


YZM 2116

Veri Yapıları



Yrd. Doç. Dr. Deniz KILINÇ
Celal Bayar Üniversitesi
Hasan Ferdi Turgutlu Teknoloji Fakültesi
Yazılım Mühendisliği

BÖLÜM - 11

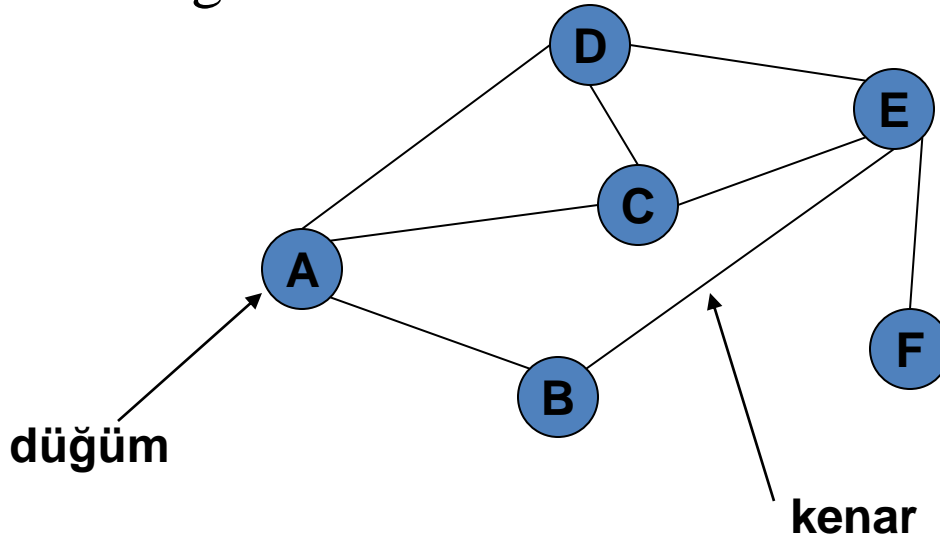
Bu bölümde,

- Graph (Çizge - Graf)
- Terminoloji
- Çizge Kullanım Alanları
- Çizge Gösterimi
 - Komşuluk Matrisi
 - Komşuluk Listesi
- Çizge Üzerinde Gezinme
 - DFS
 - BFS

konusuna değinilecektir.

Çizge (Graph) Giriş

- **Köşe** (vertex) isimli düğümlerden ve **kenar** (edge) isimli köşeleri birbirine bağlayan bağlantılardan oluşan veri yapısıdır.
- Ağaçlar gibi çizgeler de **doğrusal olmayan** veri yapıları grubuna girerler.

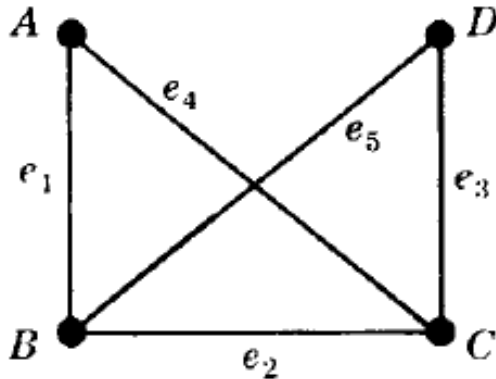


Çizge (Graph) Giriş (devam...)

- Bir G çizgesi $G(V, E)$ şeklinde gösterilir:
 - a. **$V = V(G)$ Kümesi:** Küme elemanları, G 'nin düğümleri (**nodes**), noktaları (**points**) veya köşeleri (**vertices**)
 - b. **$E = E(G)$ Kümesi:** Küme elemanları G 'nin kenarları (edges) olarak adlandırılan sırasız düğüm ikililerini içerir.
- Eğer bir $e = \{u, v\}$ kenarı varsa, u ve v düğümlerinin komşu (***adjacent or neighbors***) olduğu söylenir.
- Bu durumda, u ve v **e 'nin uç noktaları** (**endpoints**) olarak adlandırılır ve e 'nin u ve v 'yi bağladığı söylenir.

Çizge (Graph) Giriş (devam...)

- Çizgeler, **düzlemsel diyagramlarla** gösterilir.
- V kümesindeki her v düğümü bir nokta (ya da küçük çember) ile temsil edilir ve her $e = \{v_1, v_2\}$ kenarı, v_1 ve v_2 uç noktalarını bağlayan bir çizgi ile gösterilir.
- **Örnek:**



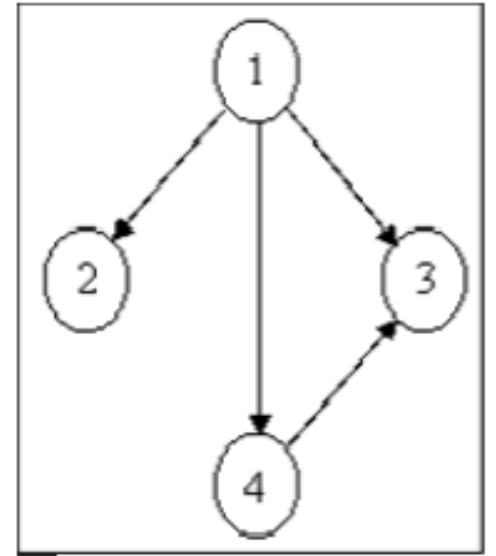
$$V = \{A, B, C, D\}$$

$$E = \{e_1, e_2, e_3, e_4, e_5\}$$

$$e_1 = \{A, B\}, e_2 = \{B, C\}, e_3 = \{C, D\}, e_4 = \{A, C\}, e_5 = \{B, D\}.$$

Yönlü ve Yönsüz Kenar

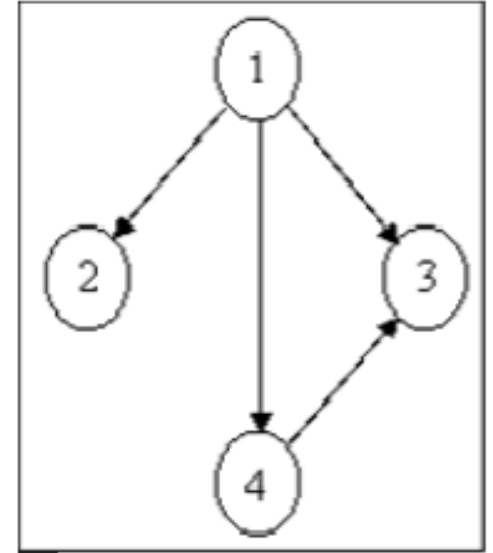
- **Yönsüz Kenar (undirected edge):** Çizgi şeklinde yönü belirtilmeyen kenarlar yönsüz kenarlardır.
 - Yönsüz kenarlarda $(v1,v2)$ olması ile $(v2,v1)$ olması arasında fark yoktur.
- **Yönlü Kenar (directed edge):** Ok şeklinde gösterilen kenarlar yönlü kenarlardır.



$G_1=(V,E)$
 $V_1=\{1,2,3,4\}$
 $E_1=\{(1,2),(1,3),(1,4),(4,3)\}$

Terminoloji

- 1. Komşu Köşeler (Adjacent):** Aralarında doğrudan bağlantı (kenar) bulunan i ve j köşeleri komşudur. Diğer köşe çiftleri komşu değildir.
- 2. Bağlantı (incident):** Komsu i ve j köşeleri arasındaki kenar (i, j) bağlantıdır.
- 3. Bir Köşenin Derecesi (degree):** Bir köşeye bağlı olan kenarların sayısıdır.



$$G_1 = (V, E)$$
$$V_1 = \{1, 2, 3, 4\}$$
$$E_1 = \{(1, 2), (1, 3), (1, 4), (4, 3)\}$$

Soru1: Komşu olmayan köşeler hangileridir?

Soru2: 1'in derecesi kaçtır?

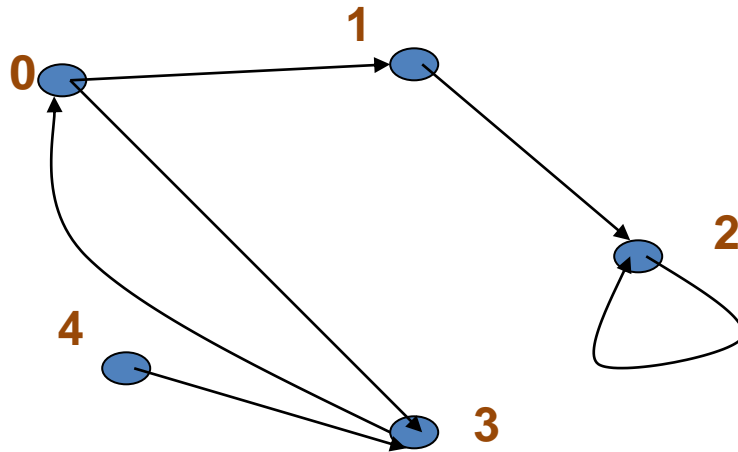
Terminoloji (devam...)

- 4. Yönsüz Çizge (undirected graph):** Tüm kenarları yönsüz olan çizgeye yönsüz çizge denilir. Yönsüz çizgede bir köşe çifti arasında en fazla bir kenar olabilir.
- 5. Yönlü Çizge (directed graph, digraph):** Tüm kenarları yönlü olan çizgeye yönlü çizge adı verilir. Yönlü çizgede bir köşe çifti arasında ters yönlerde olmak üzere *en fazla iki kenar* olabilir.
- 6. Döngü (Loop):** (i, i) şeklinde gösterilen ve bir köşeyi kendine bağlayan kenar.

Terminoloji (devam...)

Yönlü Çizge ve Döngü (Loop)

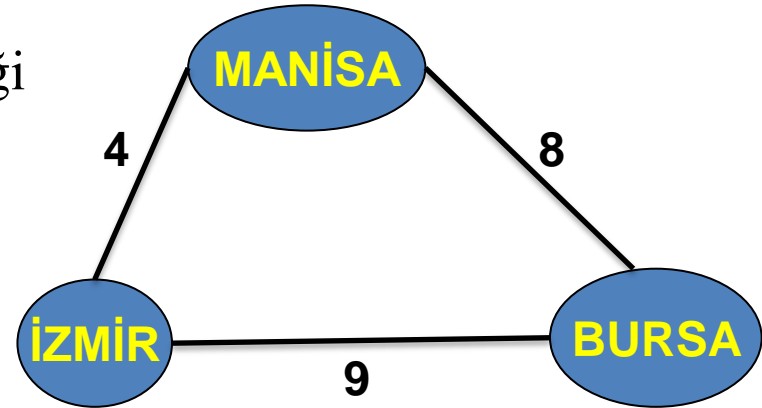
- $G=(V, E)$
- $V=\{0,1,2,3,4\}$
- $E=\{(0,1), (1,2), (0,3), (3,0), (2,2), (4,3)\}$



Terminoloji (devam...)

7. Ağırlıklı (Weighted) Çizge: Çizge kenarları üzerinde ağırlıkları olabilir. Eğer kenarlar üzerinde ağırlıklar varsa bu tür çizgelere **ağırlıklı/maliyetli çizge (Weighted Graphs)** denir. Ağırlık uygulamadan uygulamaya deęişir.

- Şehirler arasındaki uzaklık
- Routerler arası bant genişliği

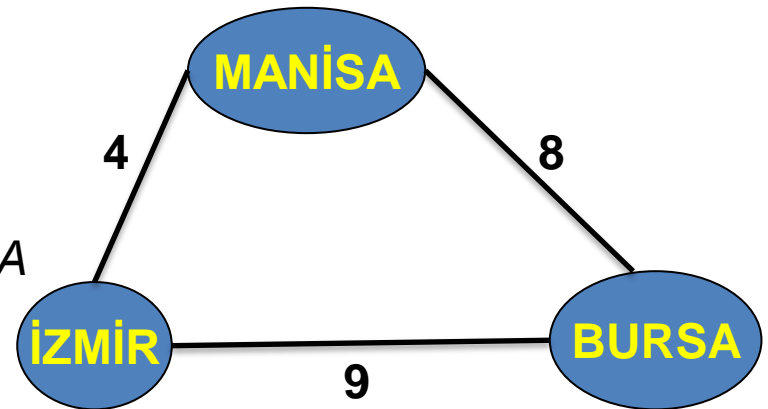


Yol (Path)

- Bir yol (**path**) düğümlerin bir sırasıdır ($v_0, v_1, v_2, \dots, v_k$) öyle ki:
 - $0 \leq i < k$ için, $\{v_i, v_{i+1}\}$ bir kenardır
 - $0 \leq i < k-1$ için, $v_i \neq v_{i+2}$
Yani, kenarlar $\{v_i, v_{i+1}\} \neq \{v_{i+1}, v_{i+2}\}$
- **Basit Yol (Simple Path):** Tüm düğümlerin farklı olduğu yoldur.
- **Daire (Cycle):** Başlangıç ve bitiş düğümleri aynı olan basit yol.
- **Uzunluk:** Bir yol üzerindeki kenarların sayısının toplamıdır.

Basit Yol: MANİSA, İZMİR

Daire: MANİSA, İZMİR, BURSA, MANİSA



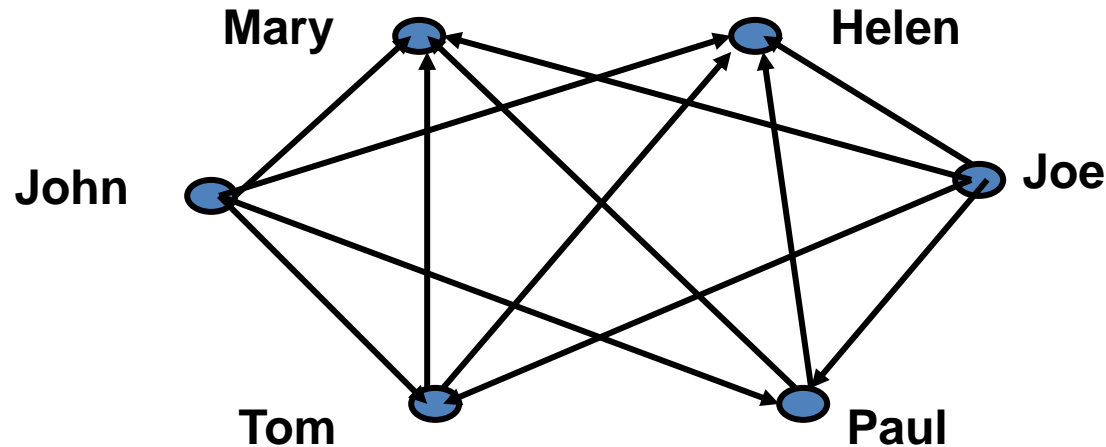
Çizge Kullanım Alanları

- Bilgisayar ağlarında, elektriksel ve diğer ağların analizinde,
- Kimyasal bileşiklerin moleküler yapılarının araştırılmasında,
- Ulaşım ağlarında (kara, deniz ve havayolları),
- Planlama projelerinde,
- Sosyal alanlarda ve diğer pek çok alanda kullanılmaktadır.

Çizge Kullanım Alanları (devam...)

Örnek (Generic)

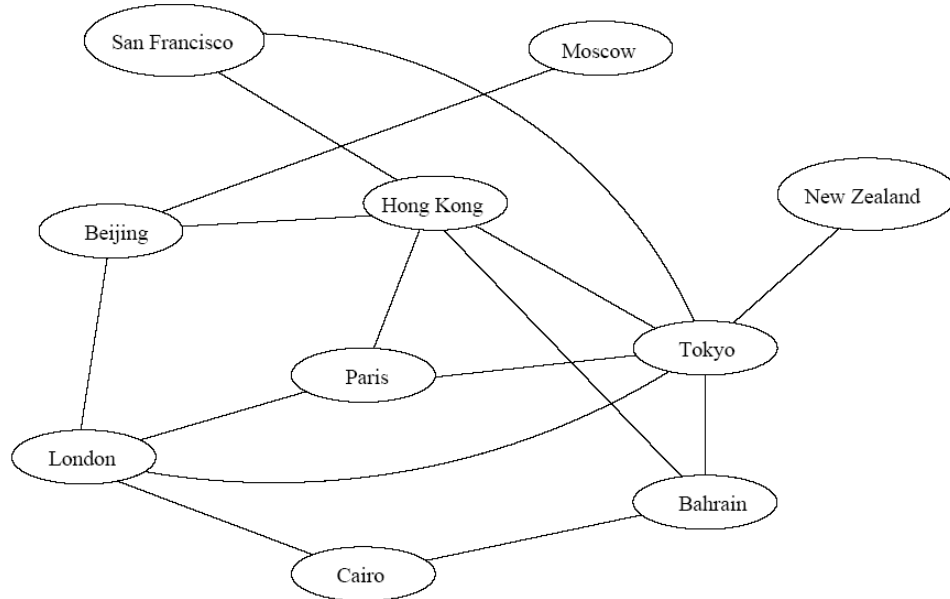
- $V = 6$ kişi: John, Mary, Joe, Helen, Tom ve Paul, sırasıyla yaşları: 12, 15, 12, 15, 13 ve 13.
- $E = \{(x, y) \mid \text{Eğer } x, y \text{ den daha genç ise}\}$



Çizge Kullanım Alanları (devam...)

Örnek (Uçuş sistemi)

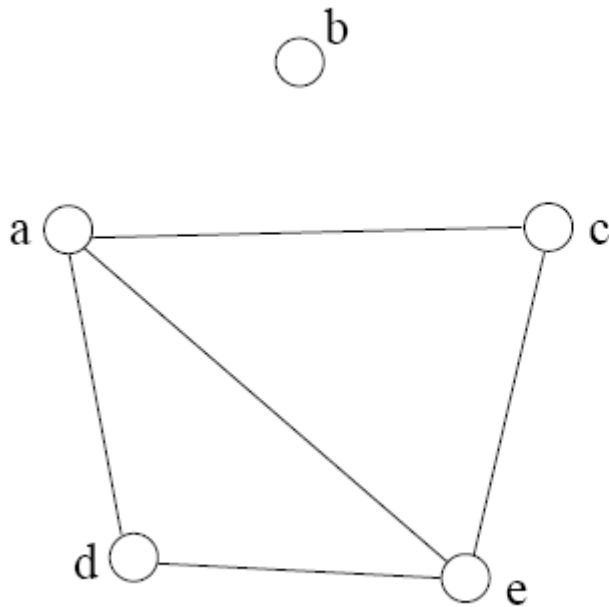
- Her bir düğüm bir şehri gösterir
- Her bir kenar iki şehir arasındaki doğrudan uçuşu gösterir
- Doğrudan uçuşların sorgulanmasında cevap bir kenardır.
- Bir yere ulaşmak için “A’ dan B’ ye yol varmı” sorusu sorulur.
- Maliyetleri kenarlara bile ekleyebiliriz. (ağırlıklı), daha sonra “A’ dan B’ ye en ucuz yol hangisidir?” diye sorabiliriz.



Çizge Gösterimi

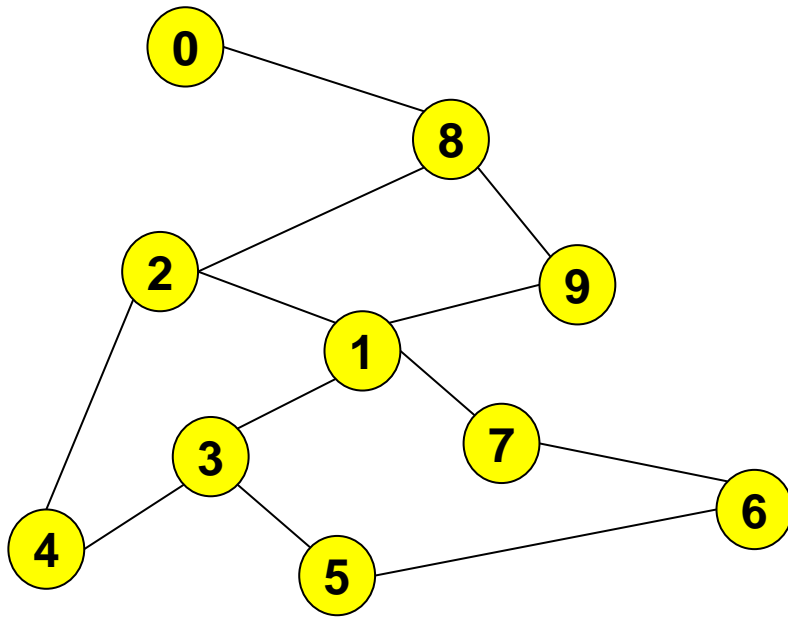
- İki popüler gösterim bulunmaktadır. Her ikisi de farklı yönlerden düğüm ve kenar kümelerini gösterir.
- 1. Komşuluk Matrisi:** Çizgeyi göstermek için D matrisi kullanılır.
 - 2. Komşuluk Listesi:** Bağlantılı listelerin bir boyutlu dizisi kullanılır.

Komşuluk Matrisi



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

Komşuluk Matrisi (devam...)

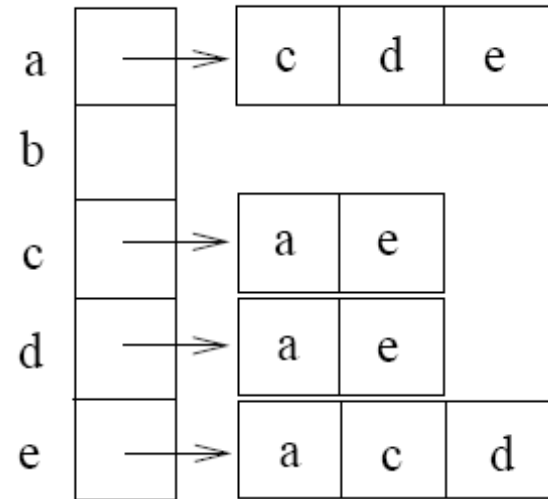
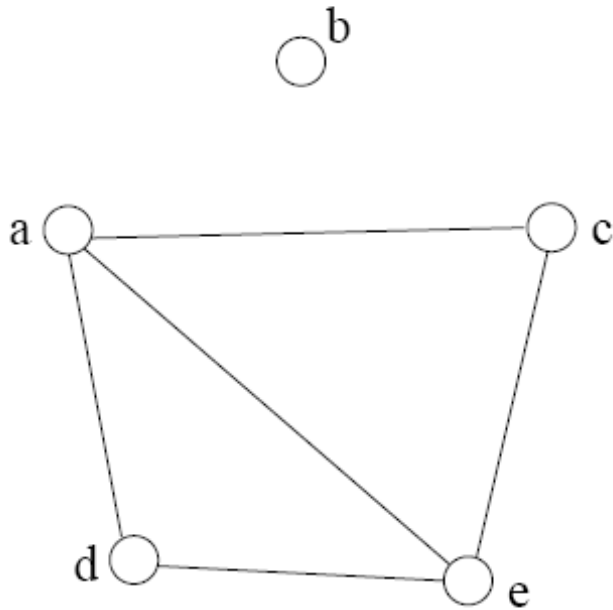


	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	1	0	0	0	1	0	1
2	0	1	0	0	1	0	0	0	1	0
3	0	1	0	0	1	1	0	0	0	0
4	0	0	1	1	0	0	0	0	0	0
5	0	0	0	1	0	0	1	0	0	0
6	0	0	0	0	0	1	0	1	0	0
7	0	1	0	0	0	0	1	0	0	0
8	1	0	1	0	0	0	0	0	0	1
9	0	1	0	0	0	0	0	0	1	0

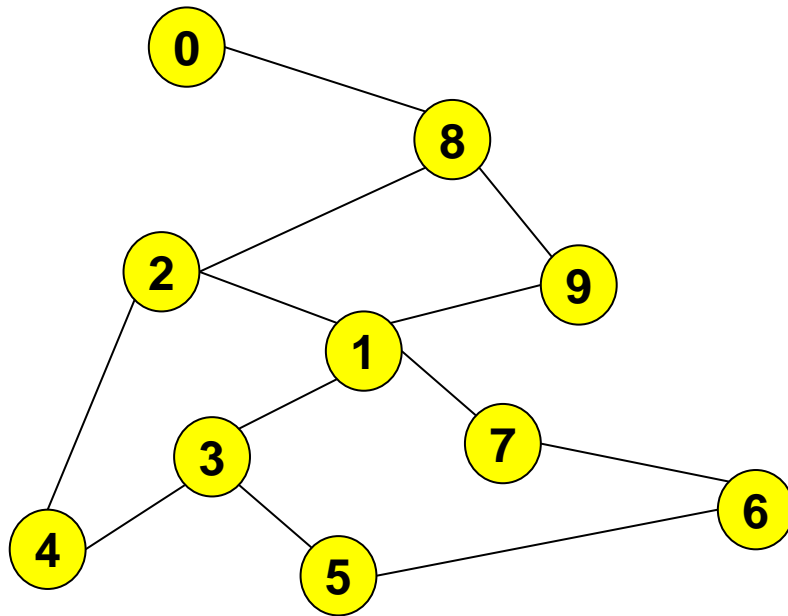
Komşuluk Matrisi (devam...)

- İki boyutlu dizi ile gerçekleştirilebilir.
- Uygulaması basittir.
 - Kenar yaratmak ve kaldırmak kolaydır.
- Hafızada fazla yer kaplar.
- Daha az hafızaya ihtiyaç duyulması için sparse matris tekniklerinin kullanılması gerekir.

Komşuluk Listesi



Komşuluk Listesi (devam...)

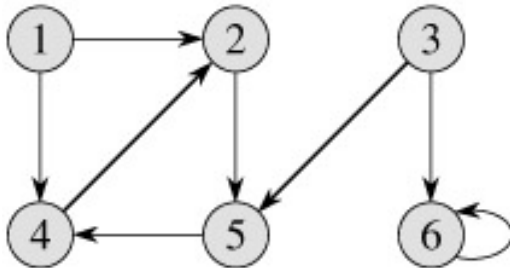


0	→	8
1	→	2 3 7 9
2	→	1 4 8
3	→	1 4 5
4	→	2 3
5	→	3 6
6	→	5 7
7	→	1 6
8	→	0 2 9
9	→	1 8

Komşuluk Listesi (devam...)

- Bağlı liste içeren dizi ile gerçekleştirimi yapılır.
- Uygulaması daha karmaşıktır.
- Hafıza kullanımını komşuluk matrisine göre daha optimaldir.

Örnek – Yönlü Çizge



Komşuluk Listesi

1	→	2	→	4	/
2	→	5	/		
3	→	6	→	5	/
4	→	2	/		
5	→	4	/		
6	→	6	/		

Komşuluk Matrisi

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Çizge Üzerinde Gezinme (Traverse)

- Çizge üzerinde dolaşma; **çizge düğümleri** ve **kenarları** üzerinde **istenen bir işi yapacak** veya bir *problemi çözecek* biçimde hareket etmektir.
- Çizge üzerinde dolaşma yapan **birçok yaklaşım ve yöntem** bulunmaktadır. En önemli iki tanesi aşağıdaki gibidir:
 - **BFS (Breadth First Search) Yöntemi**
 - **DFS (Depth First Search) Yöntemi**

Çizge Üzerinde Gezinme (devam...)

Depth-First Search (DFS)

- Bir düğümden **başla**, düğümün bir kenarında o kenar üzerinde gidilebilecek **en uzak düğüme kadar** sürdür.
- Geri gel (backtracking) ve diğer kenarı dene.
- Tüm düğümler **gezilene kadar devam et.**
- Genelde Stack kullanarak gerçekleştirim yapılır.

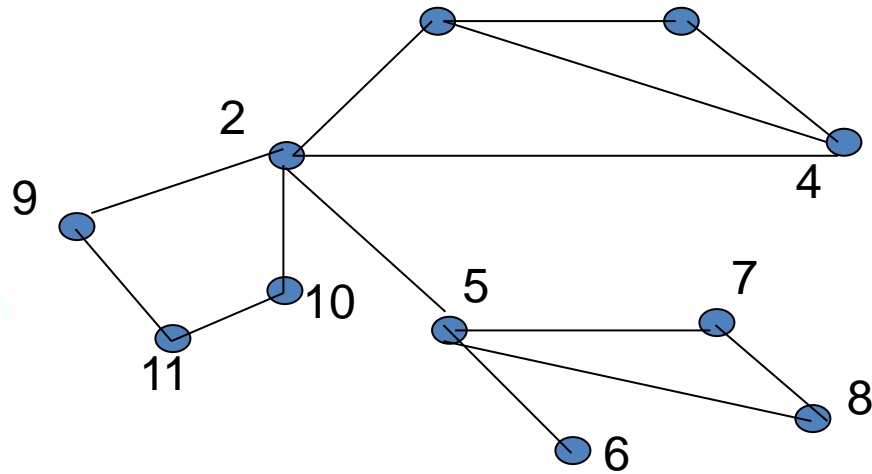
Breadth-First Search (BFS)

- Başlangıç düğümünden **başla** ve **tüm komşuları ziyaret et.**
- Daha sonra komşunun **komşularını ziyaret et.**
- Başlangıç düğümünden başlayıp **dışa doğru dalga gibi** ilerle.
- Genelde Queue kullanarak gerçekleştirim yapılır.

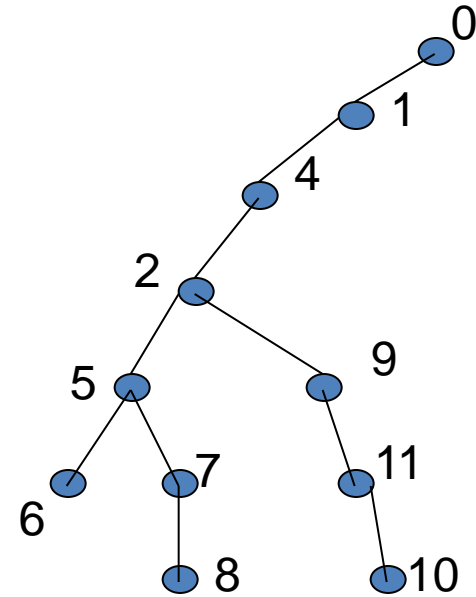
DFS Söзде Kod

```
DFS (Graph G) {  
    Stack S; Integer x, t;  
    while (G_ziyaret_edilmeyen_x_düğümüne_sahipse) {  
        visit(x);  
        push(x, S);  
        while (S boş değilse) {  
            t := peek(S);  
            if(t_ziyaret_edilmeyen_y_komşusuna_sahipse)  
            {  
                visit(y);  
                push(y, S);  
            }  
            else  
                pop(S);  
        }  
    }  
}
```

DFS Gösterim



Çizge

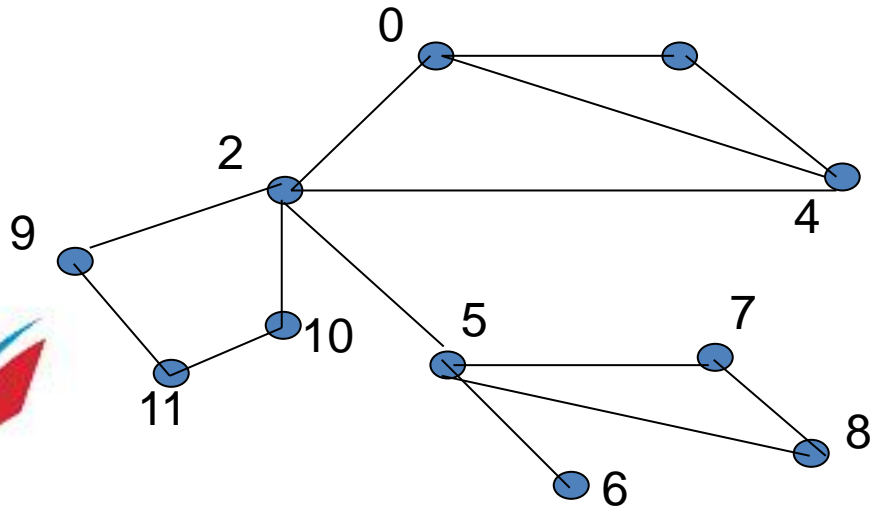


Gezinme Ağacı

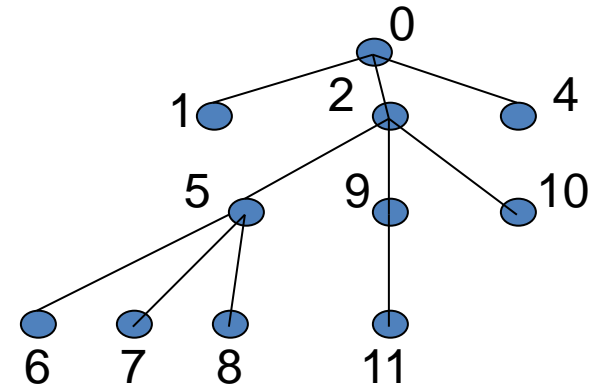
BFS Sözde Kod

```
BFS (Graph G) {  
    Queue Q;  
    Integer x, z, y;  
    while (G_ziyaret_edilmeyen_x_düğümüne_sahipse)  
    {  
        visit(x);  
        Insert(x,Q);  
        while (Q boş değilse) {  
            z := Remove(Q);  
            for all (z nin tüm y komşuları için) {  
                visit(y);  
                Insert(y,Q);  
            }  
        }  
    }  
}
```

BFS Gösterim



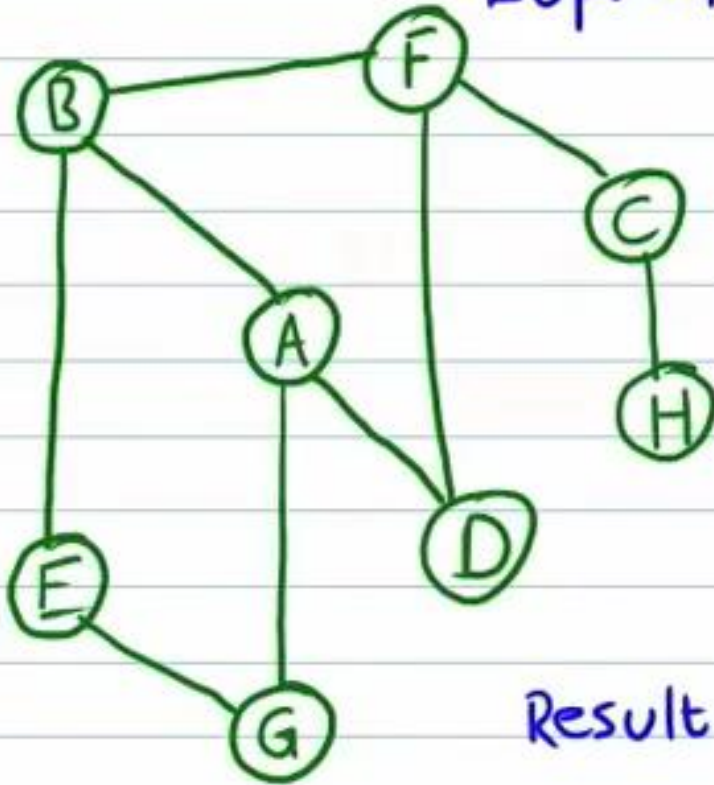
Çizge



Gezinme Ağacı

Çizge Üzerinde Gezinme (devam...)

Graph Algorithms:
Depth-First Search



Result:

İYİ ÇALIŞMALAR...

Yararlanılan Kaynaklar

- **Ders Kitabı:**
 - **Data Structures through JAVA**, V.V.Muniswamy
- **Yardımcı Okumalar:**
 - Data Structures and Algorithms in Java, Narashima Karumanchi
 - Data Structures, Algorithms and Applications in Java, Sartaj Sahni
 - Algorithms, Robert Sedgewick