tanımlanır. Bir çizge,
  G = (V, E) şeklinde ifade edilir. Burada,

  V, graftaki düğümlerin (boş olmayan) bir kümesidir. {v1, v2, ..., vn} şeklinde de ifade edilir. Düğümlere, kimi zaman nokta, köşe; ingilizce dilinde node, vertex veya point gibi isimler de verilmektedir.

  E, graftaki düğüm çiftlerini; yani, birbirleriyle doğrudan bir bağlantıya sahip düğüm nesneleri çiftlerinden oluşan bir kümesidir. {e1,e2,...,en} şeklinde de ifade edilir. Düğümler arasındaki bağlantılara kenar, yay (edge, arc) veya bağ (link, connection) adları da verilmektedir.

Çizge

- Bir **e = {v, w}** şeklinde ifade edilen bir kenarın, v ve w düğümlerini birbirine bağladığı (birleştirdiği) söylenir. Kimi durumda, "vw kenarı" veya "wv kenarı" şeklinde bir ifadeyle de temsil edilir. Diğer bir deyişle vw (veya wv) kenarının, v ve w düğümlerini birbirine bağladığı anlamına gelir. İki düğüm, bir e kenarı ile birbirine (doğrudan) bağlı ise "bitişiktir" (adjacent) denir. Benzeri mantıkla, herhangi iki kenarın da aynı ortak düğümle bağlantısı varsa, bu kenarlara da "bitişik" denir.

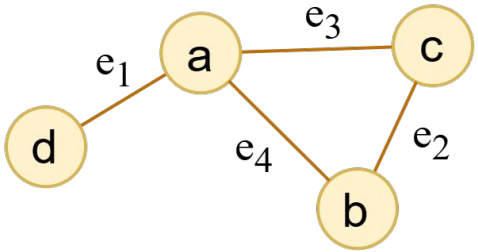
- Bir G grafı komşuluk ilişkisiyle gösteriliyorsa $G_{dd}=(d_i,d_j)$, ... ; bitişiklik ilişkisiyle gösteriliyorsa $G_{dk}=(d_i, k_j)$, ... şeklinde yazılır. Burada, G_{dd} , düğüm-düğüm; G_{dk} ise düğüm-komşu ilişkisini belirtir.

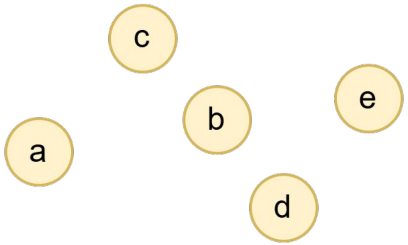
Çizge - Uygulamalar

- web'in yapısının etüdünde; örneğin, arama motorlarında,
- sosyal ağlarda; insanların tanışıklık ve arkadaşlık ilişkilerinin, akademik işbirliği içerisindeki akademisyenler ağının modellenmesinde, belirli bir etmenler kümesi içerisinde hangisinin diğer(ler)ini etkilediği vb. durumların anlaşılmasında,
- bilgisayar ağlarının modellenmesi ve analizinde: en kısa veya minimum maliyetli yolların keşfinde,
- yazılım tasarım uygulamalarında; modül bağımlılıklarının temsilinde,
- ulaşım sektöründe; havayolu güzergahlarının veya otoyol ağlarının modellenmesinde,
- Belgelerin, makalelerin, patentlerin vb. materyallerin alıntılanma durumlarının belirlenmesinde,
- derin öğrenme algoritmalarında,
- ...

Çizge

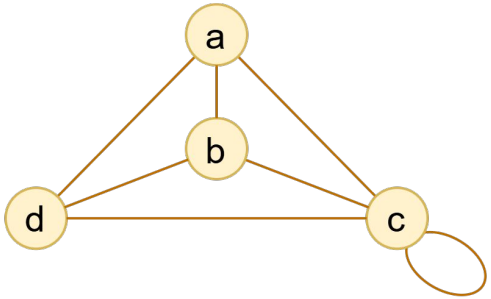
- G_{dd} (düğüm-düğüm) biçiminde yazılırsa,
- G_{dd}=(a,b),(a,c),(a,d),(b,c)
- G_{dk} (düğüm-kenar) biçiminde yazılırsa,
- G_{kd}= (a,e_1), (a,e_3), (a,e_4), (b,e_2), (b,e_4), (c,e_2), (c,e_3), (d,e_1) elde edilir.
- Bir çizgenin kenarlar kümesi boş ise; yani, hiç kenar yoksa, **"Boş Graf"** (null/empty graph) olarak adlandırılır.

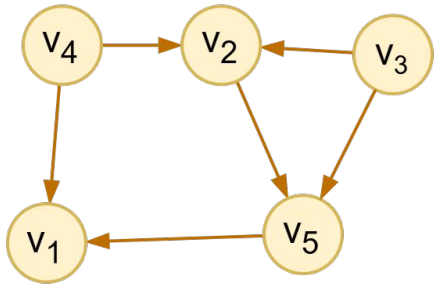




Çizge

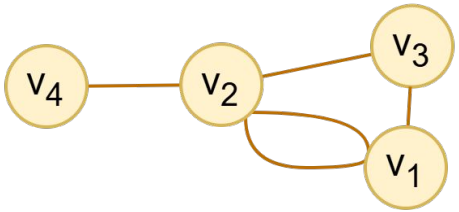
- Bir çizgenin kenarları düğümler arasında herhangi bir yön bilgisi vermiyorsa veya düğümler arasındaki hareket yönünün herhangi bir önemi yoksa, bu tür bir çizgeye **"yönsüz çizge"** adı verilir.
- Bunun tersi olarak, düğümler arasında oklara sahip kenarlar aracılığıyla bir yön/akış vb. bilgisi veriliyorsa bu tür çizgelere de **"yönlü çizge"** adı verilir.
- Bir yönsüz çizgede (v, w) düğüm ikilisi şeklinde verilen bir kenarın verilmesinde düğümlerin (örnek için, v ile w) sırasının bir önemi yoktur. Diğer taraftan, yönlü bir çizgede (v, w) şeklinde bir bağ ilişkisi verildiğinde sıralama önemlidir. Yönlü çizgelerde kenarlarda verilen ok yönü, bir veri akışının yönünü, herhangi bir sürecin ilerleme yönünü vb. belirtir.





Çizge

- Aynı düğümler arasında 1 ‘den fazla kenar bulunabilir. Bu türden kenarlara **"çokkatlı kenar"** denir. Böyle kenarlara sahip çizgelere de **"çoklu çizge"** adı verilir. Aşağıda, $\mathbf{v_1}$ ile $\mathbf{v_2}$ arasında iki farklı kenar, yani çokkatlı kenar vardır. Bu nedenle, bir çoklu çizgedir. m adet kenarlı kenar için, **m katlı kenar** adlandırması yapılır.
- Yönlü çizgede döngü ve katlı kenar yoksa **"yönlü basit çizge"** denir. Yönlü basit çizgede her sıralı (v, w) düğüm ikilisi arasında en fazla bir kenar bulunur. Bir yönlü çizge, katlı kenar içeriyorsa, **"yönlü çoklu çizge"** denir. (v, w) düğüm ikilisi arasında m adet kenar varsa, (v, w) kenarı **m katlıdır** denir. Bir çizge içerisinde, bazı kenarlar yönsüz, diğerleri ise yönlü olabiliyorsa, bu tür çizgelere **"karışık çizge"** adı verilir.



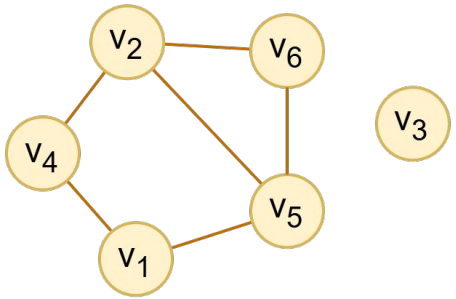
Çizge

Tür	Kenarlar	Çoklu Kenar Var Mı?	Döngü Var Mı?
Basit çizge	Yönsüz	Hayır	Hayır
Çoklu çizge	Yönsüz	Evet	Hayır
Sözde çizge	Yönsüz	Evet	Evet
Yönlü Basit çizge	Yönlü	Hayır	Hayır
Yönlü Çoklu çizge	Yönlü	Evet	Evet
Karışık çizge	Yönlü ve Yönsüz	Evet	Evet

Çizge - Derece - Yönsüz Çizge

- Bir w düğümüne bağlı kenarların toplam sayısına o düğümün **derecesi** (degree) denir ve **deg(w)** şeklinde ifade edilir. Düğümün derecesi sıfır ise o düğüme izole düğüm denir. Kenar ve bağ sayısı sonlu ise **sonlu çizge** denir.

$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ düğümleriyle, $E = \{v_1v_4, v_1v_5, v_2v_4, v_2v_5, v_2v_6, v_5v_6,\}$ bağlantılarına sahip bir G grafı:



- Bu çizgede, v_3 düğümü **izole** bir düğümdür.

Çizge - Derece - Yönlü Çizge

- Yönlü bir çizgenin düğümlerinin derecesi $\mathbf{d_{deg}=d_{in}+d_{out}}$ şeklinde verilir. Burada,

  $\mathbf{d_{in}}$, yönü düğüme doğru olan kenarların sayısı;

  $\mathbf{d_{out}}$, , yönü düğümden diğer düğümlere doğru olan kenarların sayısıdır.

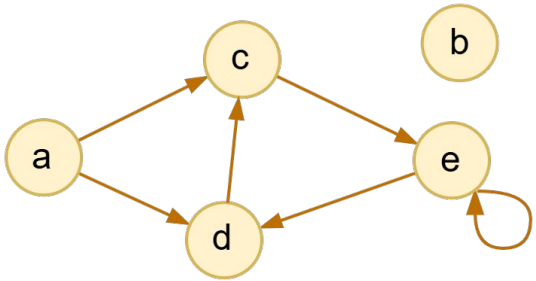
  $deg(a)=d_{in}(a)+ d_{out}(a) = 0 + 2 = 2$

  $deg(b)=d_{in}(b) + d_{out}(b) = 0 + 0 = 0$

  $deg(c)=d_{in}(c) + d_{out}(c) = 2 + 1 = 3$

  $deg(d) = d_{in}(d) + d_{out}(d) = 2 + 1 = 3$

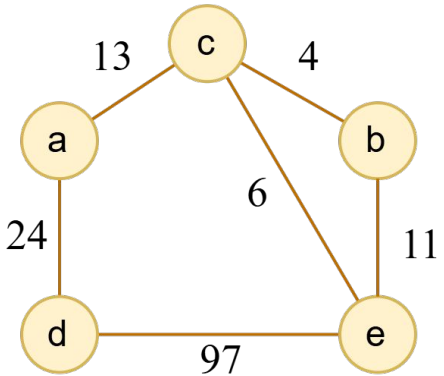
  $deg(e) = d_{in}(e) + d_{out}(e) = 2 + 2 = 4$



- Bir düğümdeki döngünün katkısı, hem gelen kenar hem de giden kenar olarak kabul edildiğinden, 2 ‘dir. Örnekte, e düğümündeki döngü d_{in} ve d_{out} için ayrı ayrı hesaplamıştır.

Çizge - Ağırlıklandırılmış Çizge

- Kenarlarına verilen değerlerin nicel bir büyüklüğü ifade ettiği çizgeler **ağırlıklandırılmış çizge** (weigthed graph) olarak adlandırılır.



Çizge - Gösterim

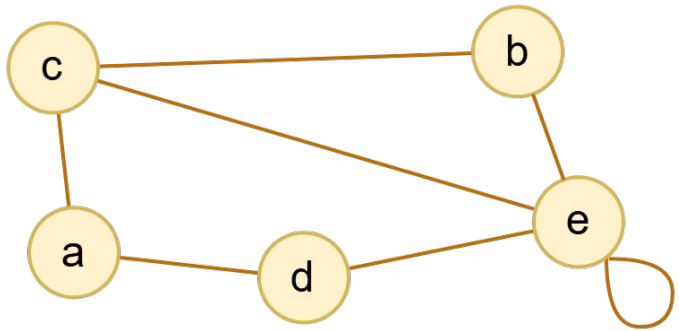
- Çizgeler şu şekillerde gösterilebilir:
  - Bitişiklik Listesi (Tablosu)
  - Bitişiklik (Adjacency) Matrisi
  - İlişki (Incidency) Matrisi

- Matrisler üzerinde, hem ilişki matrisi hem de bitişiklik matrisi biçiminde verilebilir.
- Yönsüz bir çizgenin matris üzerinde gösteriminde, Düğümler (veya kenarlar) satırlarda veya sütunlarda bulunabilir. Elemanların satır/sütun yerleşimine bağlı olarak oluşturulan iki matrisin birbirinin transpozu olacağı açıktır.

Çizge - Gösterim - Bitişiklik Tablosu -Yönsüz Çizge

- Bitişiklik listesi ile gösterimde, her satırda, bir düğümün komşuluk ilişkisinde olduğu düğümler listelenir. Yönsüz bir çizge için, bir düğümün komşuları sıralamanın önemi olmaksızın aynı satırda verilir. Komşu düğüm sıralamasının önemi yoktur.

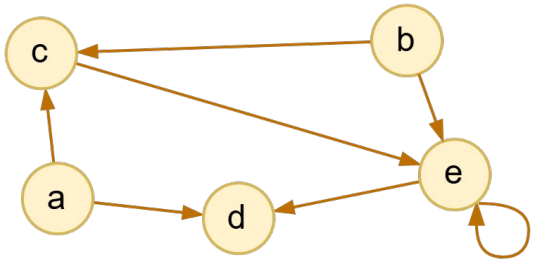
Düğüm	Komşu Düğümler
a	c, d
b	c, e
c	a, b, e
d	a, e
e	b, c, d, e



Çizge - Gösterim - Bitişiklik Tablosu -Yönlü Çizge

- Yönlü çizgenin liste ile gösteriminde, bir d düğümünün komşularından, yalnızca, bağlantı kenarındaki akış yönünün d düğümünden ilgili komşu düğümlere doğru olan komşuları verilir. Kenardaki akış yönünün komşudan d düğümüne olan komşu düğümler d satırında verilmez. Komşu düğümlerden d düğümüne doğru olan bir yönlü ilişki, ilgili komşunun kendi satırında, d düğümüne yapıldığına benzer şekilde verilir.

Düğüm	Komşu Düğümler
a	c, d
b	c, e
c	e
d	-
e	d, e



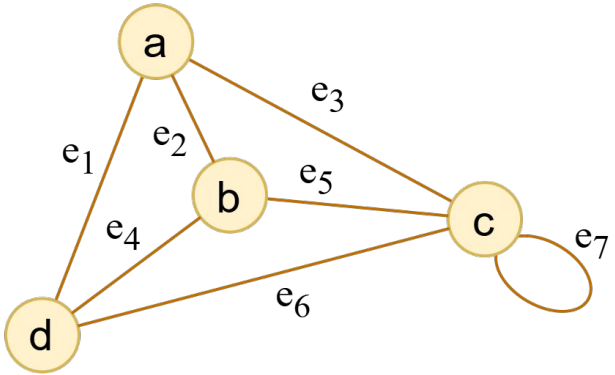
- c ‘den e ‘ye doğru yönlü komşuluk ilişkisi c düğümünün kendi satırında; a ve b düğümlerinden c ‘ye doğru yönlü komşuluk bilgisi ise a ve b düğümlerindeki satırlarda verilmiştir.

Çizge - Gösterim - İlişki (Incidency) Matrisi

- Çizgedeki, düğüm ve kenarlar arasındaki ilişkinin temsil edildiği yapı **"İlişki Matrisi"** dir. Bir kenar ve düğüm arasında bağlantı varsa, ilgili düğüm-kenar kesişimindeki hücrenin değeri 1; aksi halde 0 olur.

$A=[a_{ij}]$ $a_{ij} = \begin{cases} 1, & i \text{ düğümü } j \text{ kenarı ile ilişkili ise} \\ 0, & \text{aksi durumda} \end{cases}$

	e_1	**e_2**	**e_3**	**e_4**	**e_5**	**e_6**	**e_7**
a	1	1	1	0	0	0	0
b	0	1	0	1	1	0	0
c	0	0	1	0	1	1	1
d	1	0	0	1	0	1	0



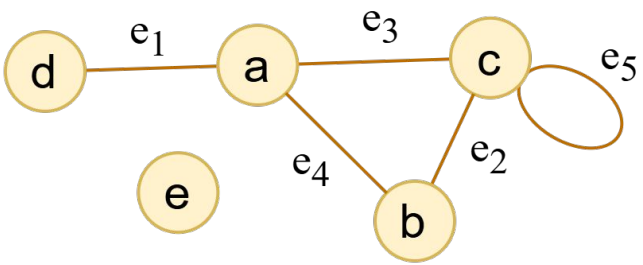
- Bir çizgenin ağırlıklı olması durumunda, düğüm ve kenar komşuluğunun varlığını 1 ile belirtmek yerine, **ilgili kenarın ağırlık değeri** verilebilir.

Çizge - Gösterim - Bitişiklik (Adjacency) Matrisi

- Matris üzerinde diğer bir temsil yöntemi **"Bitişiklik Matrisi"** dir. Bu sefer, sütunlar ve satırlar daima düğümleri ifade eder. Birbiri arasında bir kenar bağı varsa ilgili düğümlerin satır ve sütun kesişimlerine karşılık gelen hücrenin değeri 1; aksi halde 0 olur.

$A=[a_{ij}]$ $\quad a_{ij} = \begin{cases} 1, & i \text{ düğümü } j \text{ düğümü ile ilişkili ise} \\ 0, & \text{aksi durumda} \end{cases}$

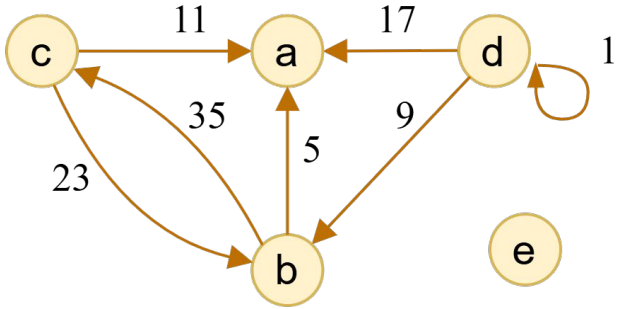
	a	b	c	d	e
a	0	1	1	1	0
b	1	0	1	0	0
c	1	1	1	0	0
d	1	0	0	0	0
e	0	0	0	0	0



Çizge - Gösterim - Bitişiklik Matrisi - Yönlü ve Ağırlıklı Çizge

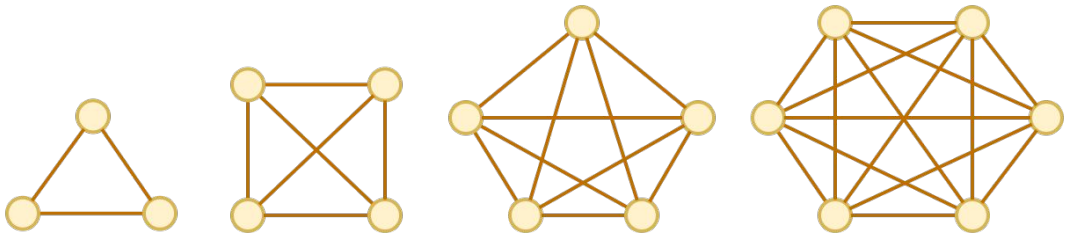
- Ağırlıklı ve/veya yönlü çizgelerin de bitişiklik matrisi üzerinde gösterimi mümkündür. Ağırlıklı çizgede ilgili matris hücresine iki düğüm arasındaki kenarın ağırlık değeri yazılır.
- Çizgenin yönlü olması durumunda, iki komşu düğüm arasındaki akış yönünün satırlardaki düğümlerden sütunlardaki düğümlere doğru olduğu kabul edilir ve değerleri bu satır ve sütunların kesiştiği hücrelere yazılır. ∞ sembolü, ilgili düğümler arasında belirtilen yönde (satırdaki düğümden sütundaki düğüme doğru) bir komşuluk ilişkisinin bulunmadığını belirtir.

	a	b	c	d	e
a	∞	∞	∞	∞	∞
b	5	∞	35	∞	∞
c	11	23	∞	∞	∞
d	17	9	∞	1	∞
e	∞	∞	∞	∞	∞



Çizge - Tamamlanmış (Complete) Çizge

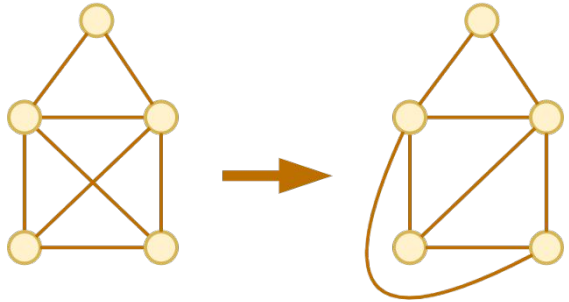
- Her bir düğümü arasında doğrudan bağlantı bulunan yönsüz çizgelere tamamlanmış çizge denir.
- Tamamlanmış bir çizgenin düğümlerinin dereceleri birbirine eşittir ve sayısı toplam düğüm sayısının bir eksiğidir. Yani, n düğümlü bir tamamlanmış çizgenin her düğümünün derecesi $n-1$ ve kenarlarının sayısı da $n(n-1)/2$ olur.
- Yönlü bir çizgenin tamamlanmış çizge olması için her düğümden her düğüme, düğümün kendisi de dahil olmak üzere bir yol bulunur.



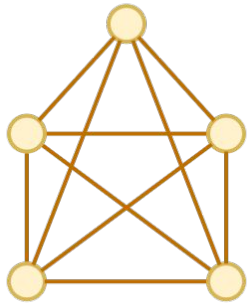
- Tüm düğümlerin derecelerinin eşit olduğu çizgeye **"Düzenli (Regular) Çizge"** denir. Tanım gereği, tamamlanmış çizge de bir düzenli çizgedir. Fakat tersi her zaman geçerli değildir.

Çizge - Düzlemsel (Planar) Çizge

- Hiçbir kenarı bir diğerini kesmeden çizilebilen çizgeye **“düzlemsel çizge”** denir. Aynı grafın alternatif çizimleri düğümlerin yerlerinin değiştirilmesi veya kenarların farklı yörüngelerden geçirilmesi ile üretilebilir.
- Hiçbir kenarının bir diğerini kesmediği bir çizim bulunabilmesi o çizgenin düzlemsel olması için yeterlidir. Yanda, sol tarafta verilen çizge, sağ taraftaki gibi yeniden çizilebilir. Bu durumda bu çizge bir düzlemsel çizgedir.



- Yandaki çizgeyi ise hiçbir kenarının birbirini kesmediği şekilde çizmek mümkün değildir. Bu nedenle düzlemsel graf değildir.



Çizge - Gezinme

Çizgelerde gezinme, bir şekilde tüm düğümlere ziyaret etmek anlamına gelir. Bunun için çok farklı metotlar geliştirilmiştir. Başlamadan önce bazı terimlerin bilinmesinde yarar vardır.

Yürüyüş/Dolaşı (Walk/Traverse): bir G grafında u düğümünden başlayıp, v düğümünde biten ve gezinme esnasında u ile v arasında geçilen ve birbiriyle ilişkili tüm düğümler ve kenarlar serisidir. Bir yürüyüşte, düğümler ve kenarlar üzerinden tekrar geçilebilir.

Uzunluk (Length): bir yürüyüşteki kenarların sayısıdır.

İz/Gezi/Tur (Trace/Trail/Tour): her kenardan yalnızca bir kez geçildiği fakat düğümler için böyle bir kısıtın olmadığı (düğümlerden çoklu geçilebilir) yürüyüşe iz veya gezi denir.

Devre/Çevrim (Cycle): aynı düğümle başlayıp aynı düğüm ile biten ize devre denir.

Yol: düğümlerden tekrarlı geçilmeden yapılan yürüyüşe yol denir.

Çevrim: kapalı bir yola çevrim adı verilir.

$\mathbf{d_i}$ **ve** $\mathbf{d_j}$ düğümleri arasında kesintisiz bir yol var ise, bu düğümlere bağlantılıdır denir.

Çizge - Gezinme

Çizgelerde gezinme, bir şekilde tüm düğümlere ziyaret etmek anlamına gelir. Bunun için çok farklı metotlar geliştirilmiştir. Başlamadan önce bazı terimlerin bilinmesinde yarar vardır.

Yürüyüş/Dolaşı (Walk/Traverse): bir G grafında u düğümünden başlayıp, v düğümünde biten ve gezinme esnasında u ile v arasında geçilen ve birbiriyle ilişkili tüm düğümler ve kenarlar serisidir. Bir yürüyüşte, düğümler ve kenarlar üzerinden tekrar geçilebilir.

Uzunluk (Length): bir yürüyüşteki kenarların sayısıdır.

İz/Gezi/Tur (Trace/Trail/Tour): her kenardan yalnızca bir kez geçildiği fakat düğümler için böyle bir kısıtın olmadığı (düğümlerden çoklu geçilebilir) yürüyüşe iz veya gezi denir.

Devre/Çevrim (Cycle): aynı düğümle başlayıp aynı düğüm ile biten ize devre denir.

Yol: düğümlerden tekrarlı geçilmeden yapılan yürüyüşe yol denir.

Çevrim: kapalı bir yola çevrim adı verilir.

$\mathbf{d_i}$ **ve** $\mathbf{d_j}$ düğümleri arasında kesintisiz bir yol var ise, bu düğümlere bağlantılıdır denir.

Çizge - Gezinme

- Çizgelerde dolaşma yapan birçok yöntem olmakla beraber en önemli iki tanesi şunlardır:
  - **Derinlik öncelikli arama** (depth first search - DFS)
  - **Genişlik öncelikli arama** (breadth first search - BFS)

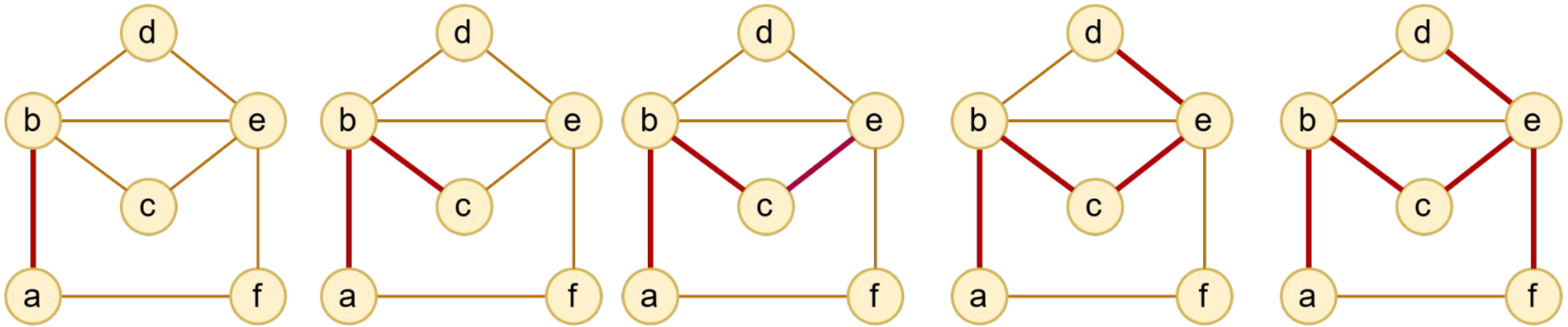
- Diğer algoritmaların önemli bir kısmı bu yöntemlere dayanmaktadır.

Çizge - Gezinme - Derinlik Öncelikli Arama (DFS)

- Derinlik öncelikli arama bir d düğümünün ziyaretiyle başlar.
- Burada, ziyaretten kasıt d düğümünün verisine ulaşmaktır.
- d düğümünden sonra, d düğümünün komşuları arasından bir tanesi seçilir ve seçilen üzerinde de aynı işlemler tekrar edilerek çizgedeki tüm düğümlerin ziyaret edilmesi sağlanır.
- Arama yaparken, her düğümün komşuluk listesindeki mevcut (gezilen, gezilmeyen gibi) konumları takip edebilmek için yığın veri yapısından yararlanılır.
- Bir d düğümünden daha ileriye gidilemediğinde (tüm komşuları aranmıştır veya hiç yoktur) yığından yeni alınan bir düğüm ile aramaya devam edilir.

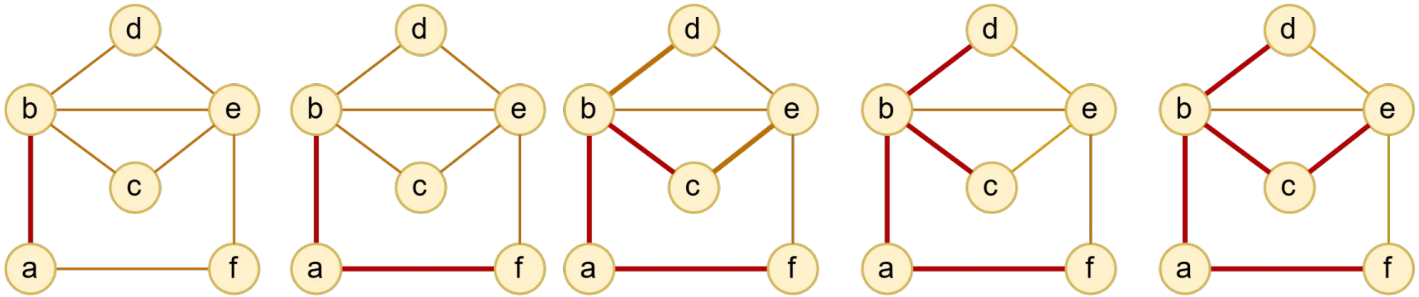
Çizge - Gezinme - Derinlik Öncelikli Arama (DFS)

- Aşağıdaki örnekte, gezinmeye a düğümü ile başlandıktan sonra sırasıyla b, c, e, d 'ye gidilir.
- d düğümünden sonra gidilebilecek yeni bir düğüm kalmadığı fark edildiğinde hemen geri dönülerek, ilk olarak e düğümünün komşularına bakılır ve gezinme esnasında henüz gidilmemiş olan f düğümü ziyaret edilerek gezinme tamamlanır.
- Gezinmenin her safhasında, gidilecek bir sonraki düğüm rastgele seçilerek ilerlenir.



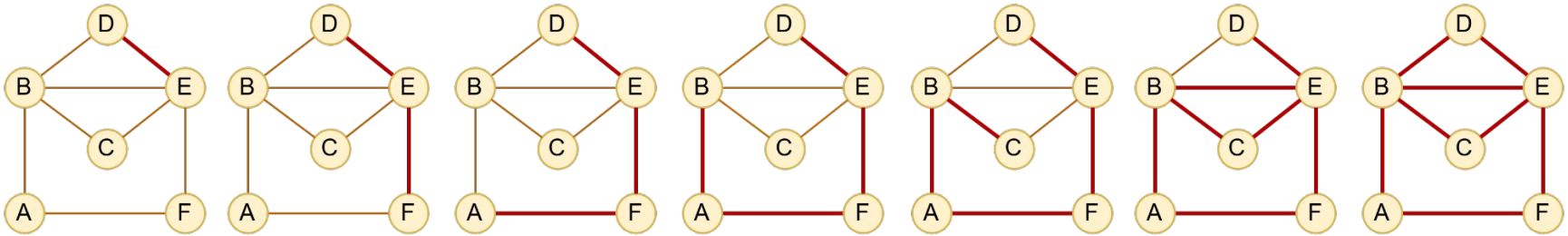
Çizge - Gezinme - Genişlik Öncelikli Arama (BFS)

- Bu yöntemde, gezinmeye bir d düğümünden başlanır ve d düğümünün tüm komşuları ziyaret edilir.
- Ardından, d 'nin ilk komşusu olan düğüm üzerinden gidilebilecek olan ve henüz ziyaret edilmemiş komşu düğümlerin ziyareti ile devam edilir. Bu süreç tüm yeni düğümler için tekrarlanır.
- En basit algoritmalardan biridir ve birçok önemli grafik algoritmasının temelidir.
- Prim'in minimum kapsayan ağaç algoritması (minimum spanning-tree) ve Dijkstra'nın tek kaynaklı en kısa yol (shortest-path) algoritması bunlardandır.



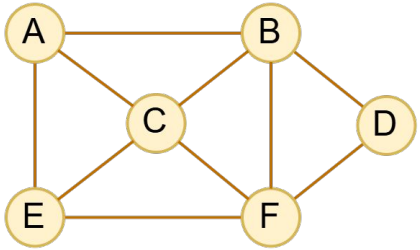
Çizge - Gezinme - Euler Turu

- Kenarlardan **yalnızca bir kez geçilerek** gerçekleştirilen gezinmeye **Euler Turu** denir. Düğümlerden tekrarlı geçilebilir. Bir Euler gezisi, başladığı düğümde sonlanırsa, **"Euler Çevrimi"** adını alır.
- Euler yolunun var olması için çizgenin tüm düğümlerinin bağlantılı ve dereceleri çift olan düğümlere sahip olması gerekir. Benzer mantıkla, tüm düğümlerinin derecesi çift olan bağlantılı bir çizge **Euler çizgesi**'dir.
- Aşağıdaki örnekte verilen çizge bir Euler çizgesidir. Çünkü,
- **DEFABCEBD** yolu bir **Euler çevrimidir**. Fakat **CEFABDEB** Euler turu **değildir**. Çünkü BC kenarı geziye dahil edilmemiştir.

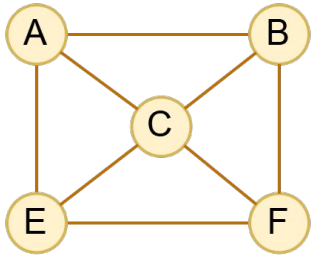


Çizge - Gezinme - Euler Turu

- Aşağıdaki çizge için, ABDFBCFECAE gezisi bir Euler turudur.

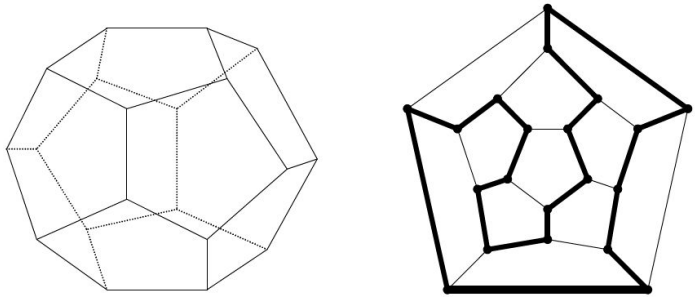


- Fakat aşağıdaki çizge ne Euler Turu ne de Euler (Çevrimi) Devresidir. Çünkü kenarlardan yalnızca bir kez geçerek tüm düğümleri gezmek mümkün değildir.



Çizge - Gezinme - Hamilton Çevrimi

- İrlandalı matematikçi Sir William Rowan Hamilton tarafından icat edilmiştir. Oyunun amacı bir gezi problemini çözmektir: Dünya üzerindeki 20 tane şehir her şehre yalnız bir kez uğramak ve geziyi başlanılan şehirde sonlandırmaktır.
- Hamilton, problemi basite indirgemek için dünyayı düzgün bir 12 yüzlü ve şehirleri de bu şeklin köşeleri şeklinde varsaymıştır. Böylece köşeleri birleştiren kenar çizgileri de şehirleri arası yolları temsil etmiştir. Düzgün 12 yüzlü, 12 adet beşgenden oluşan üç boyutlu bir top şeklinde ele alındığında, 20 köşe noktası ve 30 kenar çizgisine sahiptir. Oniki yüzlünün üç boyutlu görseline dair bir gösterim aşağıdaki resimde solda; iki boyuttaki projeksiyonu ise sağda verilmiştir.



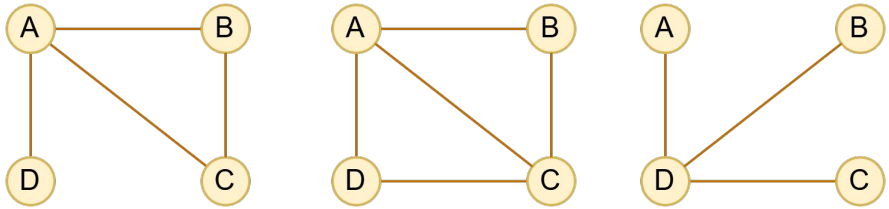
Çizge - Gezinme - Hamilton Çevrimi

- Hamilton, problemi bir graf üzerine şeklinde ele alarak çözüm bulmuş ve böylece Hamilton turu kavramı ortaya çıkmıştır. Buna göre, Hamilton gezisi, çizgedeki tüm düğümlerden yalnızca bir kez geçildiği bir gezidir. Gezi başlanılan düğümde sonlanırsa Hamilton Çevrimi adını alır. Hamilton turu içeren çizgeye Yarı-Hamilton çizgesi de denir.

  (a): CBAD turu nedeniyle Yarı-Hamilton çizgedir.

  (b): ABCDA turu nedeniyle Hamilton devresidir.

  (c): ne Hamilton Çevrimi, ne de Yarı-Hamilton olamaz. Çünkü, en az bir düğümden tekrar geçmeden tüm düğümleri gezmenin bir yolu yoktur.



(a) (b) (c)

- Hamilton Turlu çeşitli problemler bulunmaktadır. Bu tür problemlerdeki ama genellikle en düşük maliyetli (ağırlıklı) turu bulmaktır. Bunlardan bir tanesi gezgin satıcı (travelling salesman) problemidir.

Çizge - Gezinme - Gezgin Satıcı Problemi

- Ağırlıklandırılmış bir çizgede tüm düğümleri içine alan ve minimum ağırlık değeri yapılan kapalı bir dolaşının bulunması problemine **"Gezgin Satıcı Problemi", (Travelling Salesman Problem) TSP** denir.
- Böyle bir gezide tüm düğümlere uğranılır ve dolaşı bitip başlangıç noktasına geri dönüldüğünde diğer alternatiflere göre en az maliyetli (ağırlıklı) yol gidilmiş olur.
- Bu tür problemler, dağıtım şirketleri vb. uygulama alanlarında sıklıkla kullanılır. Diğer bir örnek, bilgisayar ağlarında, mesajın en kısa zamanda ulaşmasını sağlayan güzergahın ortaya konmasıdır.
- Problemde, dolaşılacak konumlar (şehirler, duraklar vb.) düğüm ve aralarındaki mesafeler (veya, diğer uygulama türleri için bir çeşit maliyet) kenarlar olarak temsil edilir.
- Şu durumda, amaç, düğüm tekrarı yapmadan en az maliyetle çevrim yapmak, yani tüm düğümleri gezip başlangıç düğümünde durmaktır.

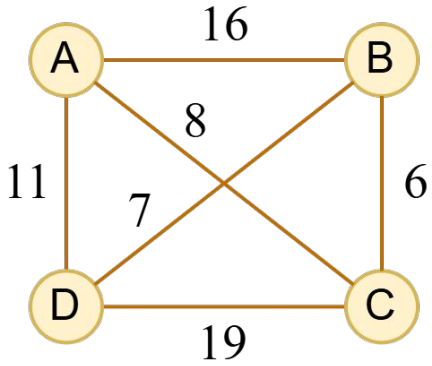
Çizge - Gezinme - Gezgin Satıcı Problemi

- Örnek olarak, aşağıdaki tamamlanmış çizgede toplam Hamilton çevrimi sayısı = (n−1)! / 2 = 3 olur. Burada n düğüm sayısı = 4 'tür.
- Bu çevrimler ve maliyetleri şöyledir:

ACBDA⇒8+6+7+11=32

ACDBA⇒8+19+7+16=50

ABCDA⇒16+6+19+11=52



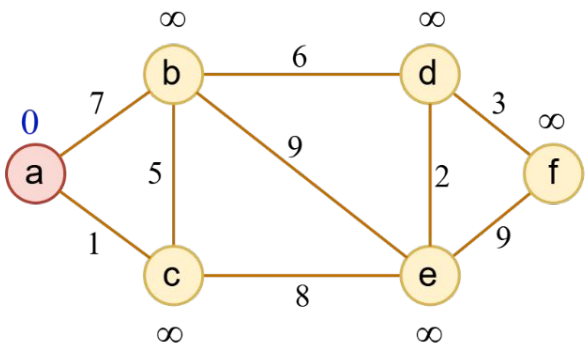
- Görüldüğü üzere, en az maliyetli çevrim **ACBDA** ‘dır.

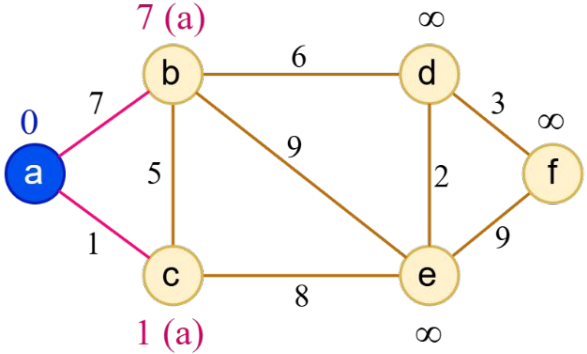
Çizge - Gezinme - Dijkstra Algoritması

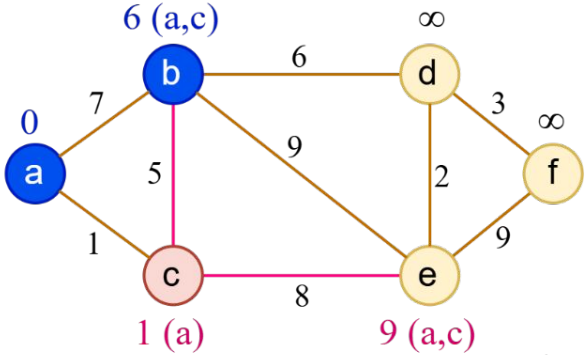
- Dijkstra algoritması, belirli bir başlangıç düğümüne göre en kısa geziyi bulmak üzere geliştirilmiş bir algoritmadır.
- Bir düğümden diğer tüm düğümlere olan yerleri belirler.
- Ağırlık değerleri 0 veya daha büyük değerlere sahip yönlü ve ağırlık çizgeleri için geliştirilmiştir.
- Negatif ağırlıklı yönlü çizgeler için Bellman-Ford algoritması geliştirilmiş olmakla beraber, bu algoritmaya burada değinilmeyecektir.

Çizge - Gezinme - Dijkstra Algoritması

- Dijkstra algoritması belirli bir düğüm ile başlar. Örnekte, a düğümü.
- a 'nın kendisine olan mesafesi 0; kendisine 0 yazıyoruz. a 'dan, a'nın tüm komşularına olan mesafeleri hesaplayarak not ediyoruz.
- a 'nın komşuları b ve c 'dir. ve uzaklıkları sırasıyla 7 ve 1 'dir. Üzerlerine not ediyoruz. Parantez içerisinde de, o düğüme ulaşmak için geçilen düğümleri sırasıyla not ediyoruz.
- Şu durumda b ve c 'nin uzaklık değerlerinin yanında parantez içerisinde (a) yazma nedeni, bu düğümlere sadece a düğümünden geçerek gelindiği içindir.







Çizge - Gezinme - Dijkstra Algoritması

- a 'nın tüm komşularına uğrandıktan sonra, işlemin aynısını komşulardan biri için sürdürüyoruz. Bu süreci diğer tüm gezilmemiş düğümler için uyguluyoruz.
- Bir düğüme farklı yoldan gelindiğinde mesafe kısalıyorsa, yeni mesafe değeri ve güzergah bilgisi ile güncelliyoruz. b 'ye a 'dan gelindiğinde maliyet 7; a-c düğümleri üzerinden 6 olduğunu görürüz. Bu gezinti daha kısa (az maliyetli) olduğundan, b 'nin bilgilerini, mesafe 6 ve güzergah bilgisi (ac) olacak şekilde güncelliyoruz.
- Tüm süreci bu şekilde, son düğüme varıncaya dek devam ettiriyoruz.