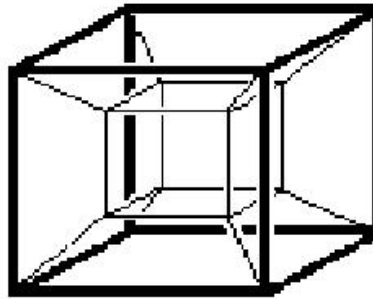
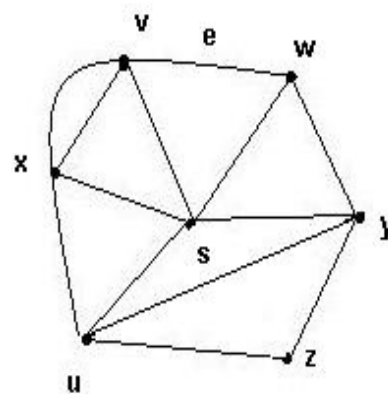


# Graf Teorisi (Graph Theory)



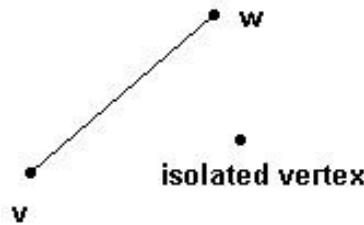
## Giriş

- G grafi nedir ?
- $G = (V, E)$ 
  - $V = V(G)$  = düğümler kümesi
  - $E = E(G)$  = kenarlar kümesi
- **Örnek:**
  - $V = \{s, u, v, w, x, y, z\}$
  - $E = \{(x,s), (x,v)_1, (x,v)_2, (x,u), (v,w), (s,v), (s,u), (s,w), (s,y), (w,y), (u,y), (u,z), (y,z)\}$



## Kenarlar (Edges)

- Kenar bir çift düğüm ile etiketlenmiş olup  $e = (v,w)$  şeklinde gösterilir.
- Ayrık düğüm (Isolated vertex) = a kenar bağlantısı olmayan düğümdür.



## Özel Kenarlar

- Paralel kenarlar(Parallel edges)

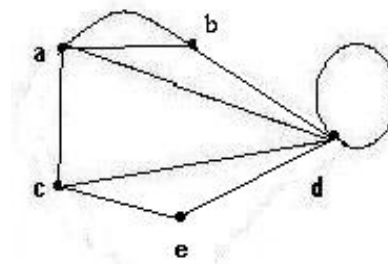
- İki veya daha fazla kenar bir düğüm çifti ile bağlanmıştır.

- a ve b iki paralel kenar ile birleşmiştir

- Döngüler (Loops)

- Kenarın başlangıç ve bitiş noktası aynı düğümdür.

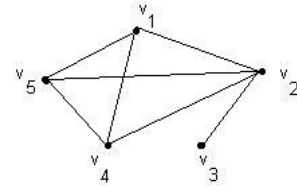
- d düğümü gibi.



## Özel Graflar

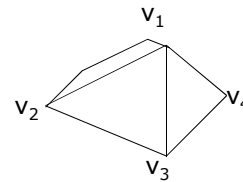
### ■ Basit (Simple) Graf

- Yönsüz, paralel kenar olmayan ve döngü içermeyen graflardır.



### ■ Çoklu (Multi) Graf

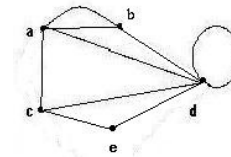
- Basit grafların yeterli olmadığı durumlarda kullanılır.
- Yönsüz, paralel kenarı olan ve döngü içermeyen graflardır.



Basit graflar, çoklu graftır fakat çoklu graflar basit graf değildir.

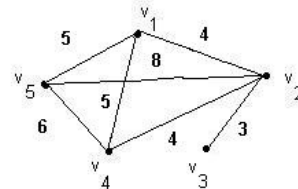
### ■ Pseudo Graflar

- Çoklu grafların yeterli olmadığı durumlarda kullanılır.
- Yönsüz, Paralel kenarı olan ve döngü içeren graflardır.
- Yönsüz grafların en temel halidir.



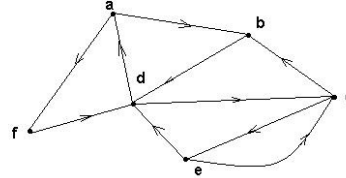
### ■ Ağırlıklı (Weighted) Graf

Her bir kenarına nümerik bir değer, ağırlık verilmiş bir grafıdır.



## Yönlü (Directed) Graflar (digraphs)

**G**, yönlü bir graf  
(*directed*) veya *digraph*  
ise her bir kenarı sıralı  
bir düğüm çifti ile  
ilişkilendirilmiş ve her  
kenarı yönlüdür.



Tip	Kenar	Çoklu Kenara İzin ?	Döngüye İzin ?
Basit Graf	Yönsüz	Hayır	Hayır
Çoklu Graf	Yönsüz	Evet	Hayır
Pseudo Graf	Yönsüz	Evet	Evet
Yönlü Graf	Yönlü	Hayır	Evet
Yönlü Çoklu Graf	Yönlü	Evet	Evet

## Graflarda Benzerlik (similarity) (1)

**Problem:** Nesnelerin değişik özellikleri referans alınarak nesneleri sınıflandırabiliriz.

**Örnek:**

- Bilgisayar programlarında üç ayrı özelliğin olduğunu kabul edelim.  $k = 1, 2, 3$  gibi:
- 1. Programın satır sayısı
- 2. Kullanılan "return" sayısı
- 3. Çağrılan fonksiyon sayısı

## Graflarda benzerlik (2)

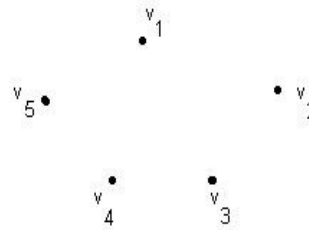
Aşağıdaki tabloda 5 programın birbirleriyle karşılaştırıldığını farzedelim.

Program	# of lines	# of "return"	# of function calls
1	66	20	1
2	41	10	2
3	68	5	8
4	90	34	5
5	75	12	14

## Graflarda benzerlik (3)

□ G grafinı aşağıdaki gibi oluşturun:

- $V(G)$  programlardan oluşan bir küme  $\{v_1, v_2, v_3, v_4, v_5\}$ .
- Her düğüm,  $v_i$  bir üçlü ile gösterilir  $(p_1, p_2, p_3)$ ,
- burada  $p_k$  özellik değerleridir  $k = 1, 2$ , veya  $3$
- $v_1 = (66, 20, 1)$
- $v_2 = (41, 10, 2)$
- $v_3 = (68, 5, 8)$
- $v_4 = (90, 34, 5)$
- $v_5 = (75, 12, 14)$



## Benzer olmayan fonksiyonlar (1)

- Benzer olmayan (*dissimilarity function*) bir fonksiyon aşağıdaki gibi tanımlanır.
- Her bir düğüm çifti  $v = (p_1, p_2, p_3)$  ve  $w = (q_1, q_2, q_3)$  ile gösterilsin.

$$s(v, w) = \sum_{k=1}^3 |p_k - q_k|$$

- $v$  ve  $w$  gibi iki programın *dissimilarity*  $s(v, w)$  ile ölçülür.
- $N$  seçilen sabit bir sayı olsun. Eğer  $s(v, w) < N$  ise  $v$  ve  $w$  arasındaki kenar eklenir. Sonra:
- Eğer  $v = w$  veya  $v$  ve  $w$  arasında bir yol varsa  $v$  ve  $w$  nun aynı sınıfta olduğunu söyleyebiliriz.

## Benzer olmayan fonksiyonlar(2)

□  $N = 25$  (denemeler ile belirleniyor)

$$\begin{array}{lll}
 s(v_1, v_2) = 36 & s(v_2, v_3) = 38 & s(v_3, v_4) = 54 \\
 s(v_1, v_3) = 24 & s(v_2, v_4) = 76 & s(v_3, v_5) = 20 \\
 s(v_1, v_4) = 42 & s(v_2, v_5) = 48 & s(v_4, v_5) = 46 \\
 s(v_1, v_5) = 30
 \end{array}$$

## Benzer olmayan fonksiyonlar(2)

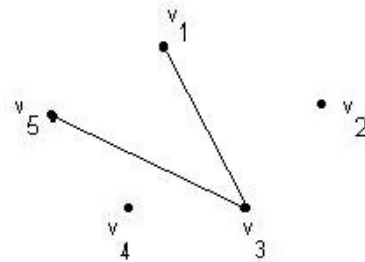
□  $N = 25$ .

□  $s(v_1, v_3) = 24$ ,  $s(v_3, v_5) = 20$   
ve diğerleri  $s(v_i, v_j) > 25$

□ Üç sınıf vardır:

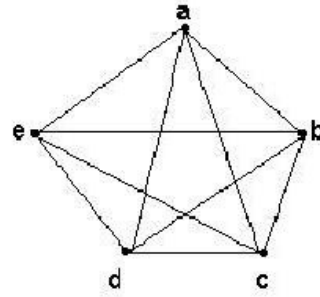
□  $\{v_1, v_3, v_5\}$ ,  $\{v_2\}$  and  $\{v_4\}$

□ **similarity graph** şekildeki gibidir.



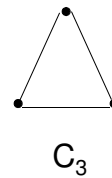
## Tam (Complete) Graf $K_n$

- $n \geq 3$
- *complete graph*  $K_n$  :  $n$  adet düğüm içeren basit graf yapısındadır. Her düğüm, diğer düğümlere bir kenar ile bağlantılıdır.
- Şekilde  $K_5$  grafi gösterilmiştir.
- Soru:  $K_3$ ,  $K_4$ ,  $K_6$  graflarını çiziniz.



## Cycles (Çember) Graf $C_n$

- $n \geq 3$
- *cycles graph*  $C_n$  :  $n$  adet düğüm ve  $\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{n-1}, v_n\}, \{v_n, v_1\}$ , düğüm çiftlerinden oluşan kenarlardan meydana gelir.
- Şekilde  $C_3$  grafi gösterilmiştir.
- Soru:  $C_4$ ,  $C_5$ ,  $C_6$  graflarını çiziniz.





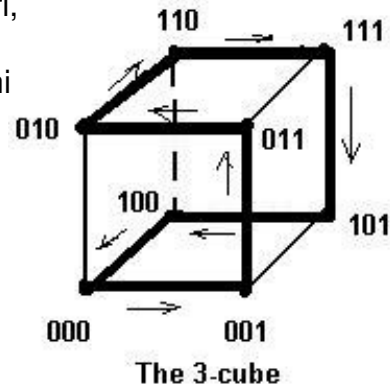
## Wheel (Tekerlek) Graf $W_n$

- *wheel graph*  $W_n$  : Cycle  $C_n$  grafına ek bir düğüm eklenerek oluşturulur. Eklenen yeni düğüm, diğer bütün düğümlere bağlıdır.
- Şekilde  $W_3$  grafi gösterilmiştir.
- Soru:  $W_4, W_5, W_6$  graflarını çiziniz.

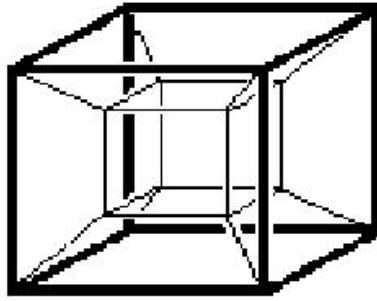


## N-Cube (Küp) Graf $Q_n$

- *N-cube*  $Q_n$  : Grafın düğüm noktaları  $n$  uzunluğunda  $2^n$  bit stringi ile gösterilir. Düğümlerin string değeri, bir düğümden diğerine geçerken aynı anda sadece bir bitin değerini değiştirmektedir.
- (000, 001, 011, 010, 110, 111, 101, 100, 000)
- Şekilde  $Q_3$  grafi gösterilmiştir.
- Soru:  $Q_1, Q_2$  graflarını çiziniz.



## hypercube veya 4-cube



16 düğüm, 32 kenar ve 20 yüzey

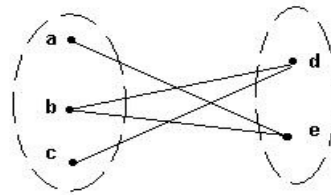
■ Düğüm etiketleri:

0000 0001 0010 0011  
0100 0101 0110 0111  
1000 1001 1010 1011  
1100 1101 1110 1111

## İki Parçalı (Bipartite) Graflar

■  $G$ , bipartite graf ise:

- $V(G) = V(G_1) \cup V(G_2)$
- $|V(G_1)| = m, |V(G_2)| = n$
- $V(G_1) \cap V(G_2) = \emptyset$



- Bir grafi oluşturan düğümleri iki ayrı kümeye bölerek grafi ikiye ayırabiliriz. Bu ayırma işleminde izlenecek yol; bir kenar ile birbirine bağlanabilecek durumda olan düğümleri aynı küme içerisine yerleştirmemektir.
- Mevcut küme içerisindeki düğümler birbirlerine herhangi bir kenar ile bağlanmamalıdır.

- $K_3$  Bipartite graf mıdır ?

Hayır

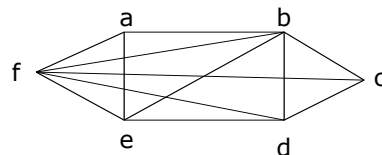
- $C_6$  Bipartite graf mıdır?

Evet

$\{1,3,5\}$  ve  $\{2,4,6\}$

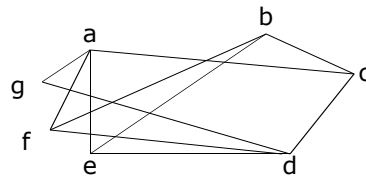
Yandaki graf Bipartite graf mıdır?

Hayır

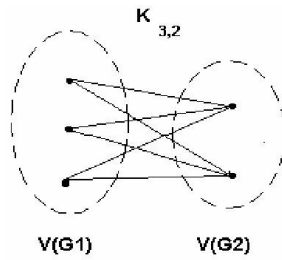


Yandaki graf Bipartite graf mıdır?

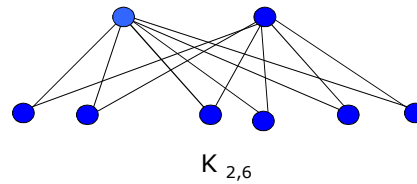
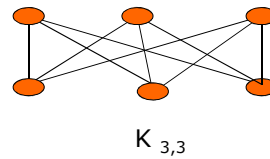
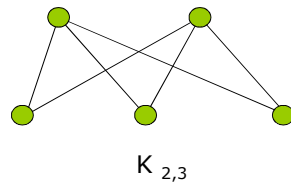
Evet.  $\{a,b,d\}$  ve  $\{c,e,f,g\}$



## Tam (complete) bipartite graph $K_{m,n}$



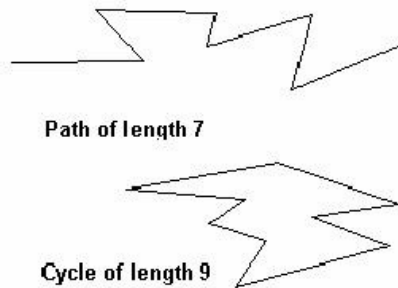
- *complete* bipartite graf  $K_{m,n}$  şeklinde gösterilir. İlgili grafın düğümlerinin kümesi  $m$  ve  $n$  elemanlı iki alt kümeye ayrılır.
- Bir kenarı birbirine bağlayan iki düğümünde farklı alt kümelerin elemanı olmak zorundadırlar.
- $|V(G_1)| = m$
- $|V(G_2)| = n$



$K_n$ ,  $C_n$ ,  $W_n$ ,  $K_{m,n}$ ,  $Q_n$  graflarının kenar ve düğüm sayılarını formüle edecek olursak:

$K_n$	$n$ düğüm	$n(n-1)/2$ kenar
$C_n$	$n$ düğüm	$n$ kenar
$W_n$	$n+1$ düğüm	$2n$ kenar
$K_{m,n}$	$m+n$ düğüm	$m*n$ kenar
$Q_n$	$2^n$ düğüm	$n2^{n-1}$ kenar

## Yollar (Paths) ve Döngüler(Cycles)



- $n$  uzunluğundaki bir yol'un (path)  $n+1$  adet düğümü ve  $n$  adet de ardışık kenarı vardır
- Bir döngü içeren yol başladığı düğümde son bulur. Uzunluğu  $n$  olan bir döngüde  $n$  adet düğüm vardır.

## Euler Döngüsü (Euler cycles)

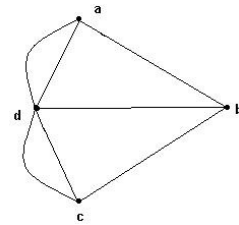
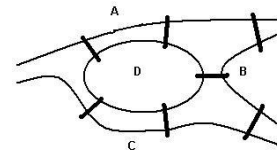
□  $G$  grafi içerisindeki *Euler cycle* basit bir çevrim olup  $G$  grafi içerisindeki her kenardan sadece bir kez geçilmesine izin verir.

□ Königsberg köprü problemi:

□ Başlangıç ve Bitiş noktası aynıdır, yedi köprüden sadece bir kez geçerek başlangıç noktasına dönmek mümkün müdür?

□ Bu problemi grafa indirgeyelim.

□ Kenarlar köprüleri ve düğüm noktaları da bölgeleri gösterebilir.

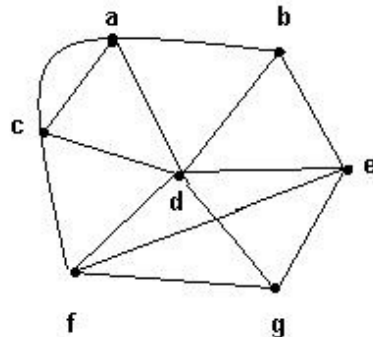


## Bir düğümün derecesi

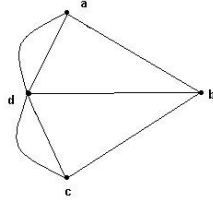
□  $v$  düğümünün derecesi  $\delta(v)$  ile gösterilir ve bu da yönsüz bir grafa düğüme gelen kenarlar toplamıdır. Düğüm noktalarındaki döngü düğüm derecesine 2 kez katılır.

□ Örnek:

- $\delta(a) = 4, \delta(b) = 3,$
- $\delta(c) = 4, \delta(d) = 6,$
- $\delta(e) = 4, \delta(f) = 4,$
- $\delta(g) = 3.$



## Euler Grafi



- Bir **G** grafi *Euler cycle*'ına sahip ise *Euler Grafi* adını alır.
- Euler grafında tüm düğümlerin derecesi çifttir.
- Königsberg bridge problemi bir Euler grafi değildir.
- Königsberg bridge probleminin çözümü yoktur.

## Grafın düğüm derecelerinin toplamı

- Sıfır dereceli bir düğüm **isolated** olarak adlandırılır. Isolated olan bir düğümden, başka bir düğüme yol yoktur.
- Düğüm derecesi bir olan düğüme **pendant** denir.
- **Teorem: Handshaking**  
**e** adet kenarlı ve **n** adet düğümlü bir grafın  $G(V,E)$  düğümlerinin dereceleri toplamı kenar sayısının iki katıdır.

$$\sum_{i=1}^n \delta(v_i) = 2e$$

**Örnek:** Her birinin derecesi 6 olan 10 düğümlü bir grafın kaç tane kenarı vardır.

$$e=30$$

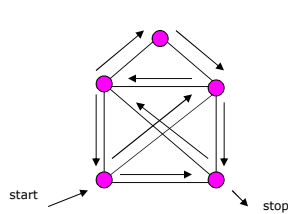
□ G grafında  $(v,w)$  yönlü bir kenar olsun ve yön  $v$ 'den  $w$ 'ya verilsin.  $v$  **initial vertex**,  $w$ 'da **terminal** veya **end vertex** olarak adlandırılır. Bir düğüm noktasında döngü söz konusu ise bu düğümün *initial vertex*'i ve *end vertex*'i birbirinin aynıdır.

□ Yönlü bir grafta, herhangi bir düğümün **in\_degree**'si  $\delta^-(v)$ , **out\_degree**'si  $\delta^+(v)$  olarak gösterilir.

□ Yönlü bir grafin in\_degree ve out\_degree'lerinin toplamı birbirinin aynıdır.

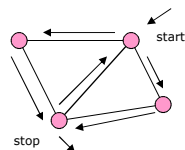
$$\sum_{v \in V} \delta^-(v) = \sum_{w \in V} \delta^+(w)$$

**Örnek:** Aşağıda verilmiş olan graflardan hangilerinde her kenardan en az bir kez geçirilerek graf gezilmiştir, hangileri Euler grafıdır, eğer değilse sebebi nedir ?



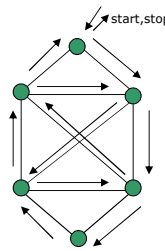
Path var, Euler grafi değil

Düğüm dereceleri çift değil

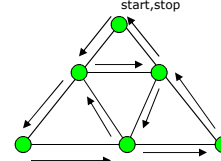


Path var, Euler grafi değil

Düğüm dereceleri çift değil



Euler grafi

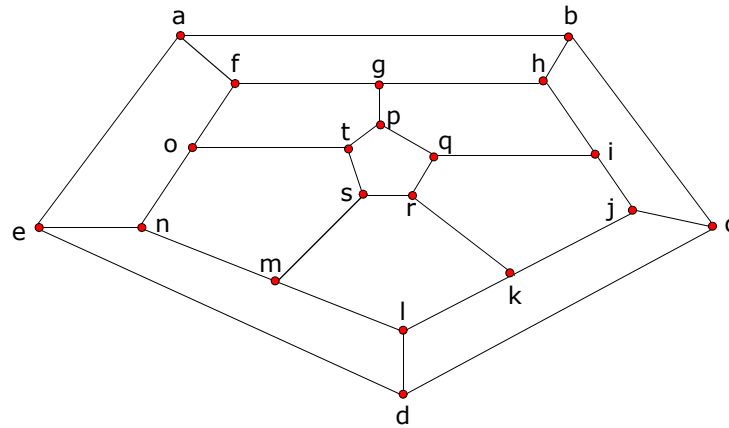
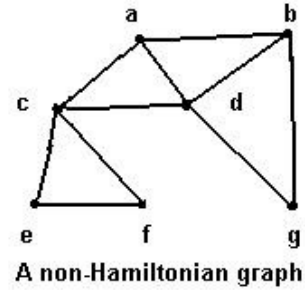


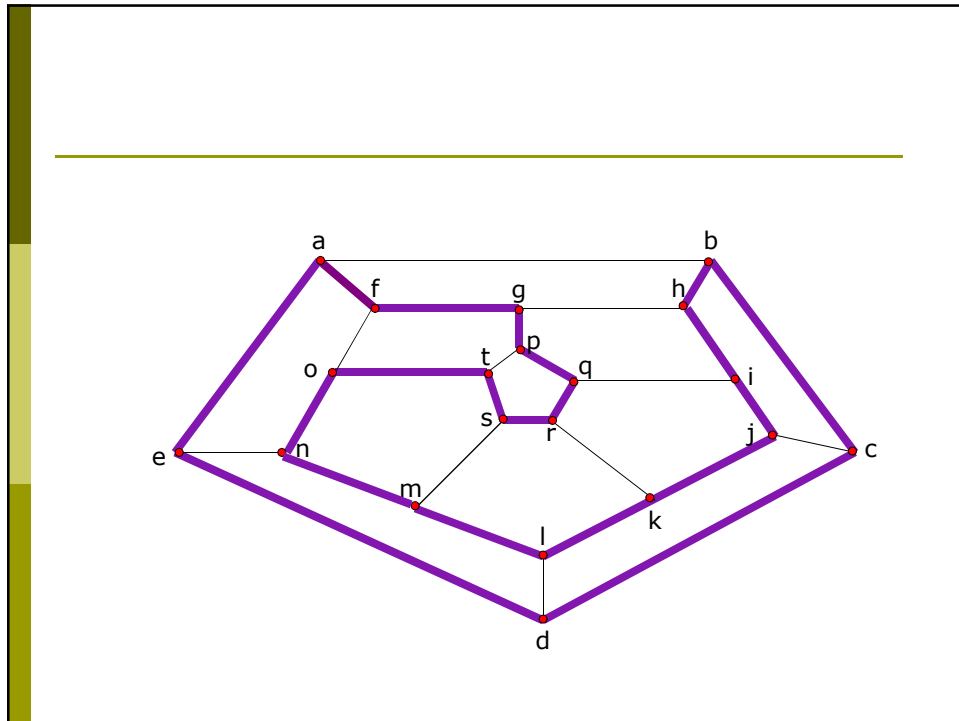
Euler grafi



## Hamilton Döngüsü (Hamiltonian Cycles)

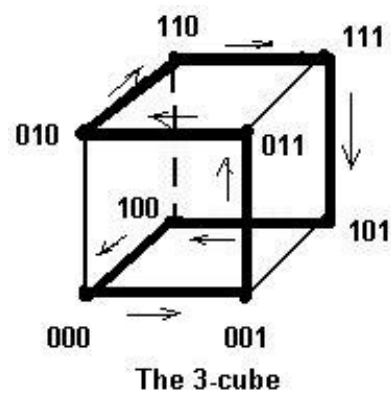
- G grafinın üzerindeki her düğümden yalnız bir kez geçmek şartı ile kapalı bir yol oluşturabilen graflardır (*Traveling salesperson*)
- Bu kapalı yol *Hamiltonian cycle* olarak adlandırılır.
- *Hamiltonian cycle* sahip bir G grafi *Hamiltonian* graf olarak adlandırılır.





## 3-cube

Hamiltonian cycle  
 (000, 001, 011, 010,  
 110, 111, 101, 100,  
 000) örnek bir graf  
 3-cube olarak  
 verilebilir.



## EN KISA YOL (SHORTEST PATH) ALGORİTMASI

### Dijkstra's Algorithm

#### Dijkstra's Algorithm

Dijkstra's algorithm is known to be a good algorithm to find a shortest path.

1. Set  $i=0$ ,  $S_0 = \{u_0=s\}$ ,  $L(u_0)=0$ , and  $L(v)=\text{infinity}$  for  $v \neq u_0$ .  
If  $|V| = 1$  then stop, otherwise go to step 2.
2. For each  $v$  in  $V \setminus S_i$ , replace  $L(v)$  by  $\min\{L(v), L(u_i) + d_{u_i v}^u\}$ .  
If  $L(v)$  is replaced, put a label  $(L(v), u_i)$  on  $v$ .
3. Find a vertex  $v$  which minimizes  $\{L(v) : v \in V \setminus S_i\}$ , say  $u_{i+1}$ .
4. Let  $S_{i+1} = S_i \cup \{u_{i+1}\}$ .
5. Replace  $i$  by  $i+1$ . If  $i=|V|-1$  then stop, otherwise go to step 2.

The time required by Dijkstra's algorithm is  $O(|V|^2)$ .

It will be reduced to  $O(|E|\log|V|)$  if heap is used to keep  $\{v \in V \setminus S_i : L(v) < \text{infinity}\}$ .

Java Applet Demos of Dijkstra's Algorithm - Microsoft Internet Explorer

Address: <http://www.b2.is.tokushima-u.ac.jp/~ikeeda/suuri/dijkstra/DijkstraApp.shtml#demo1>

Google: shortest path

Mathematical Programming

Simplex

Twophase

Dijkstra

Prim

Kruskal

Ford-Fulkerson

Dijkstra

Java applet demos:

- demo1
- demo2
- demo3
- demo4
- demo5
- demo6
- demo7
- demo8
- demo9
- demo10

FAQ

Java Applet Demos of Dijkstra's Algorithm

CLICK on the java applet below for several times to find a shortest path from  $u_0$ .

A (6,10) graph

Start | Banu DIRI - Micr... | shortest path - ... | Java Applet D... | Ch-6 | Microsoft Power... | Internet | 17:5

Java Applet Demos of Dijkstra's Algorithm - Microsoft Internet Explorer

Address: <http://www-b2.is.tokushima-u.ac.jp/~ikedai/suun/dijkstra/DijkstraApp.shtml?demo1>

Google shortest path

**Mathematical Programming**

Simplex

Twophase

Dijkstra

Prim

Kruskal

Ford-Fulkerson

Dijkstra

Java applet demos:

- demo1
- demo2
- demo3
- demo4
- demo5
- demo6
- demo7
- demo8
- demo9
- demo10

FAO

**Java Applet Demos of Dijkstra's Algorithm**

CLICK on the java applet below for several times to find a shortest path from  $u_0$ .

A (6,10) graph

Java Applet Demos of Dijkstra's Algorithm - Microsoft Internet Explorer

Address: <http://www-b2.is.tokushima-u.ac.jp/~ikedai/suun/dijkstra/DijkstraApp.shtml?demo1>

Google shortest path

**Mathematical Programming**

Simplex

Twophase

Dijkstra

Prim

Kruskal

Ford-Fulkerson

Dijkstra

Java applet demos:

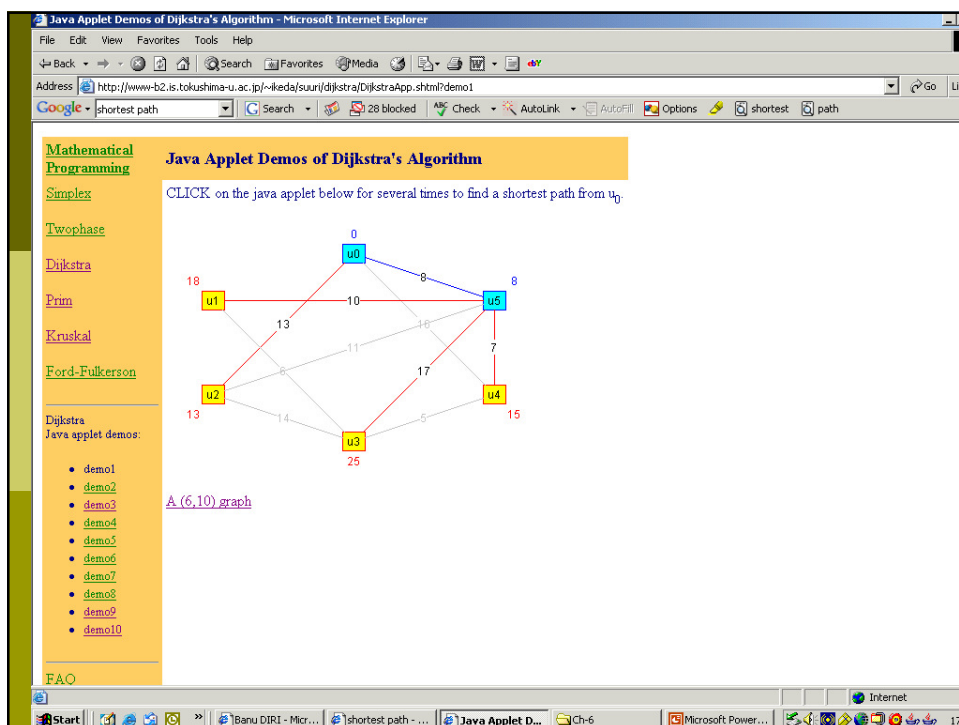
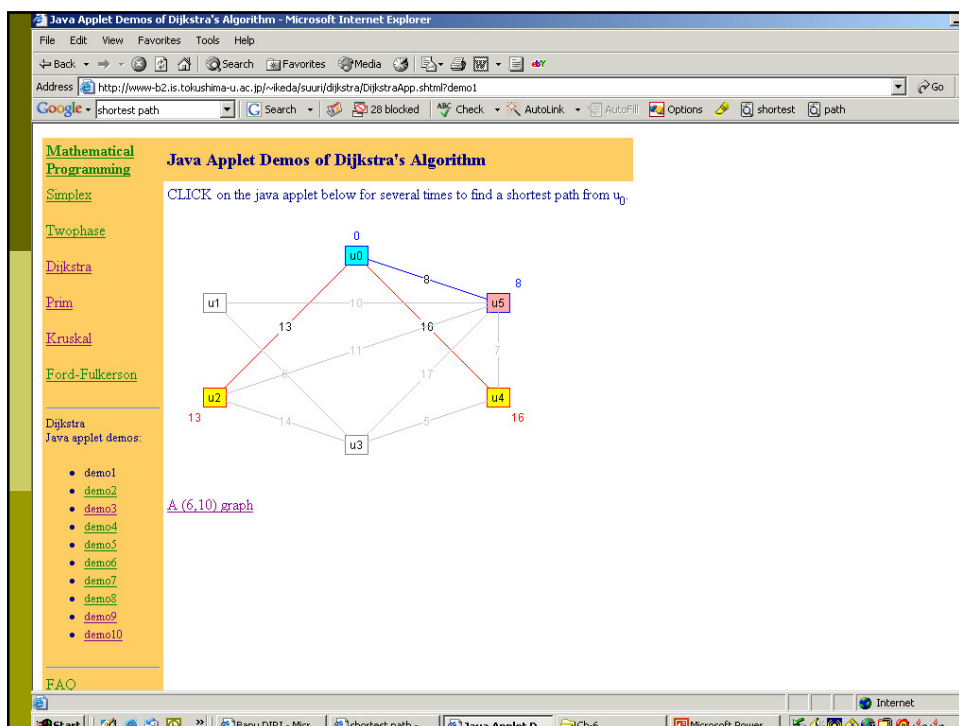
- demo1
- demo2
- demo3
- demo4
- demo5
- demo6
- demo7
- demo8
- demo9
- demo10

FAO

**Java Applet Demos of Dijkstra's Algorithm**

CLICK on the java applet below for several times to find a shortest path from  $u_0$ .

A (6,10) graph



Java Applet Demos of Dijkstra's Algorithm - Microsoft Internet Explorer

Address: <http://www.b2.is.tokushima-u.ac.jp/~ikedai/suuri/dijkstra/DijkstraApp.shtml?demo1>

Google shortest path

**Mathematical Programming**

Simplex

Twophase

Dijkstra

Prim

Kruskal

Ford-Fulkerson

Dijkstra

Java applet demos:

- demo1
- demo2
- demo3
- demo4
- demo5
- demo6
- demo7
- demo8
- demo9
- demo10

FAO

**Java Applet Demos of Dijkstra's Algorithm**

CLICK on the java applet below for several times to find a shortest path from  $u_0$ .

A (6,10) graph

Java Applet Demos of Dijkstra's Algorithm - Microsoft Internet Explorer

Address: <http://www.b2.is.tokushima-u.ac.jp/~ikedai/suuri/dijkstra/DijkstraApp.shtml?demo1>

Google shortest path

**Mathematical Programming**

Simplex

Twophase

Dijkstra

Prim

Kruskal

Ford-Fulkerson

Dijkstra

Java applet demos:

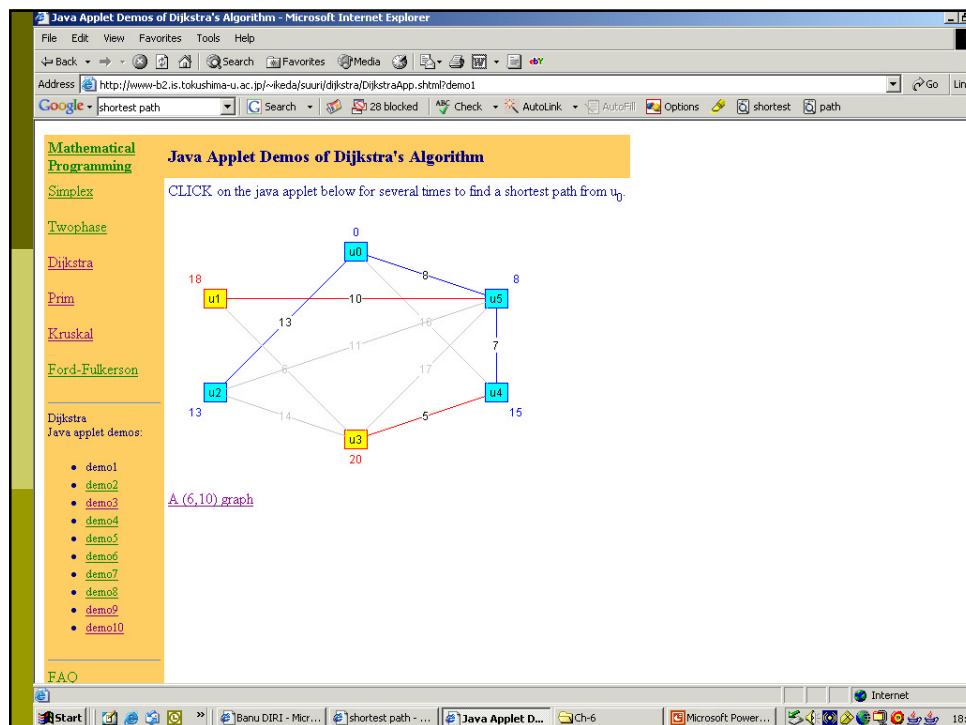
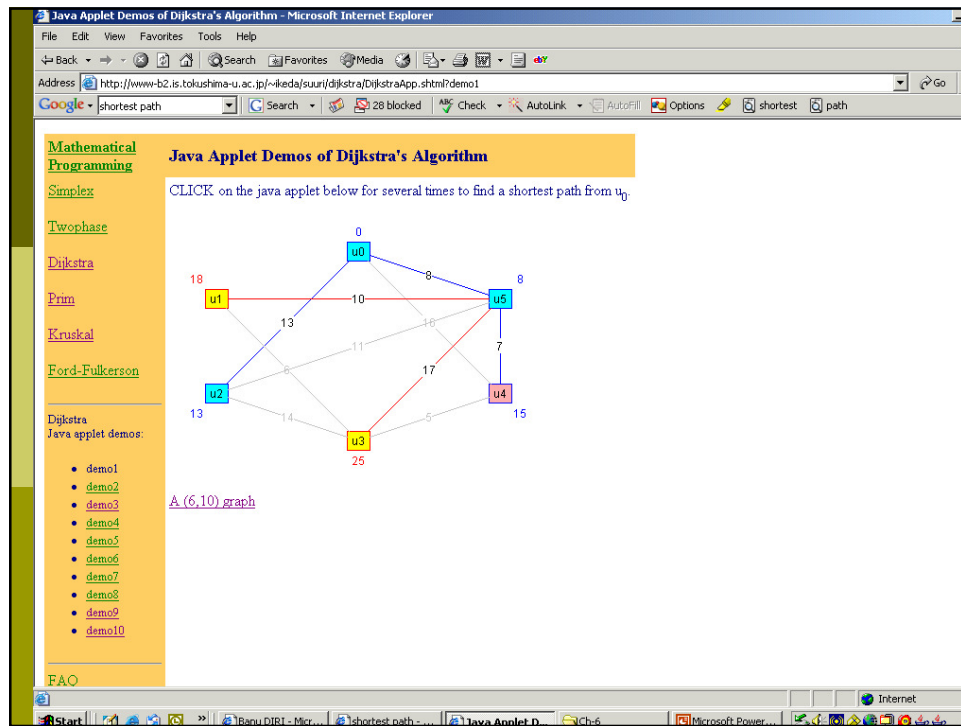
- demo1
- demo2
- demo3
- demo4
- demo5
- demo6
- demo7
- demo8
- demo9
- demo10

FAO

**Java Applet Demos of Dijkstra's Algorithm**

CLICK on the java applet below for several times to find a shortest path from  $u_0$ .

A (6,10) graph



Java Applet Demos of Dijkstra's Algorithm - Microsoft Internet Explorer

Address: <http://www-b2.is.tokushima-u.ac.jp/~ikeda/suun/dijkstra/DijkstraApp.shtml?demo1>

Google shortest path

**Mathematical Programming**

[Simplex](#)

[Twophase](#)

[Dijkstra](#)

[Prim](#)

[Kruskal](#)

[Ford-Fulkerson](#)

**Dijkstra**

Java applet demos:

- demo1
- demo2
- demo3
- demo4
- demo5
- demo6
- demo7
- demo8
- demo9
- demo10

FAO

**Java Applet Demos of Dijkstra's Algorithm**

CLICK on the java applet below for several times to find a shortest path from  $u_0$ .

A (6,10) graph

Java Applet Demos of Dijkstra's Algorithm - Microsoft Internet Explorer

Address: <http://www-b2.is.tokushima-u.ac.jp/~ikeda/suun/dijkstra/DijkstraApp.shtml?demo1>

Google shortest path

**Mathematical Programming**

[Simplex](#)

[Twophase](#)

[Dijkstra](#)

[Prim](#)

[Kruskal](#)

[Ford-Fulkerson](#)

**Dijkstra**

Java applet demos:

- demo1
- demo2
- demo3
- demo4
- demo5
- demo6
- demo7
- demo8
- demo9
- demo10

FAO

**Java Applet Demos of Dijkstra's Algorithm**

CLICK on the java applet below for several times to find a shortest path from  $u_0$ .

A (6,10) graph



Java Applet Demos of Dijkstra's Algorithm - Microsoft Internet Explorer

Address: <http://www.b2.is.tokushima-u.ac.jp/~iked/suun/dijkstra/DijkstraApp.shtml?demo1>

Google shortest path

**Mathematical Programming**

[Simplex](#)

[Twophase](#)

[Dijkstra](#)

[Prim](#)

[Kruskal](#)

[Ford-Fulkerson](#)

**Dijkstra**

Java applet demos:

- [demo1](#)
- [demo2](#)
- [demo3](#)
- [demo4](#)
- [demo5](#)
- [demo6](#)
- [demo7](#)
- [demo8](#)
- [demo9](#)
- [demo10](#)

[FAO](#)

**Java Applet Demos of Dijkstra's Algorithm**

CLICK on the java applet below for several times to find a shortest path from  $u_0$ .

[A \(6,10\) graph](#)

Java Applet Demos of Dijkstra's Algorithm - Microsoft Internet Explorer

Address: <http://www.b2.is.tokushima-u.ac.jp/~iked/suun/dijkstra/DijkstraApp.shtml?demo1>

Google shortest path

**Mathematical Programming**

[Simplex](#)

[Twophase](#)

[Dijkstra](#)

[Prim](#)

[Kruskal](#)

[Ford-Fulkerson](#)

**Dijkstra**

Java applet demos:

- [demo1](#)
- [demo2](#)
- [demo3](#)
- [demo4](#)
- [demo5](#)
- [demo6](#)
- [demo7](#)
- [demo8](#)
- [demo9](#)
- [demo10](#)

[FAO](#)

**Java Applet Demos of Dijkstra's Algorithm**

CLICK on the java applet below for several times to find a shortest path from  $u_0$ .

[A \(6,10\) graph](#)

Java Applet Demos of Dijkstra's Algorithm - Microsoft Internet Explorer

Address: <http://www.b2.is.tokushima-u.ac.jp/~ikeda/suun/dijkstra/DijkstraApp.shtml/demo3>

Google shortest path

**Mathematical Programming**

[Simplex](#)

[Twophase](#)

[Dijkstra](#)

[Prim](#)

[Kruskal](#)

[Ford-Fulkerson](#)

**Dijkstra**

Java applet demos:

- [demo1](#)
- [demo2](#)
- [demo3](#)
- [demo4](#)
- [demo5](#)
- [demo6](#)
- [demo7](#)
- [demo8](#)
- [demo9](#)
- [demo10](#)

[FAQ](#)

**Java Applet Demos of Dijkstra's Algorithm**

Click on the applet below for several times to find a shortest path from node a.

A (12,23) graph

Java Applet Demos of Dijkstra's Algorithm - Microsoft Internet Explorer

Address: <http://www.b2.is.tokushima-u.ac.jp/~ikeda/suun/dijkstra/DijkstraApp.shtml/demo3>

Google shortest path

**Mathematical Programming**

[Simplex](#)

[Twophase](#)

[Dijkstra](#)

[Prim](#)

[Kruskal](#)

[Ford-Fulkerson](#)

**Dijkstra**

Java applet demos:

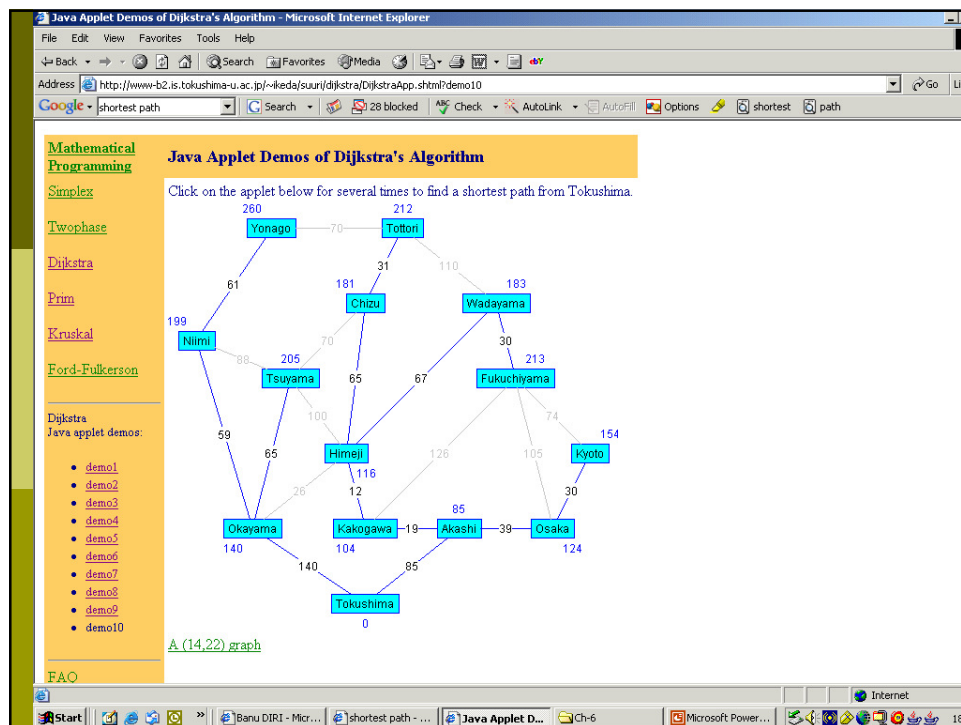
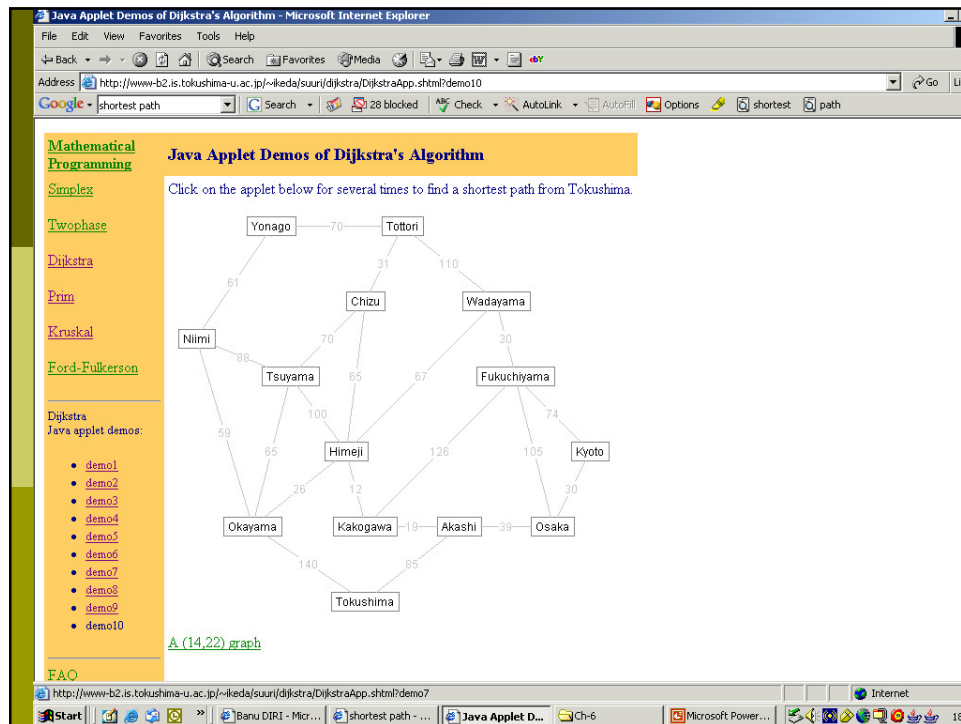
- [demo1](#)
- [demo2](#)
- [demo3](#)
- [demo4](#)
- [demo5](#)
- [demo6](#)
- [demo7](#)
- [demo8](#)
- [demo9](#)
- [demo10](#)

[FAQ](#)

**Java Applet Demos of Dijkstra's Algorithm**

Click on the applet below for several times to find a shortest path from node a.

A (12,23) graph



# Graf Modelleri

Farklı alanlarda farklı graf modelleri kullanılır.

**Niche Overlap Graf :** Eko sistem içerisindeki farklı grupları modellemede kullanılır.

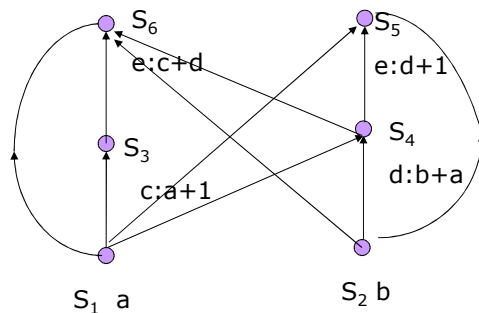
**Influence Graf:** Grup çalışmalarında, grup içerisindeki kişilerin birbirlerini etkilemesini modellemede kullanılır.

**Round-Robin Tournament Graf:** Turnuvada yer alan her takımın, hangi takımla karşılaştığını ve oyunu kimin kazandığını göstermede kullanılır.

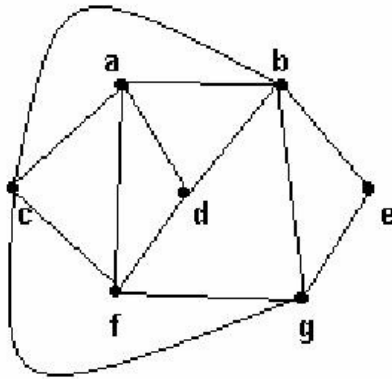
**Precedence Graf:** Bir işlemin sonucu, kendisinden önce gelen işlemin sonucuna bağlı olarak değişen sistemleri modellemede kullanılır.

## Precedence grafa örnek....

$S_1$  **a:0**    $S_2$  **b:1**    $S_3$  **c:a+1**    $S_4$  **d:b+a**    $S_5$  **e:d+1**    $S_6$  **e:c+d**



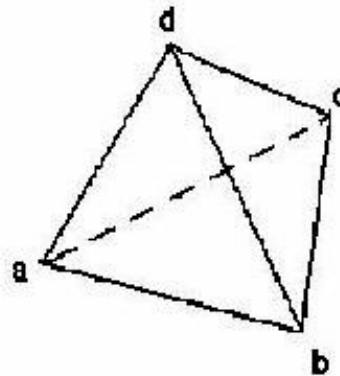
## Planar Graflar



Bir  $G$  grafinın kenarları birbirlerini kesmeyecek şekilde çizilebiliyorsa **Planar** graf olarak adlandırılır.

## Euler'in formülü

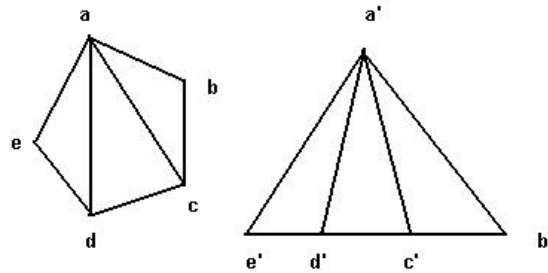
- Eğer  $G$  bir *planar* graph is
  - $v$  = düğüm sayısı
  - $e$  = kenar sayısı
  - $f$  = yüzey sayısı
- Öyleyse  $v - e + f = 2$



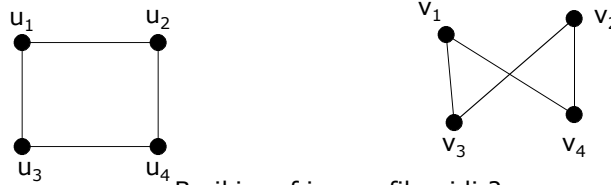
## İzomorfik (Isomorphic) Graflar

İki grafın izomorfik olup olmadığı nasıl kontrol edilir ?

- ▣ Kenar sayıları aynı olmalıdır.
- ▣ Düğüm sayıları aynı olmalıdır.
- ▣ Düğüm dereceleri aynı olmalıdır.
- ▣ Düğümler arasındaki ilişkiyi gösteren matrisler aynı olmalıdır. Bu matrislerdeki benzerlik satır ve sütunlardaki yer değişikliği ile de sağlanabilir.



### Örnek



Bu iki graf izomorfik midir?

**EYET**

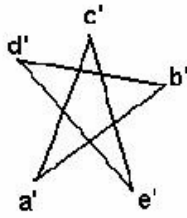
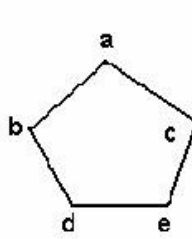
Her iki grafında 4 düğümü, 4 kenarı ve her düğümünün de derecesi 2 dir

	u1	u2	u3	u4		v1	v2	v3	v4
u1	0	1	1	0	v1	0	0	1	1
u2	1	0	0	1	v2	0	0	1	1
u3	1	0	0	1	v3	1	1	0	0
u4	0	1	1	0	v4	1	1	0	0

*u<sub>2</sub> ve u<sub>4</sub> satır ve sütunlar yerdeğiřtiriyor*

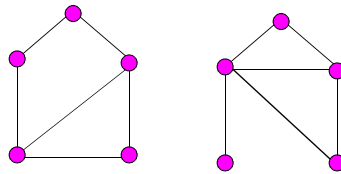
## Örnek

□ Aşağıda verilmiş olan iki graf izomorfik midir? **EVET**



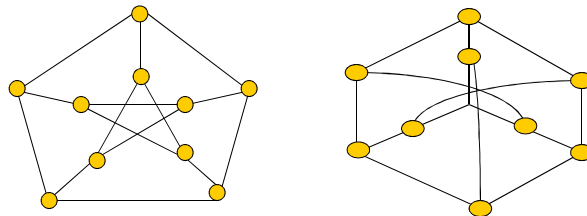
	a	b	c	d	e
a	0	1	1	0	0
b	1	0	0	1	0
c	1	0	0	0	1
d	0	1	0	0	1
e	0	0	1	1	0

## Örnek



Bu iki graf izomorfik midir ?

**HAYIR**



Bu iki graf izomorfik midir ?

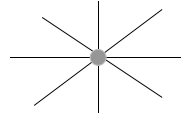
**EVET**

## Özel Tip Graflar

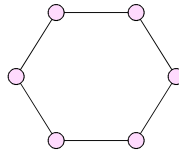
❑ Özel tip graflar genellikle veri iletişimi ve paralel veri işleme uygulamalarında kullanılır.

**Local Area Network :** Bir bina içerisindeki midi ve pc gibi farklı bilgisayarları ve çevre birimlerini birbirine bağlamak için kullanılır. Bu ağların farklı topolojileri mevcuttur.

« **Star Topology** : Bütün cihazlar, merkezdeki cihaz üzerinden birbirlerine bağlanırlar.  $K_{1,n}$  complete Bipartite Graf kullanılır.



« **Ring Topology** : Bu modelde her cihaz diğer iki farklı cihaz ile birbirine bağlıdır. n-cycles  $C_n$  modelidir.



« **Hybrid Topology** : Star ve Ring topology'sini birlikte kullanır. Bu tekrarlılık network'ün daha güvenli olmasını sağlar. Wheel,  $W_n$  graf modeline karşılık gelir.

