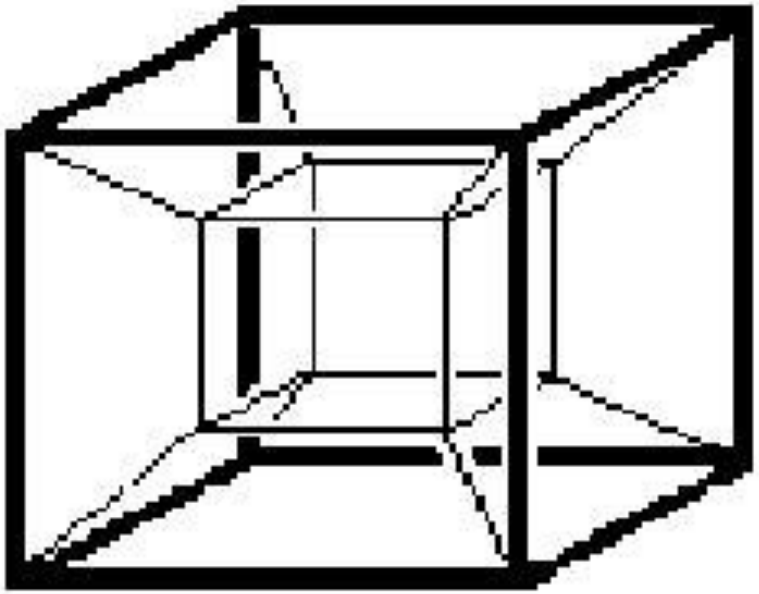
Graf Teorisi
(Graph Theory)

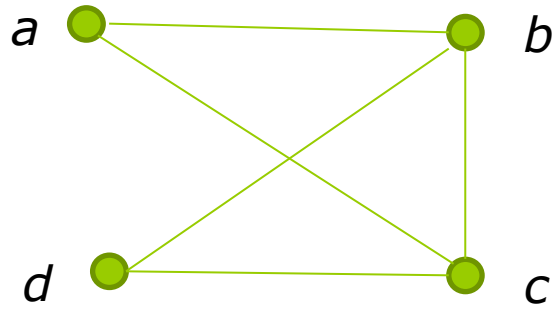


Graf – Çizge

Tanım: Bir G = (V, E) grafı, boş olmayan bir dizi V köşesinden (veya düğümlerden) ve bir dizi E kenarından oluşur. Her kenarın, uç noktaları adı verilen, kendisiyle ilişkili bir veya iki köşesi vardır. Bir kenar, graftaki bu köşeleri birbirine bağlayan ve iki köşe arasında bir ilişkiyi gösteren elemandır

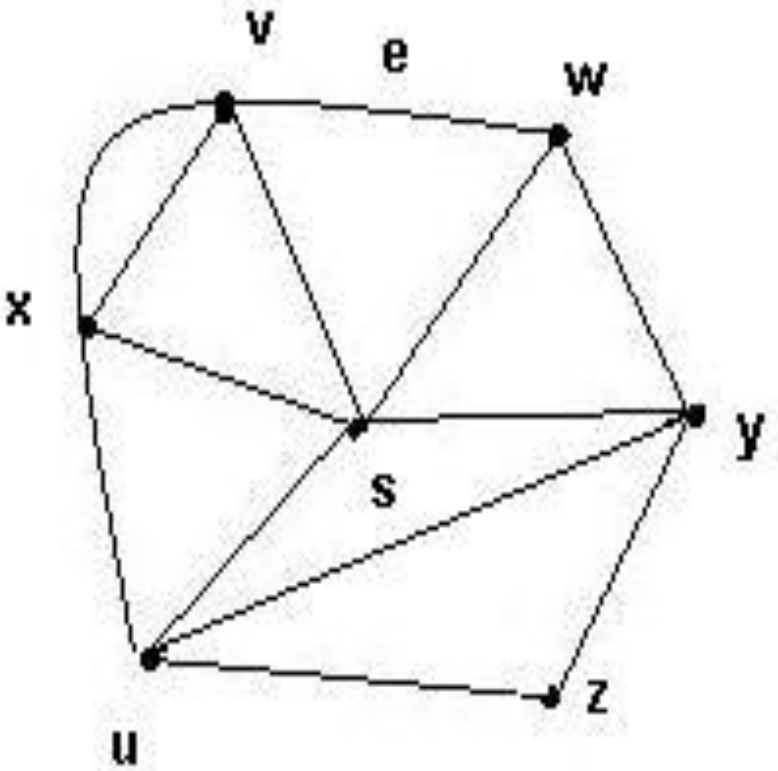
Örnek:

Dört köşeli ve beş kenarlı bir graf



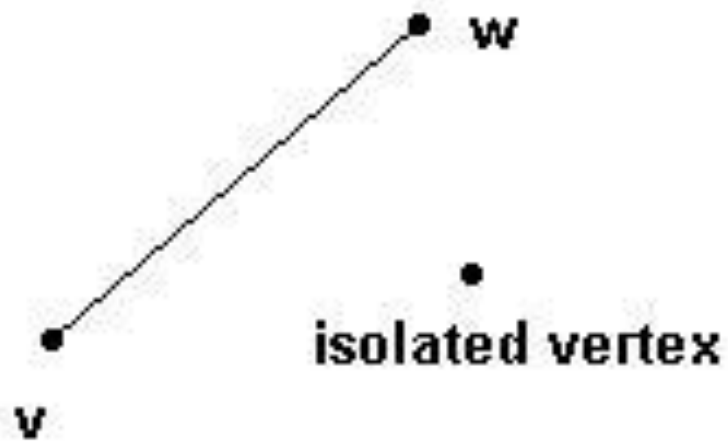
Giriş

- G grafı 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)}



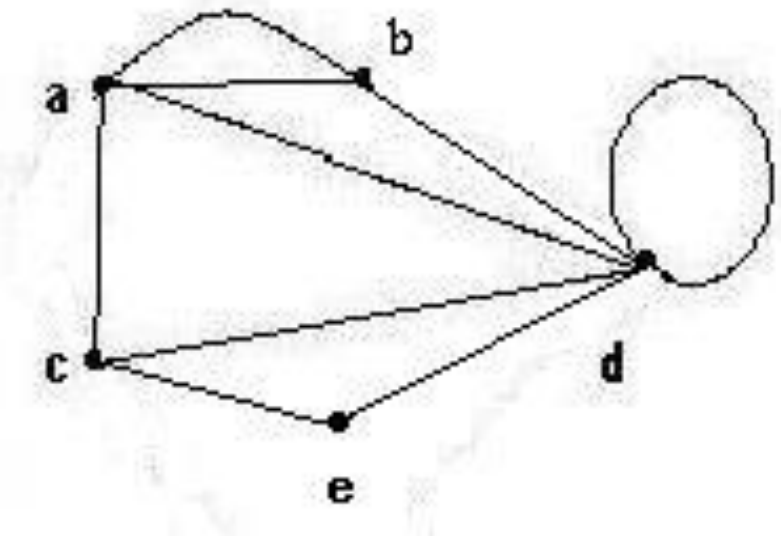
Kenarlar (Edges)

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



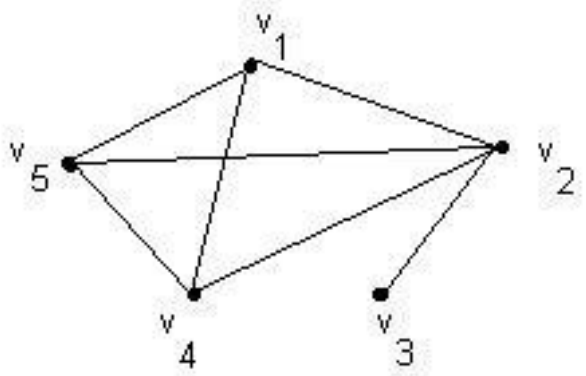
Özel Kenarlar

- 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.

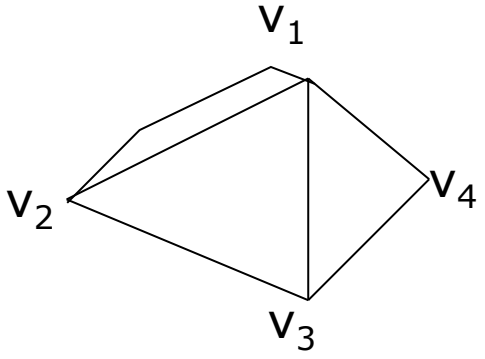


Özel Graflar

- Basit (Simple) Graf
  - Yönsüz, paralel kenar olmayan ve döngü içermeyen graflardır.



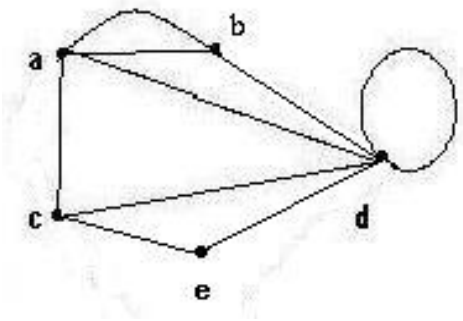
- Çoklu (Multi) Graf
  - Basit grafların yeterli olmadığı durumlarda kullanılır.
  - Yönsüz, paralel kenarı olan ve döngü içermeyen graflardır.



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

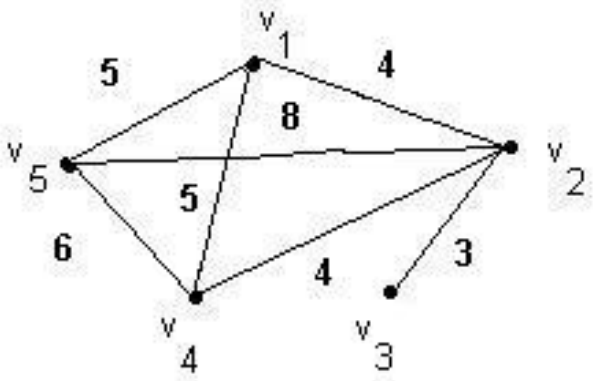
Pseudo Graflar

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



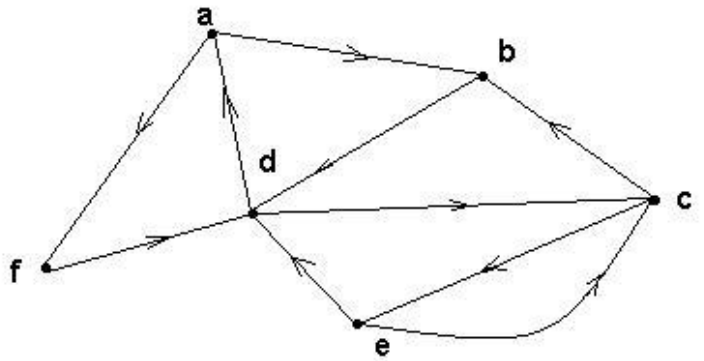
Ağırlıklı (Weighted) Graf

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



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

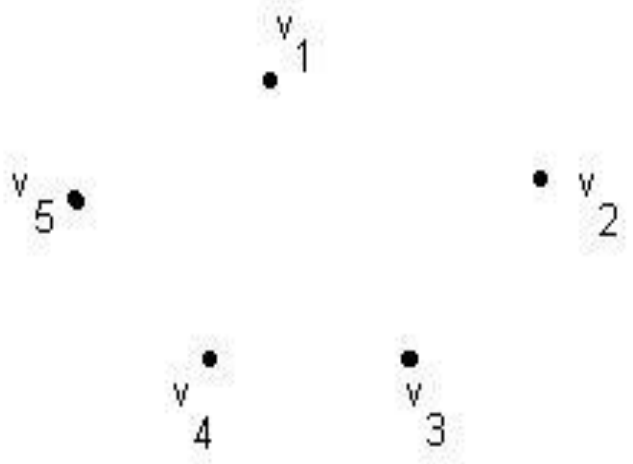
Graflarda benzerlik (2)

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 grafını aşağıdaki gibi oluşsun:
  - 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)

$s(v_1,v_2) = 36$ $s(v_2,v_3) = 38$ $s(v_3,v_4) = 54$

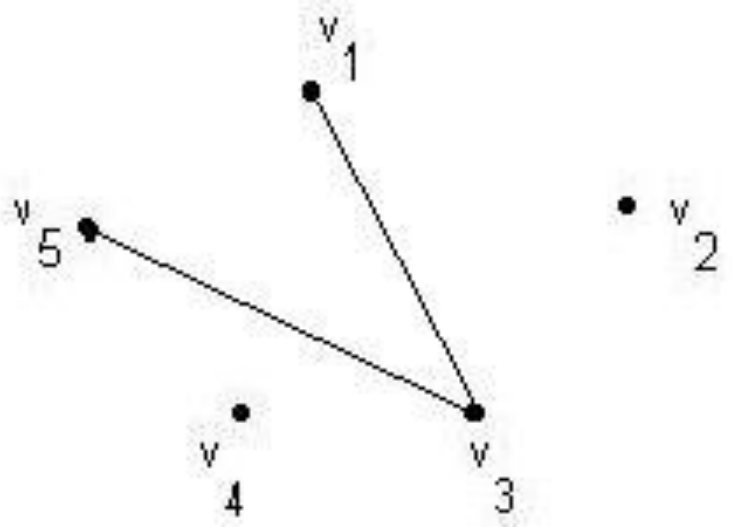
$s(v_1,v_3) = 24$ $s(v_2,v_4) = 76$ $s(v_3,v_5) = 20$

$s(v_1,v_4) = 42$ $s(v_2,v_5) = 48$ $s(v_4,v_5) = 46$

$s(v_1,v_5) = 30$

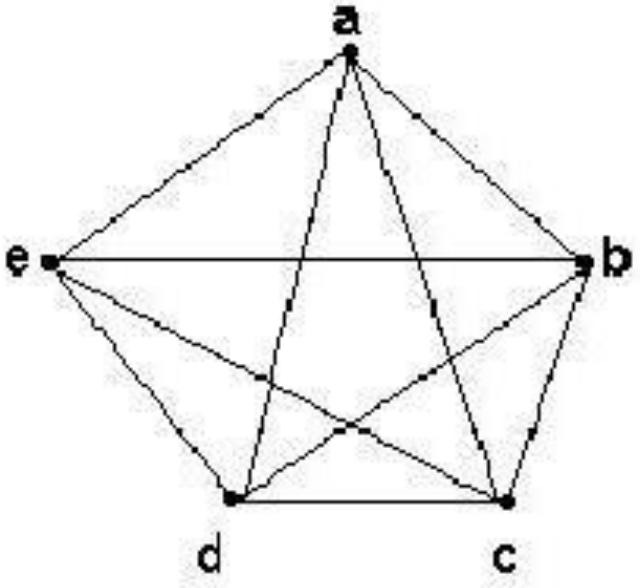
Benzer olmayan fonksiyonlar(2)

- 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

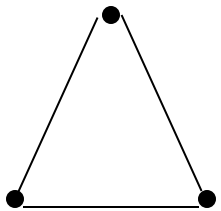
- $n \geq 3$
- *complete graph* K_n : n adet düğüm içeren basit graf yapısındadır. Her düğüm, diğer düğümlere bir kenar ile bağlantılıdır.
- Şekilde K_5 grafı gösterilmiştir.



- 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 $\{v_1,v_2\}$, $\{v_2,v_3\}$, ..., $\{v_{n-1},v_n\}$, $\{v_n,v_1\}$, düğüm çiftlerinden oluşan kenarlardan meydana gelir.
- Şekilde C_3 grafı gösterilmiştir.

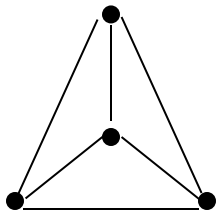


C_3

- 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 grafı gösterilmiştir.



W_3

- Soru: W_4, W_5, W_6 graflarını çiziniz.

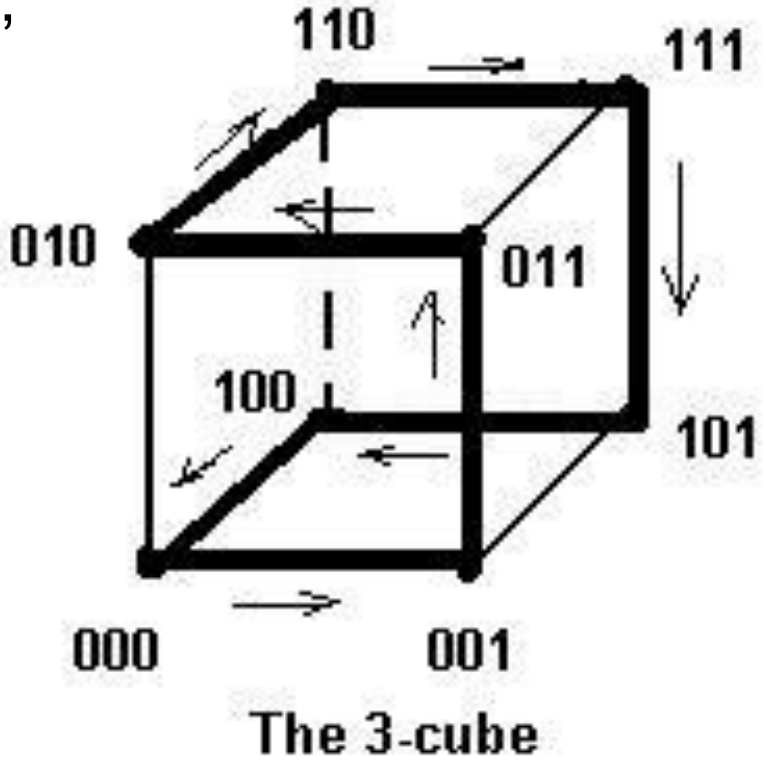
N-Cube (Küp) Graf Q_n

- *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)

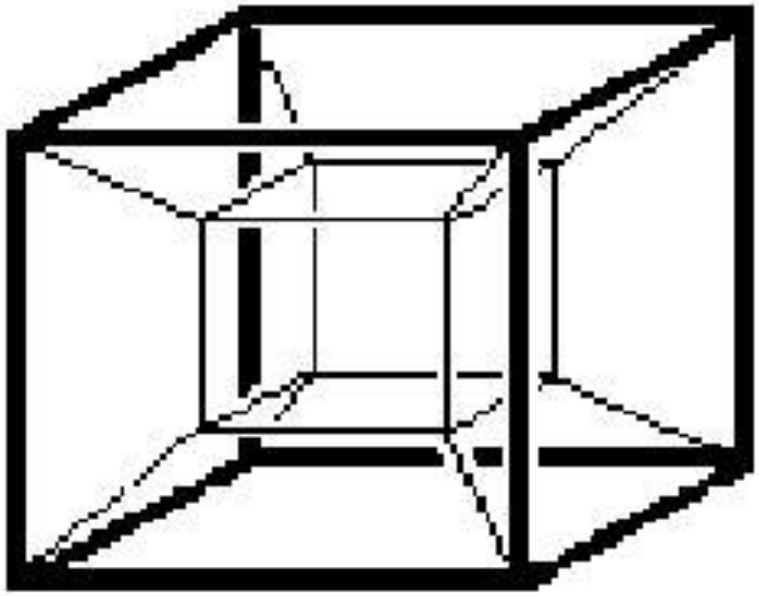
- Şekilde Q_3 grafı gösterilmiştir.

  Soru: Q_1, Q_2 graflarını çiziniz.



The 3-cube

hypercube veya 4-cube



16 düğüm, 32 kenar ve 20 yüzey

- Düğüm etiketleri:

0000 0001 0010 0011

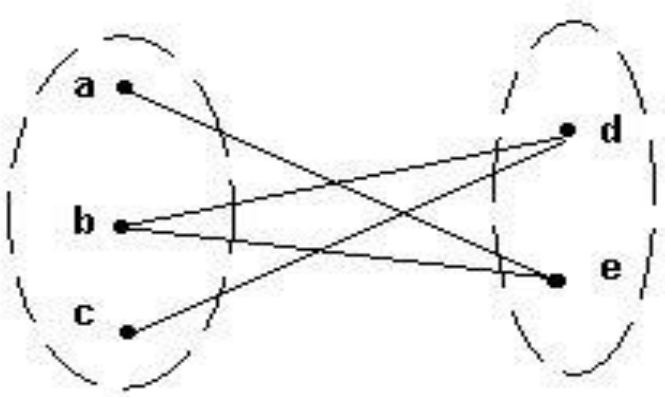
0100 0101 0110 0111

1000 1001 1010 1011

1100 1101 1110 1111

İki Parçalı (Bipartite) Graflar

- G, bipartite graf ise:
  - $V(G) = V(G_1) \cup V(G_2)$
  - $|V(G_1)| = m$, $|V(G_2)| = n$
  - $V(G_1) \cap V(G_2) = \varnothing$



• Bir grafı oluşturan düğümleri iki ayrı kümeye bölerek grafı 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

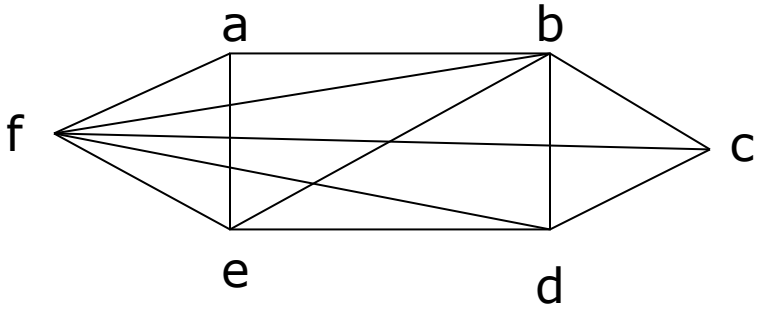
- C_6 Bipartite graf mıdır?

  Evet

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

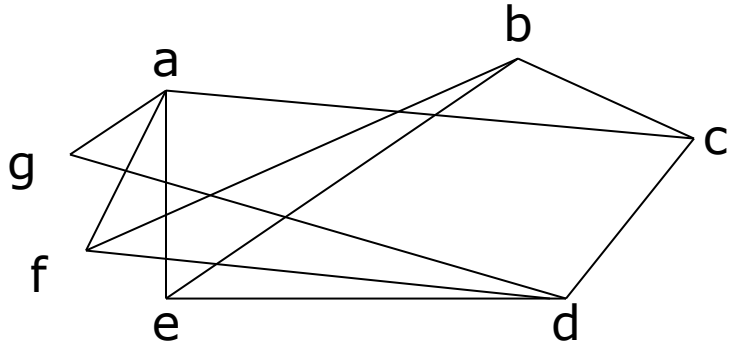
Yandaki graf Bipartite graf mıdır?

Hayır

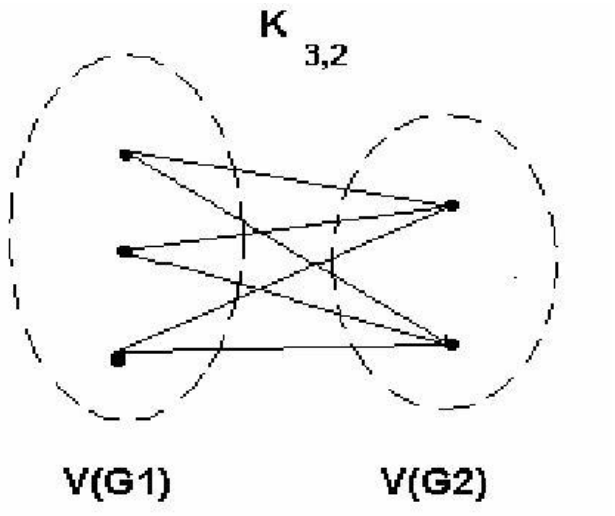


Yandaki graf Bipartite graf mıdır?

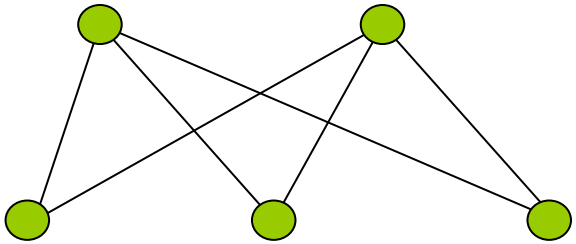
Evet. {a,b,d} ve {c,e,f,g}



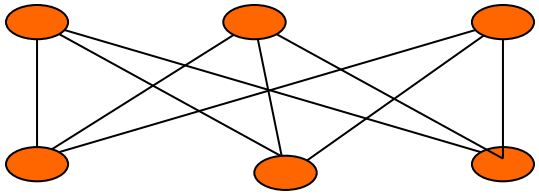
Tam (complete) bipartite graph $K_{m,n}$



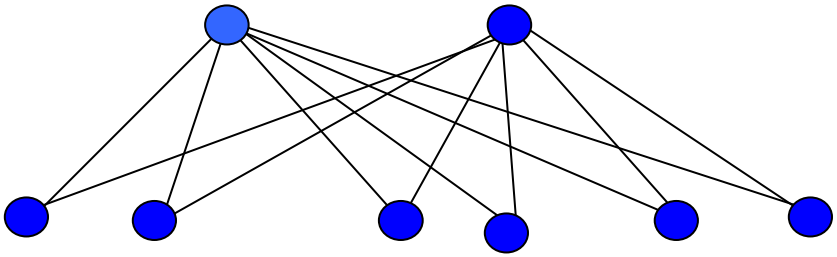
- *complete* bipartite graf $K_{m,n}$ şeklinde gösterilir. İlgili grafın düğümlerinin kümesi m ve n elemanlı iki alt kümeye ayrılır.
- Bir kenarı birbirine bağlayan iki düğümünde farklı alt kümelerin elemanı olmak zorundadırlar.
- $|V(G_1)| = m$
- $|V(G_2)| = n$



$K_{2,3}$



$K_{3,3}$

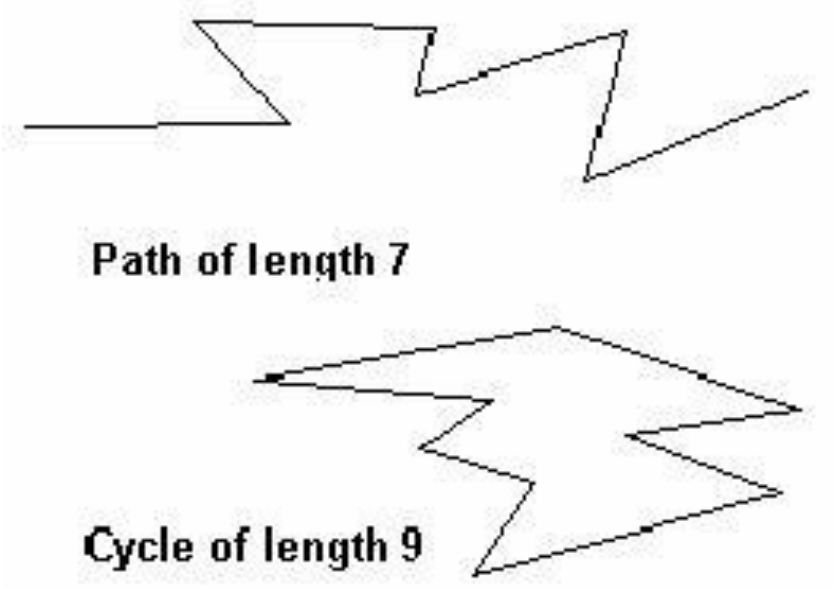


$K_{2,6}$

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

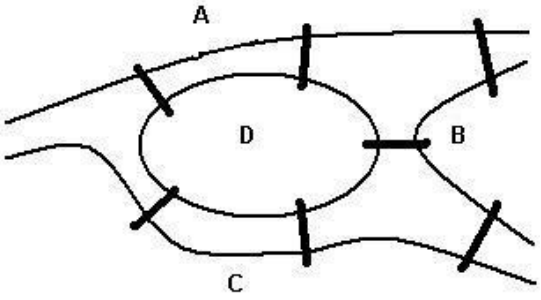
Yollar (Paths) ve Döngüler(Cycles)

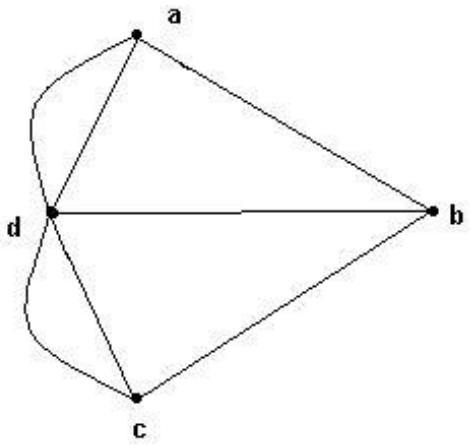


- **n** uzunluğundaki bir yol'un (path) **n+1** adet düğümü ve n adet de ardışık kenarı vardır
- Bir döngü içeren yol başladığı düğümde son bulur. Uzunluğu **n** olan bir döngüde **n** adet düğüm vardır.

Euler Döngüsü (Euler cycles)

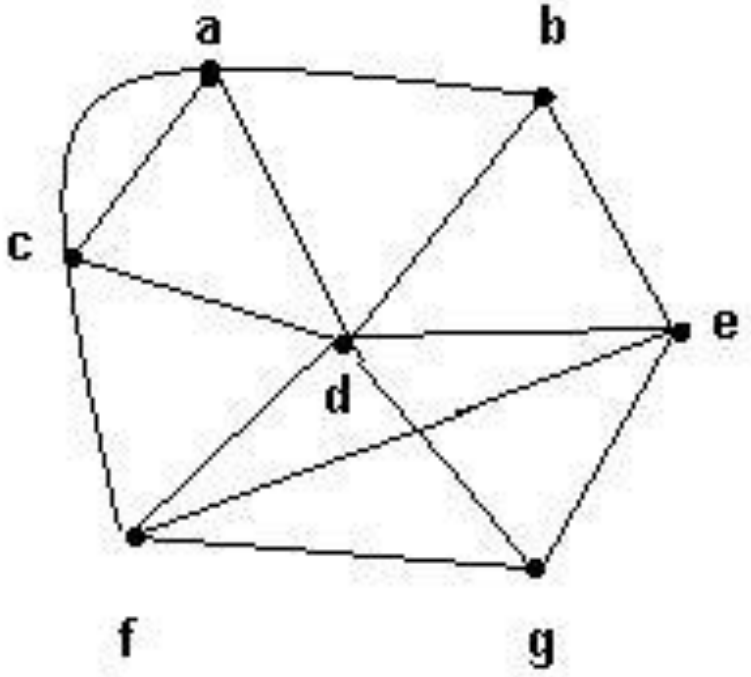
- G grafı içerisindeki *Euler cycle* basit bir çevrim olup G grafı içerisindeki her kenardan sadece bir kez geçilmesine izin verir.
- Königsberg köprü problemi:
  - Başlangıç ve Bitiş noktası aynıdır, yedi köprüden sadece bir kez geçerek başlangıç noktasına dönmek mümkün müdür?
  - Bu problemi grafa indirgeyelim.
  - Kenarlar köprüleri ve düğüm noktalarıda bölgeleri göstersin.





Bir düğümün derecesi

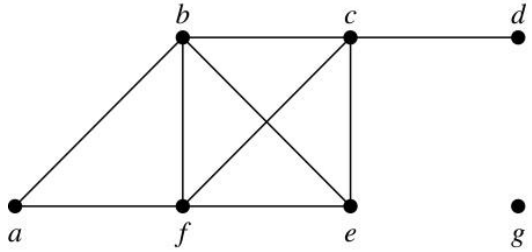
- **v** düğümünün derecesi $\delta(v)$ ile gösterilir ve bu da yönsüz bir grafta düğüme gelen kenarlar toplamıdır. Düğüm noktalarındaki döngü düğüm derecesine 2 kez katılır.



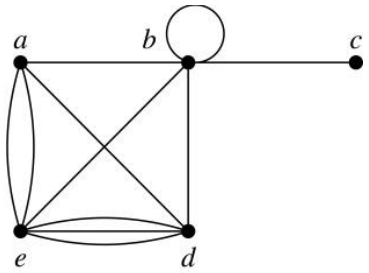
- **Örnek**:
  - $\delta(a) = 4$, $\delta(b) = 3$,
  - $\delta(c) = 4$, $\delta(d) = 6$,
  - $\delta(e) = 4$, $\delta(f) = 4$,
  - $\delta(g) = 3$.

Degrees and Neighborhoods of Vertices

Example: What are the degrees and neighborhoods of the vertices in the graphs *G* and *H*?



G



H

Solution:

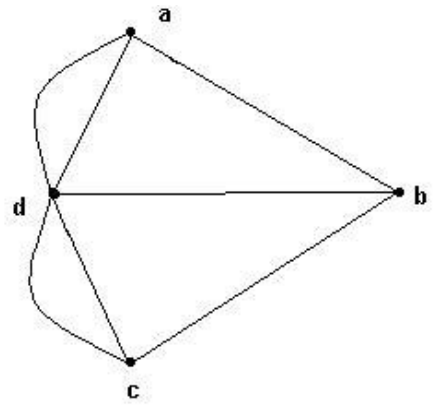
G: $\deg(a) = 2$, $\deg(b) = \deg(c) = \deg(f) = 4$, $\deg(d) = 1$, $\deg(e) = 3$, $\deg(g) = 0$.

$N(a)=\{b, f\}$, $N(b)=\{a, c, e, f\}$, $N(c)=\{b, d, e, f\}$, $N(d)=\{c\}$, $N(e) = \{b, c, f\}$, $N(f) = \{a, b, c, e\}$, $N(g) = \varnothing$.

H: $\deg(a) = 4$, $\deg(b) = \deg(e) = 6$, $\deg(c) = 1$, $\deg(d) = 5$.

$N(a) = \{b, d, e\}$, $N(b) = \{a, b, c, d, e\}$, $N(c) = \{b\}$, $N(d) = \{a, b, e\}$, $N(e) = \{a, b, d\}$.

Euler Grafı



- Bir **G** grafı *Euler cycle'*ına sahip ise *Euler Grafı* adını alır.
- Euler grafında tüm düğümlerin derecesi çifttir.
- Konigsberg bridge problemi bir Euler grafı değildir.
- Konigsberg 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

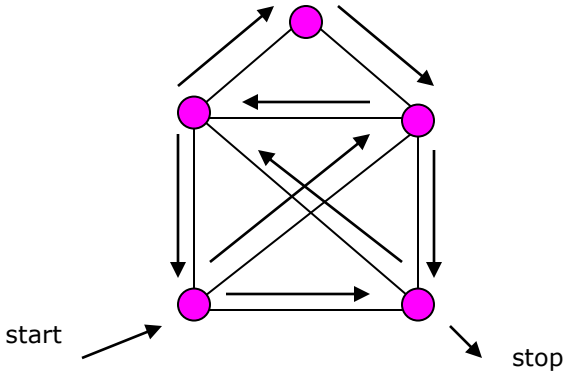
- G grafında (v,w) yönlü bir kenar olsun ve yön *v*'den *w*'ya verilsin. *v* ***initial vertex***, *w*'da ***terminal*** veya ***end vertex*** olarak adlandırılır. Bir düğüm noktasında döngü söz konusu ise bu düğümün *initial vertex*'i ve *end vertex*'i birbirinin aynıdır.

- Yönlü bir grafta, herhangi bir düğümün ***in_degree'***si $\delta^-(v)$, ***out_degree'***si $\delta^+(v)$ olarak gösterilir.

- Yönlü bir grafın in_degree ve out_degree'lerinin toplamı birbirinin aynıdır.

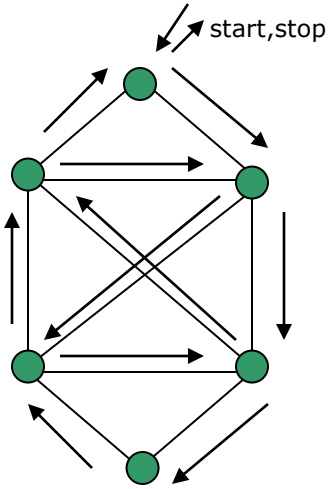
$$\sum_{v \in V} \delta^-(v) = \sum_{w \in V} \delta^+(v)$$

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

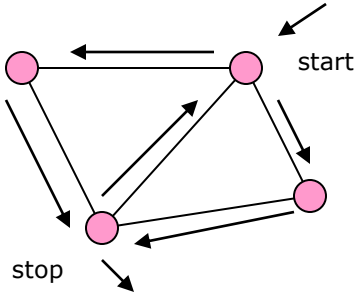


Path var, Euler grafı değil

Düğüm dereceleri çift değil

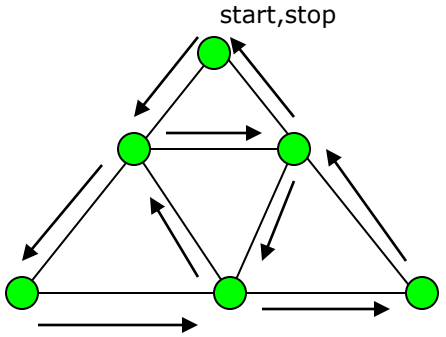


Euler grafı



Path var, Euler grafı değil

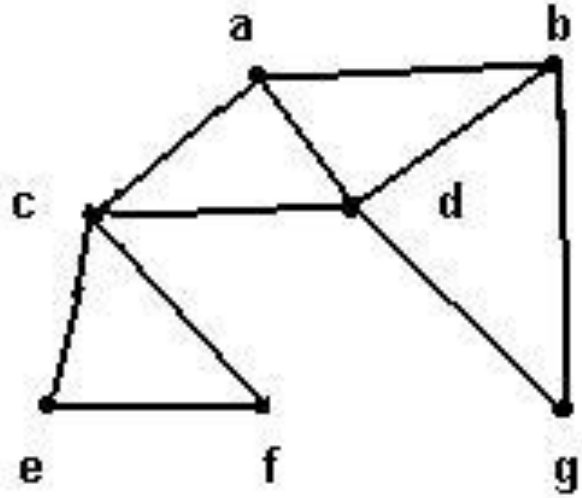
Düğüm dereceleri çift değil



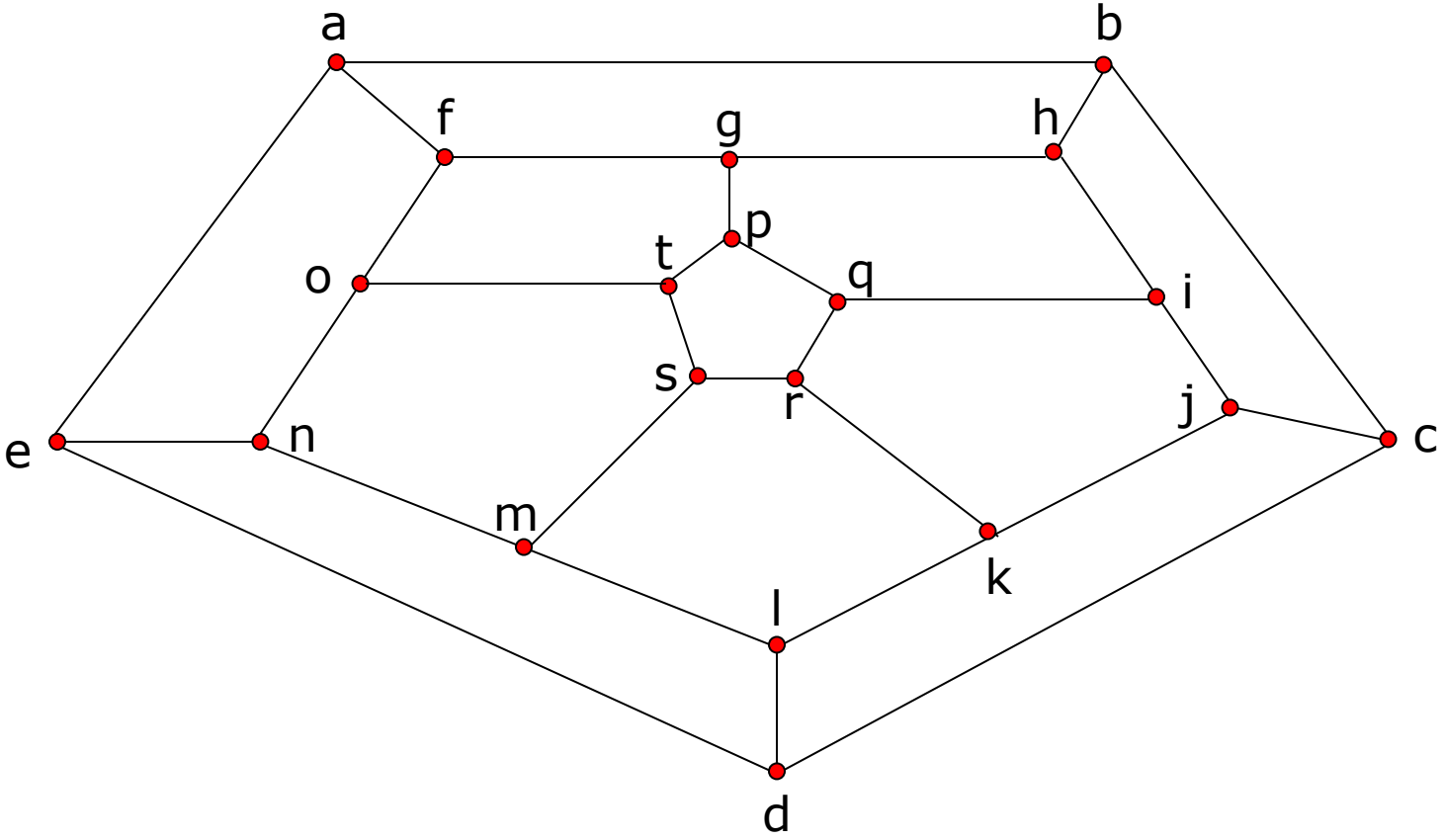
Euler grafı

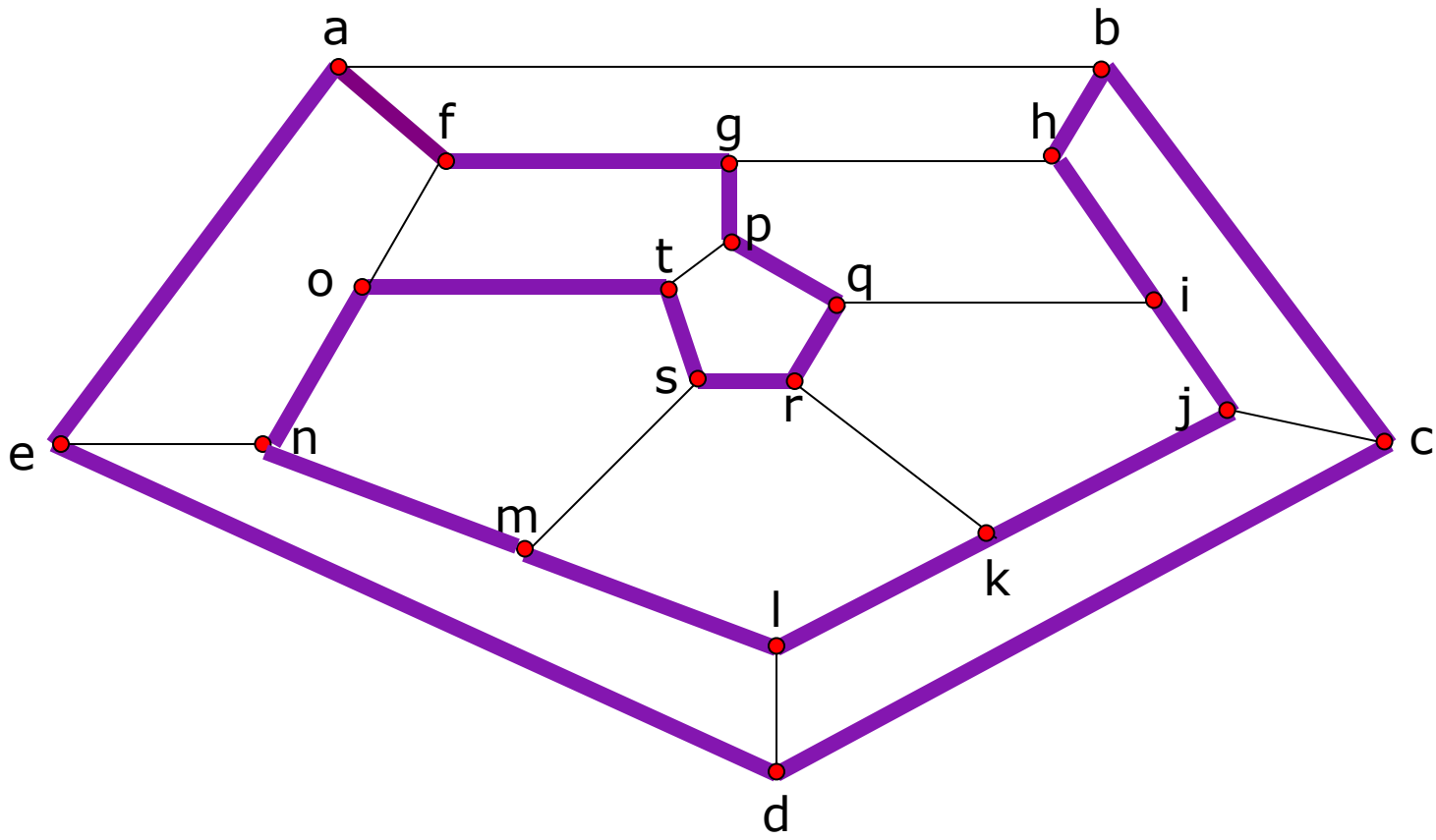
Hamilton Döngüsü (Hamiltonian Cycles)

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



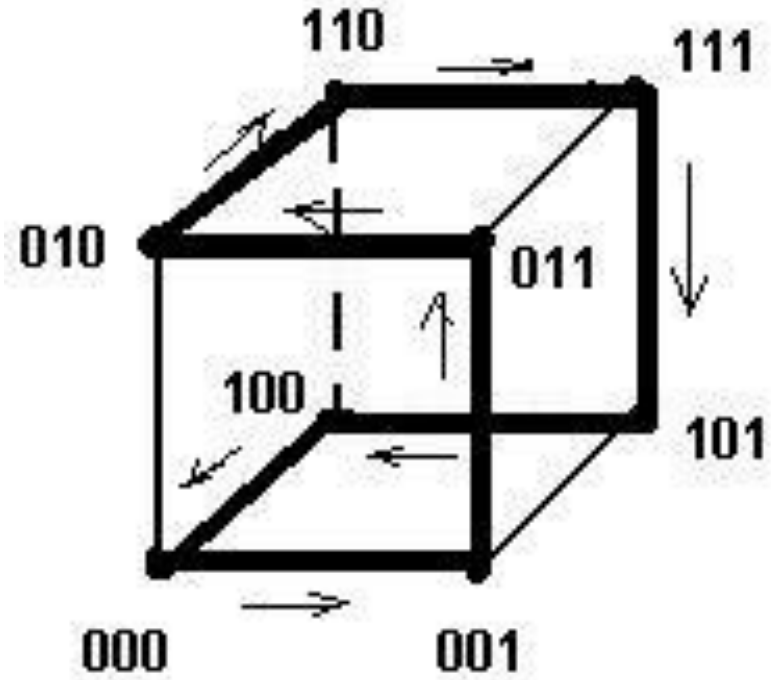
A non-Hamiltonian graph


a
b
f
g
h
p
t
o
q
i
s
r
j
e
n
c
m
k
l
d


a
b
f
g
h
p
t
o
q
i
s
r
j
e
n
c
m
k
l
d

3-cube

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



The 3-cube

Grafların Temsil Edilmesi
Representing Graphs

- Adjacency Lists
- Adjacency Matrices
- Incidence Matrices

Representing Graphs: Adjacency Lists

Definition: An *adjacency list* can be used to represent a graph with no multiple edges by specifying the vertices that are adjacent to each vertex of the graph.

Example:

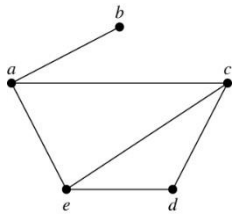


TABLE 1 An Adjacency List for a Simple Graph.

Vertex	*Adjacent Vertices*
a	b, c, e
b	a
c	a, d, e
d	c, e
e	a, c, d

Example:

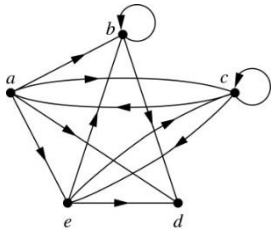


TABLE 2 An Adjacency List for a Directed Graph.

Initial Vertex	*Terminal Vertices*
a	b, c, d, e
b	b, d
c	a, c, e
d	
e	b, c, d

Representation of Graphs: Adjacency Matrices

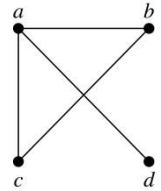
Definition: Suppose that $G = (V, E)$ is a simple graph where $|V| = n$. Arbitrarily list the vertices of G as $v_1, v_2, \ldots, v_n$. The *adjacency matrix* $\mathbf{A}_G$ of G, with respect to the listing of vertices, is the $n \times n$ zero-one matrix with 1 as its (i, j)th entry when v_i and v_j are adjacent, and 0 as its (i, j)th entry when they are not adjacent.

- In other words, if the graphs adjacency matrix is $\mathbf{A}_G = [a_{ij}]$, then

$$a_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \text{ is an edge of } G, \\ 0 & \text{otherwise.} \end{cases}$$

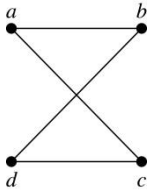
Adjacency Matrices (*continued*)

Example:



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

The ordering of vertices is a, b, c, d.



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

The ordering of vertices is a, b, c, d.

When a graph is sparse, that is, it has few edges relatively to the total number of possible edges, it is much more efficient to represent the graph using an adjacency list than an adjacency matrix. But for a dense graph, which includes a high percentage of possible edges, an adjacency matrix is preferable.

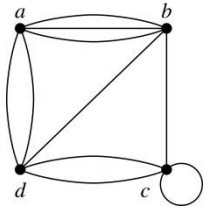
Note: The adjacency matrix of a simple graph is symmetric, i.e., $a_{ij} = a_{ji}$

Also, since there are no loops, each diagonal entry a_{ij} for $i = 1, 2, 3, \ldots, n$, is 0.

Adjacency Matrices (*continued*)

- Adjacency matrices can also be used to represent graphs with loops and multiple edges.
- A loop at the vertex v_i is represented by a 1 at the (i, j)th position of the matrix.
- When multiple edges connect the same pair of vertices v_i and v_j, (or if multiple loops are present at the same vertex), the (i, j)th entry equals the number of edges connecting the pair of vertices.

  Example: We give the adjacency matrix of the pseudograph shown here using the ordering of vertices a, b, c, d.



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

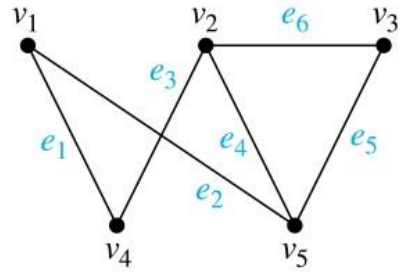
Representation of Graphs: Incidence Matrices

Definition: Let $G = (V, E)$ be an undirected graph with vertices where v_1, v_2, … v_n and edges e_1, e_2, … e_m. The incidence matrix with respect to the ordering of V and E is the $n \times m$ matrix $\mathbf{M} = [m_{ij}]$, where

$$m_{ij} = \begin{cases} 1 & \text{when edge } e_j \text{ is incident with } v_i, \\ 0 & \text{otherwise.} \end{cases}$$

Incidence Matrices (*continued*)

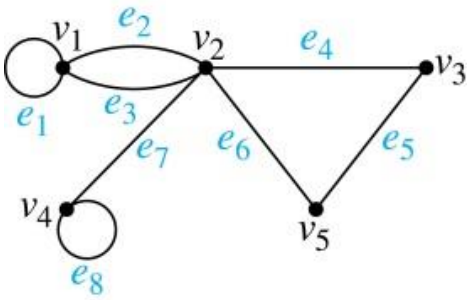
Example: Simple Graph and Incidence Matrix



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

The rows going from top to bottom represent v_1 through v_5 and the columns going from left to right represent e_1 through e_6.

Example: Pseudograph and Incidence Matrix



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

The rows going from top to bottom represent v_1 through v_5 and the columns going from left to right represent e_1 through e_8.

Connectivity

Paths

Informal Definition: A *path* is a sequence of edges that begins at a vertex of a graph and travels from vertex to vertex along edges of the graph. As the path travels along its edges, it visits the vertices along this path, that is, the endpoints of these.

Applications: Numerous problems can be modeled with paths formed by traveling along edges of graphs such as:

- determining whether a message can be sent between two computers.
- efficiently planning routes for mail delivery.

Paths

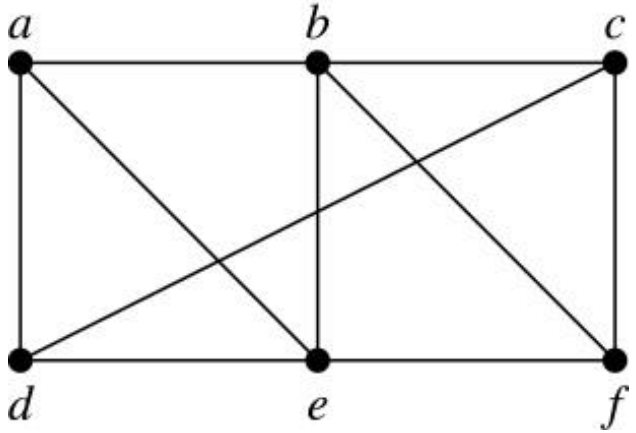
Definition: Let n be a nonnegative integer and G an undirected graph. A *path* of *length n* from u to v in G is a sequence of n edges $e_1, \ldots, e_n$ of G for which there exists a sequence $x_0 = u, x_1, \ldots, x_{n-1}, x_n = v$ of vertices such that e_i has, for $i = 1, \ldots, n$, the endpoints x_{i-1} and x_i.

- When the graph is simple, we denote this path by its vertex sequence $x_0, x_1, \ldots, x_n$ (since listing the vertices uniquely determines the path).
- The path is a *circuit* if it begins and ends at the same vertex ($u = v$) and has length greater than zero.
- The path or circuit is said to *pass through* the vertices $x_1, x_2, \ldots, x_{n-1}$ and *traverse* the edges $e_1, \ldots, e_n$.
- A path or circuit is *simple* if it does not contain the same edge more than once.

Paths (*continued*)

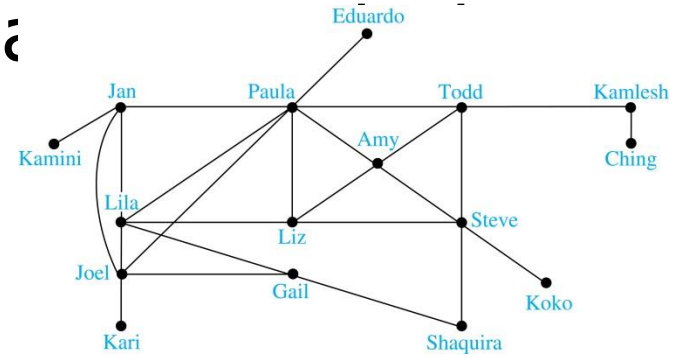
Example: In the simple graph here:

- *a*, *d*, *c*, *f*, *e* is a simple path of length 4.
- *d*, *e*, *c*, *a* is not a path because *e* is not connected to *c*.
- *b*, *c*, *f*, *e*, *b* is a circuit of length 4.
- *a*, *b*, *e*, *d*, *a*, *b* is a path of length 5, but it is not a simple path.



Degrees of Separation

Example: *Paths in Acquaintanceship Graphs*. In an acquaintanceship graph there is a path between two people if there is a chain of people linking these people, where two people adjacent in the chain know one another. In this graph there is a chain of six people linking Ka



Some have speculated that almost every pair of people in the world are linked by a small chain of no more than six, or maybe even, five people. The play *Six Degrees of Separation* by John Guare is based on this notion.

Bacon Numbrers

- In the Hollywood graph, two actors *a* and *b* are linked when there is a chain of actors linking *a* and *b*, where every two actors adjacent in the chain have acted in the same movie.
- The *Bacon number* of an actor *c* is defined to be the length of the shortest path connecting *c* and the well-known actor Kevin Bacon. (Note that we can define a similar number by replacing Kevin Bacon by a different actor.)

  The *oracle of Bacon* web site http://oracleofbacon.org/how.php provides a tool for finding Bacon numbers.

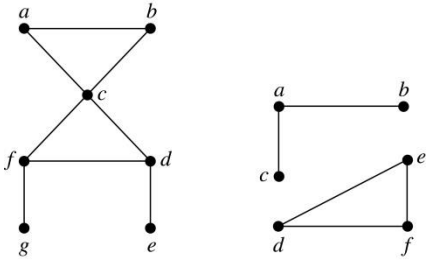
TABLE 2 The Number of Actors with a Given Bacon Number (as of early 2011).

Bacon Number	*Number of People*
0	1
1	2,367
2	242,407
3	785,389
4	200,602
5	14,048
6	1,277
7	114
8	16

Connectedness in Undirected Graphs

Definition: An undirected graph is called *connected* if there is a path between every pair of vertices. An undirected graph that is not *connected* is called *disconnected*. We say that we *disconnect* a graph when we remove vertices or edges, or both, to produce a disconnected subgraph.

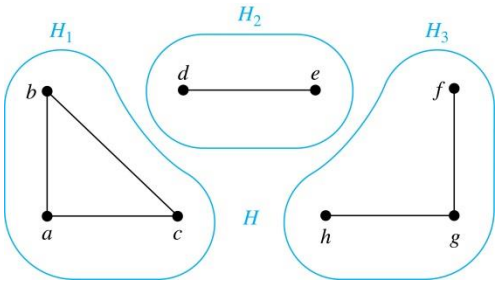
Example: G_1 is connected because there is a path between any pair of its vertices, as can be easily seen. However G_2 is not connected because there is no path between vertices *a* and *f*, for example.



Connected Components

Definition: A *connected component* of a graph *G* is a connected subgraph of *G* that is not a proper subgraph of another connected subgraph of *G*. A graph *G* that is not connected has two or more connected components that are disjoint and have *G* as their union.

Example: The graph *H* is the union of three disjoint subgraphs H_1, H_2, and H_3, none of which are proper subgraphs of a larger connected subgraph of *G*.These three subgraphs are the connected components of *H*.

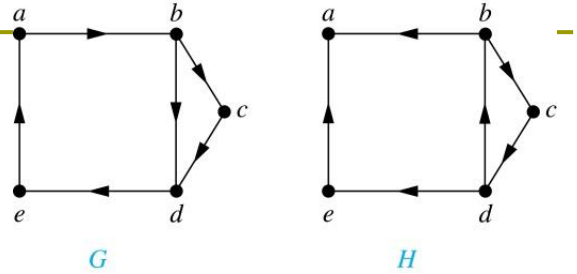


Connectedness in Directed Graphs

Definition: A directed graph is *strongly connected* if there is a path from *a* to *b* and a path from *b* to *a* whenever *a* and *b* are vertices in the graph.

Definition: A directed graph is *weakly connected* if there is a path between every two vertices in the underlying undirected graph, which is the undirected graph obtained by ignoring the directions of the edges of the directed graph.

Connectedness in Directed Graphs (*continued*)



Example: *G* is strongly connected because there is a path between any two vertices in the directed graph. Hence, *G* is also weakly connected. The graph *H* is not strongly connected, since there is no directed path from *a* to *b*, but it is weakly connected.

Definition: The subgraphs of a directed graph *G* that are strongly connected but not contained in larger strongly connected subgraphs, that is, the maximal strongly connected subgraphs, are called the *strongly connected components* or *strong components* of *G*.

Example (*continued***)**: The graph *H* has three strongly connected components, consisting of the vertex *a*; the vertex *e;* and the subgraph consisting of the vertices *b*, *c*, *d* and edges (*b*,*c*), (*c*,*d*), and (*d*,*b*).

Counting Paths between Vertices

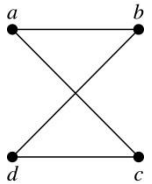
- We can use the adjacency matrix of a graph to find the number of paths between two vertices in the graph.

  Theorem: Let G be a graph with adjacency matrix **A** with respect to the ordering $v_1, \ldots, v_n$ of vertices (with directed or undirected edges, multiple edges and loops allowed). The number of different paths of length r from v_i to v_j, where $r > 0$ is a positive integer, equals the (i,j)th entry of $\mathbf{A}^r$.

Counting Paths between Vertices (*continued*)

Example: How many paths of length four are there from *a* to *d* in the graph G.

G



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

adjacency matrix of G

Solution: The adjacency matrix of *G* (ordering the vertices as *a*, *b*, *c*, *d*) is given above. Hence the number of paths of length four from *a* to *d* is the $(1, 4)$th entry of $\mathbf{A}^4$. The eight paths are as:

a, *b*, *a*, *b*, *d*	*a*, *b*, *a*, *c*, *d*
a, *b*, *d*, *b*, *d*	*a*, *b*, *d*, *c*, *d*
a, *c*, *a*, *b*, *d*	*a*, *c*, *a*, *c*, *d*
a, *c*, *d*, *b*, *d*	*a*, *c*, *d*, *c*, *d*

$$\mathbf{A}^4 = \begin{bmatrix} 8 & 0 & 0 & 8 \\ 0 & 8 & 8 & 0 \\ 0 & 8 & 8 & 0 \\ 8 & 0 & 0 & 8 \end{bmatrix}$$

EN KISA YOL (SHORTEST PATH) ALGORİTMASI

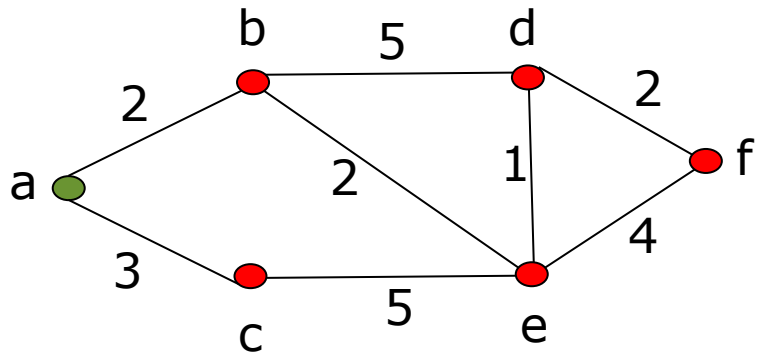
Dijkstra's Algorithm

Dijkstra's Algorithm

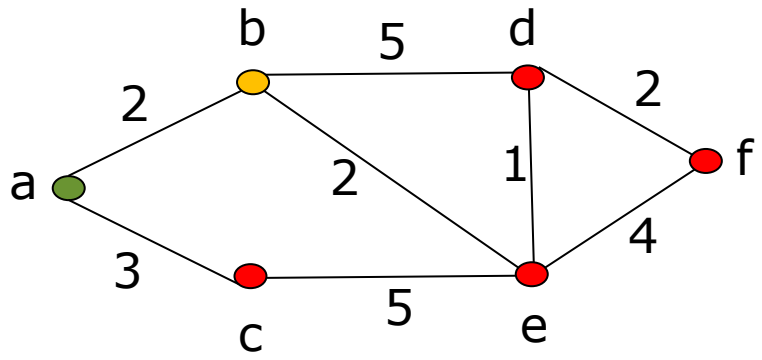
Dijkstra's algorithm is known to be a good algorithm to find a shortest path.

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

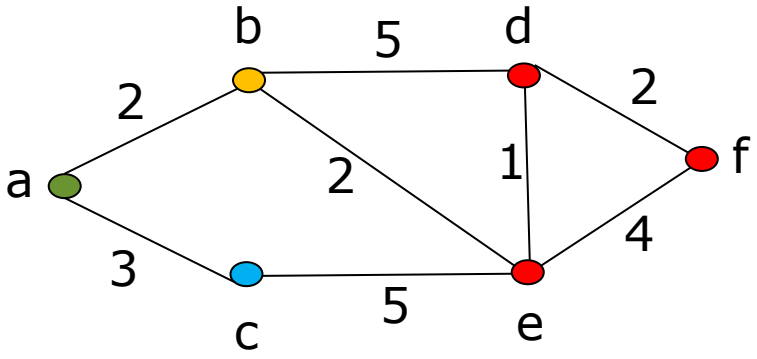
The time required by Dijkstra's algorithm is $O(|V|^2)$.
It will be reduced to $O(|E|\log|V|)$ if heap is used to keep
$\{v \text{ in } V \backslash S_i : L(v) < \text{infinity}\}$.



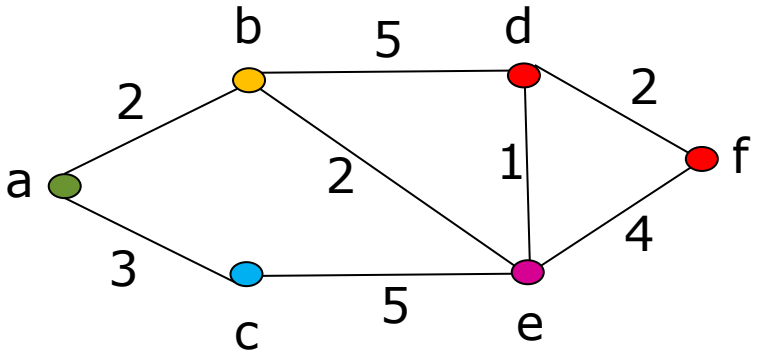
a	**0**					
b	**M**					
c	**M**					
d	**M**					
e	**M**					
f	**M**					



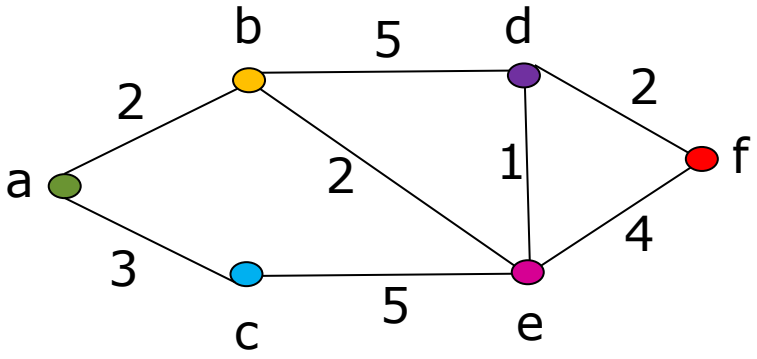
a	**0**	**-**				
b	**M**	**2a**				
c	**M**	**3a**				
d	**M**	**M**				
e	**M**	**M**				
f	**M**	**M**				



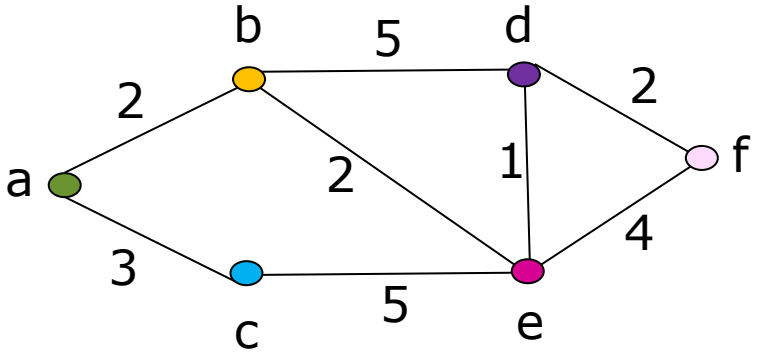
a	**0**	**-**				
b	**M**	**2a**	**-**			
c	**M**	**3a**	**3a**			
d	**M**	**M**	**7ab**			
e	**M**	**M**	**4ab**			
f	**M**	**M**	**M**			

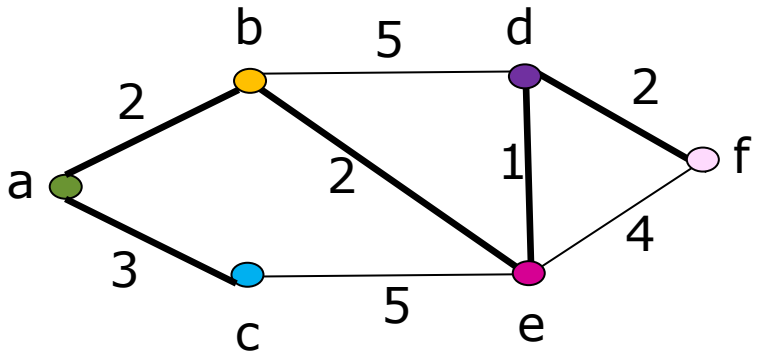


a	**0**	**-**				
b	**M**	**2a**	**-**			
c	**M**	**3a**	**3a**	**-**		
d	**M**	**M**	**7ab**	**7ab**		
e	**M**	**M**	**4ab**	**8ac**		
f	**M**	**M**	**M**	**M**		



a	**0**	**-**				
b	**M**	**2a**	**-**			
c	**M**	**3a**	**3a**	**-**		
d	**M**	**M**	**7ab**	**7ab**	**5abe**	
e	**M**	**M**	**4ab**	**8ac**	**-**	
f	**M**	**M**	**M**	**M**	**8abe**	

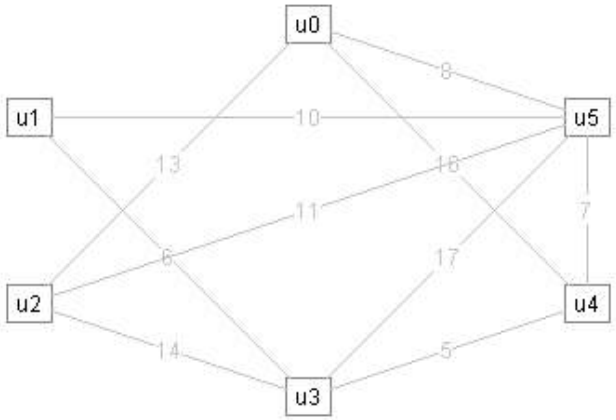




a	**0**	**-**				
b	**M**	**2a**	**-**			
c	**M**	**3a**	**3a**	**-**		
d	**M**	**M**	**7ab**	**7ab**	**5abe**	
e	**M**	**M**	**4ab**	**8ac** ✖	**-**	
f	**M**	**M**	**M**	**M**	**8abe**	**7abed**

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

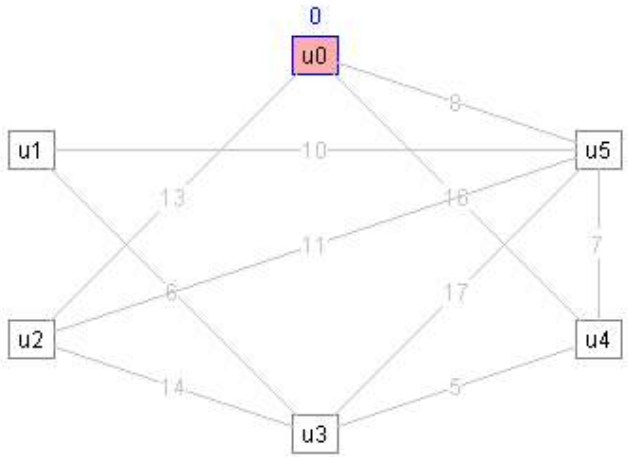
Dijkstra
Java applet demos:

- 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

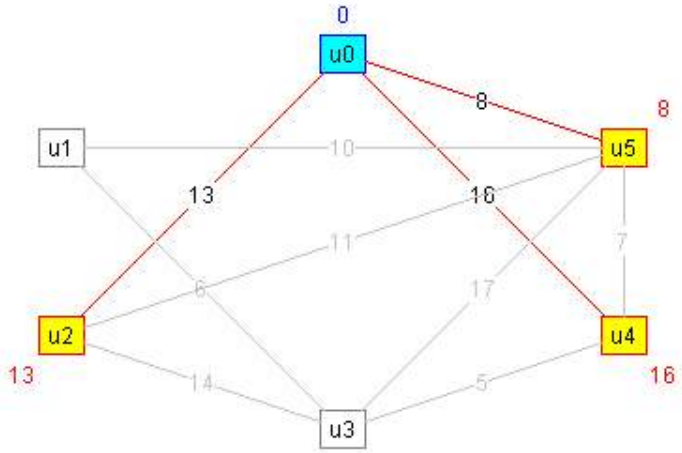
Dijkstra
Java applet demos:

- 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

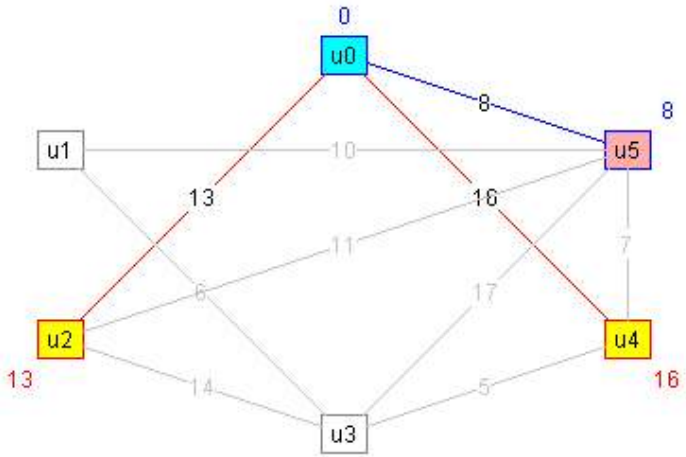
Dijkstra
Java applet demos:

- 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

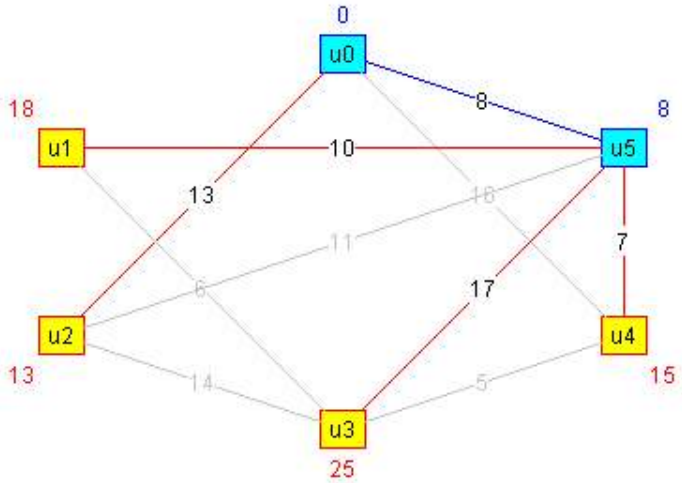
Dijkstra
Java applet demos:

- 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

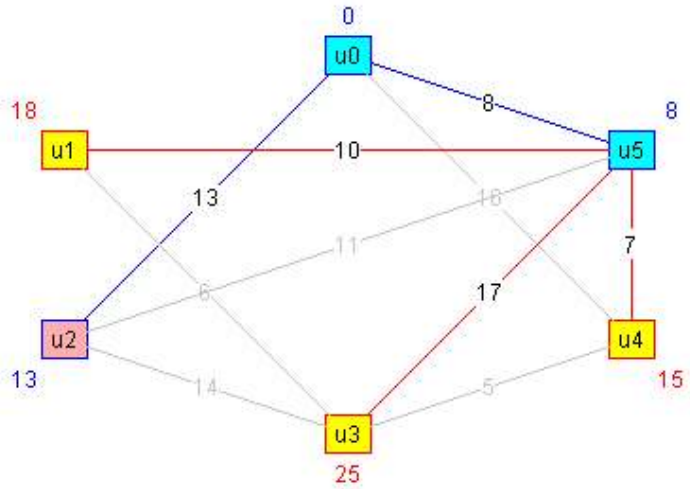
Dijkstra
Java applet demos:

- 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

Dijkstra
Java applet demos:

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

FAQ

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

File Edit View Favorites Tools Help

Back Search Favorites Media

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

Google shortest path Search 28 blocked Check AutoLink AutoFill Options shortest path

Mathematical Programming

Simplex

Twophase

Dijkstra

Prim

Kruskal

Ford-Fulkerson

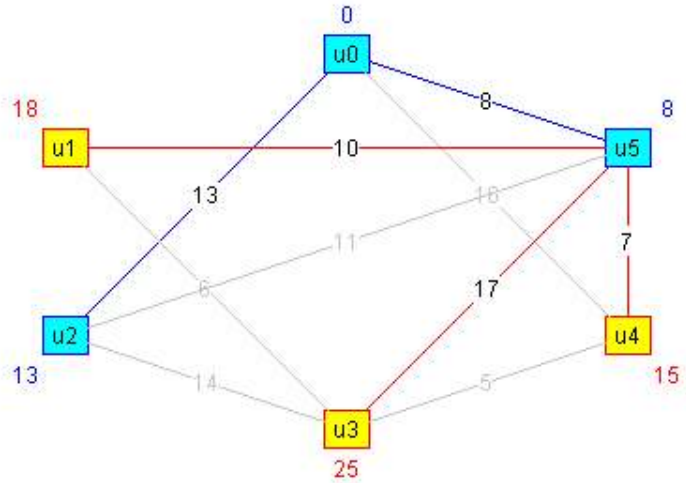
Dijkstra
Java applet demos:

- 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.



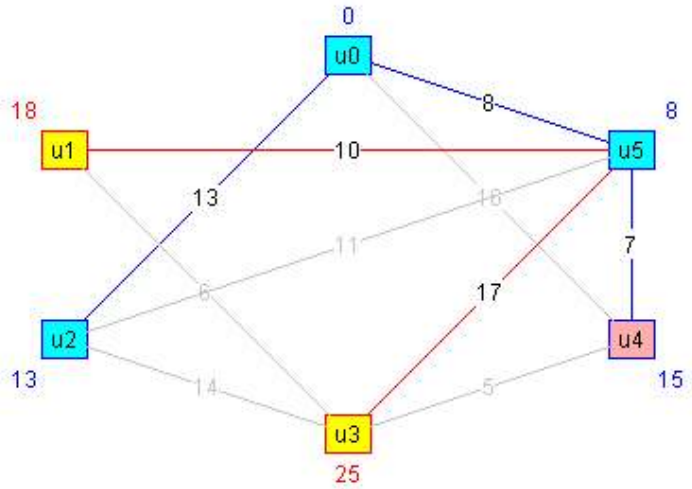
A (6,10) graph

Internet

Start Banu DIRI - Micr... shortest path - ... Java Applet D... Ch-6 Microsoft Power...

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

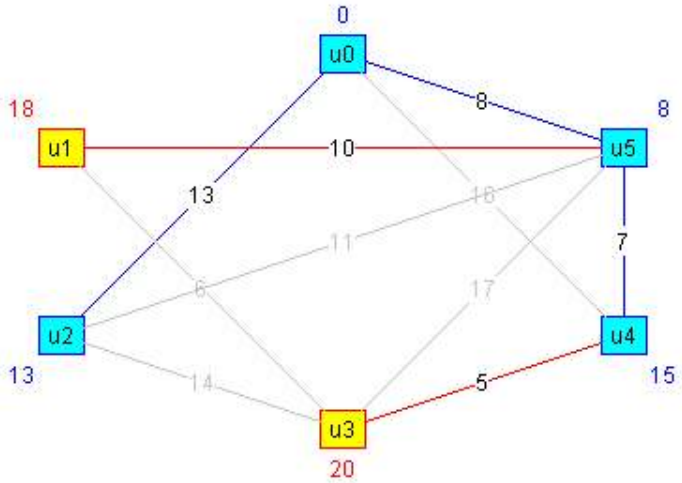
Dijkstra
Java applet demos:

- 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

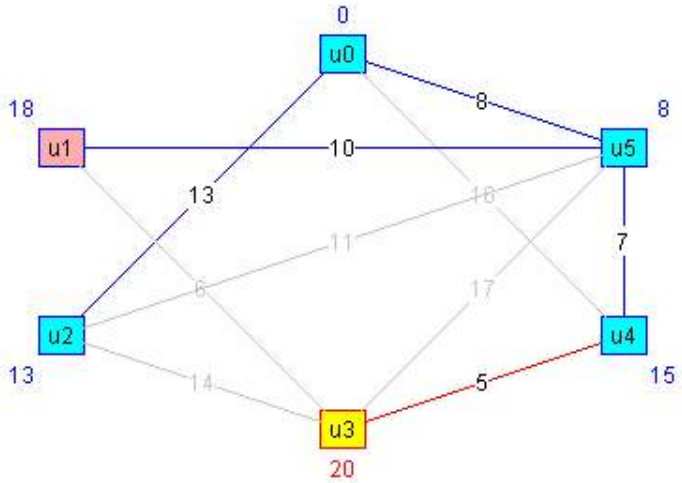
Dijkstra
Java applet demos:

- 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

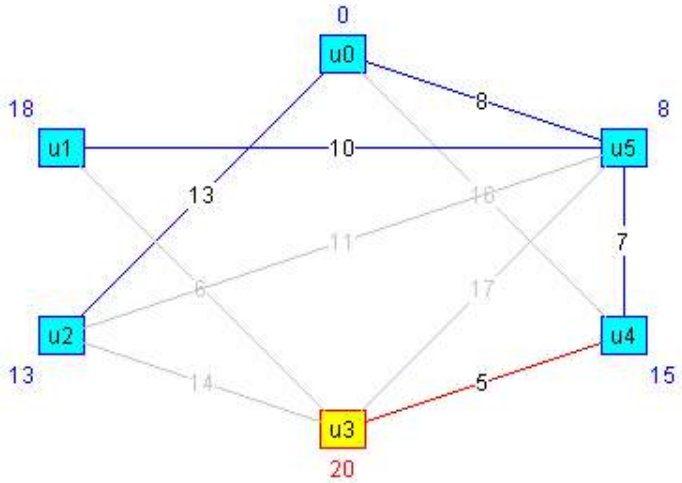
Dijkstra
Java applet demos:

- 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

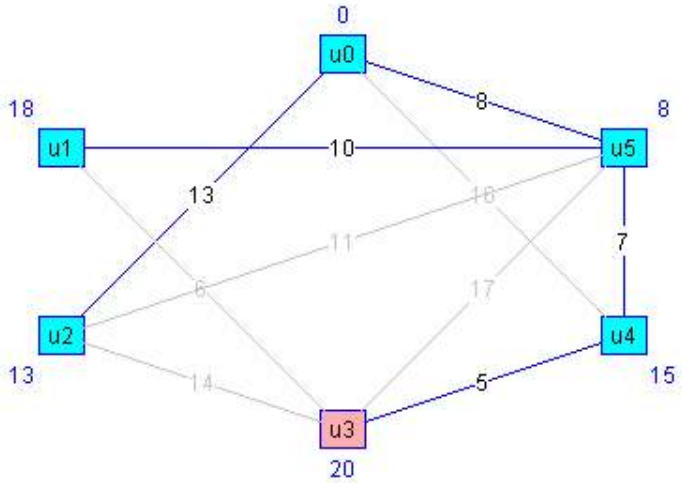
Dijkstra
Java applet demos:

- 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

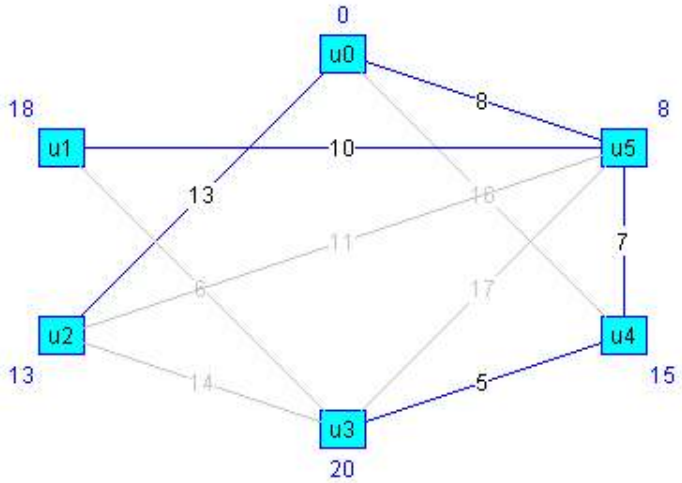
Dijkstra
Java applet demos:

- 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

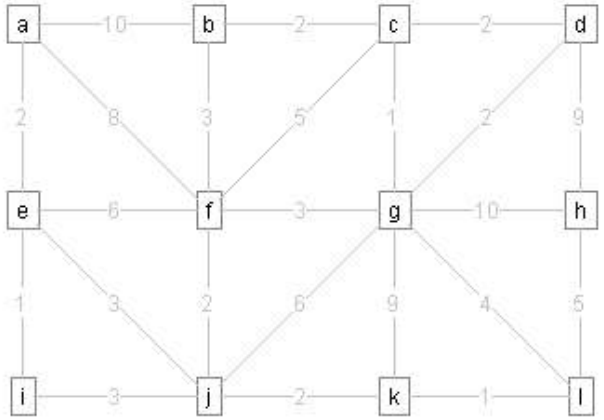
Dijkstra
Java applet demos:

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

FAQ

Java Applet Demos of Dijkstra's Algorithm

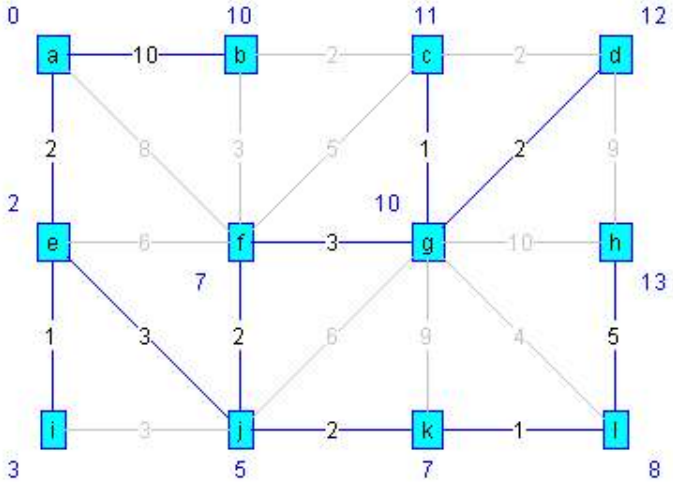
Click on the applet below for several times to find a shortest path from node a.



A (12,23) graph

Java Applet Demos of Dijkstra's Algorithm

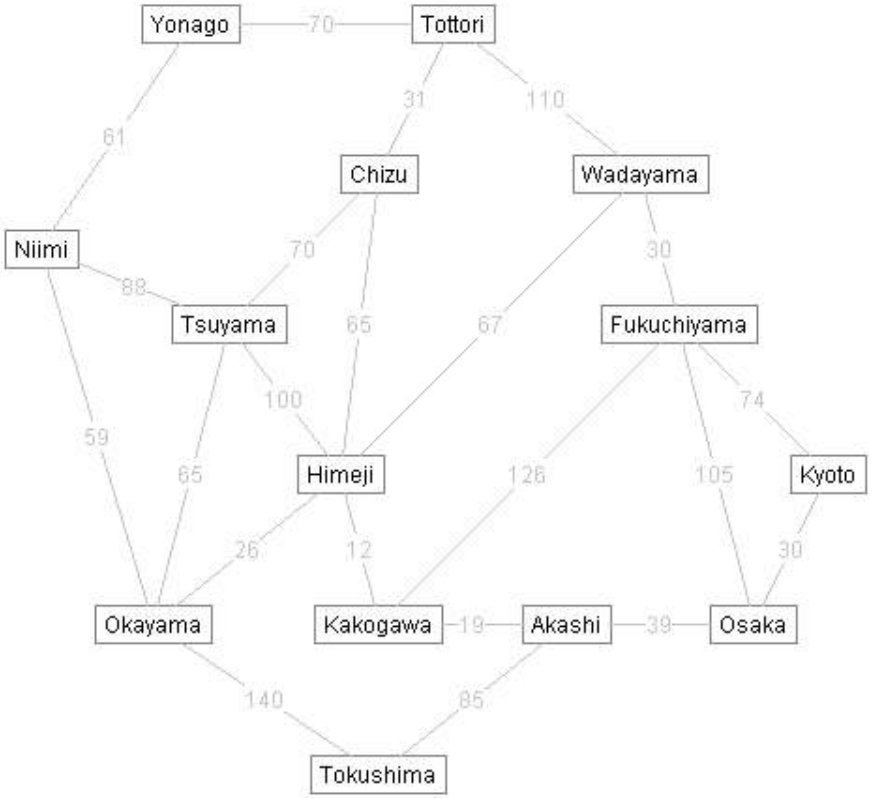
Click on the applet below for several times to find a shortest path from node a.



A (12,23) graph

Java Applet Demos of Dijkstra's Algorithm

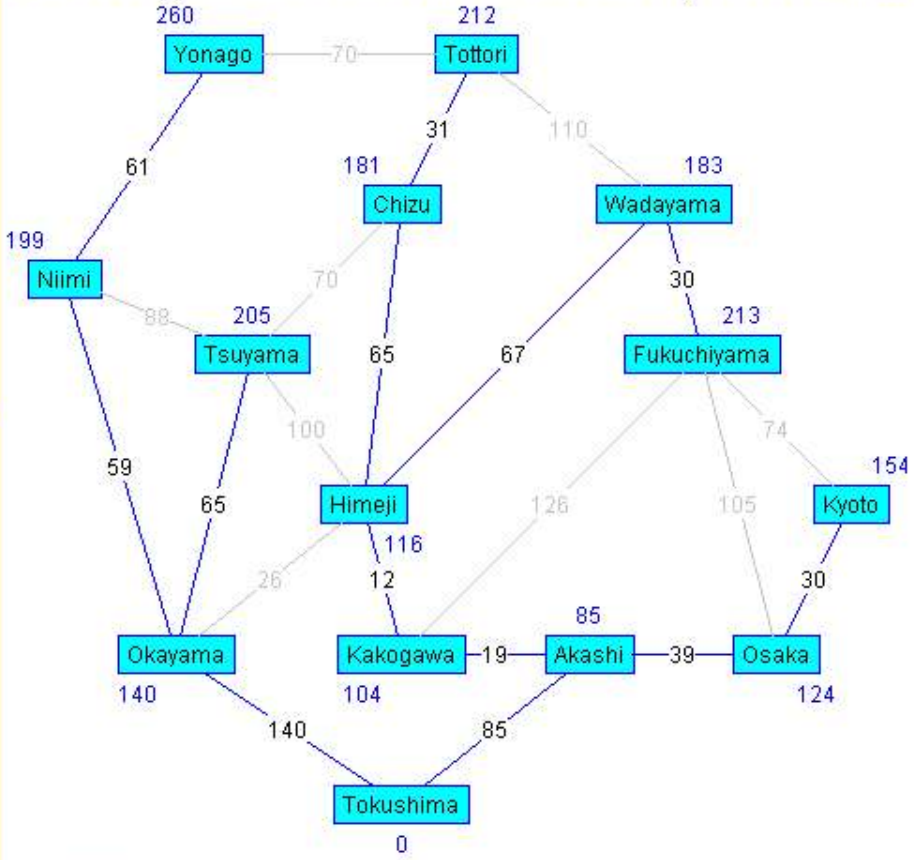
Click on the applet below for several times to find a shortest path from Tokushima.



A (14,22) graph

Java Applet Demos of Dijkstra's Algorithm

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



A (14,22) graph

Graf Modelleri

Farklı alanlarda farklı graf modelleri kullanılır.

Niche Overlap Graf : Eko sistem içerisindeki farklı grubları modellemede kullanılır.

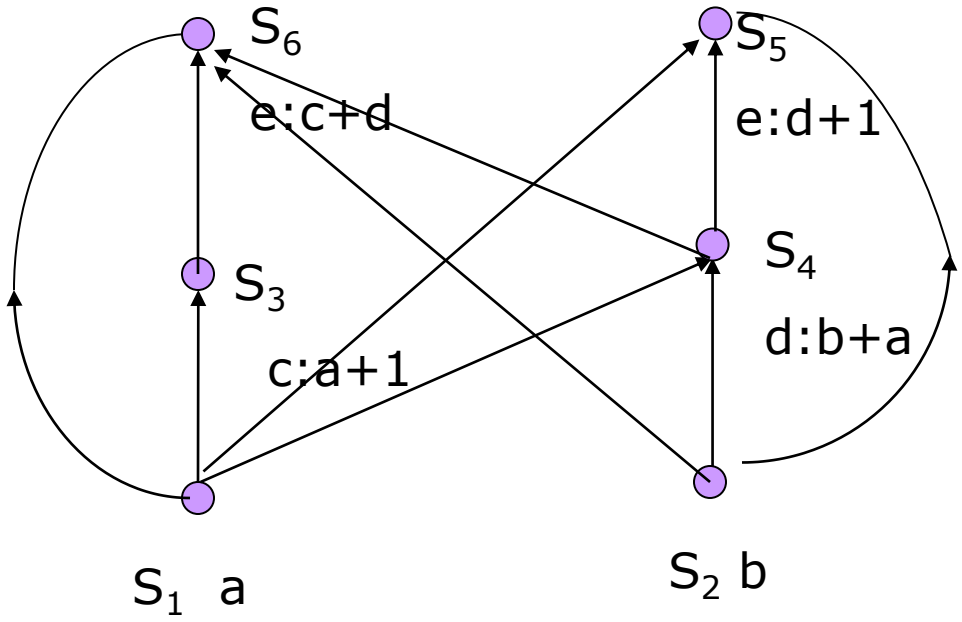
Influence Graf: Grup çalışmalarında, grup içerisindeki kişilerin birbirlerini etkilemesini modellemede kullanılır.

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

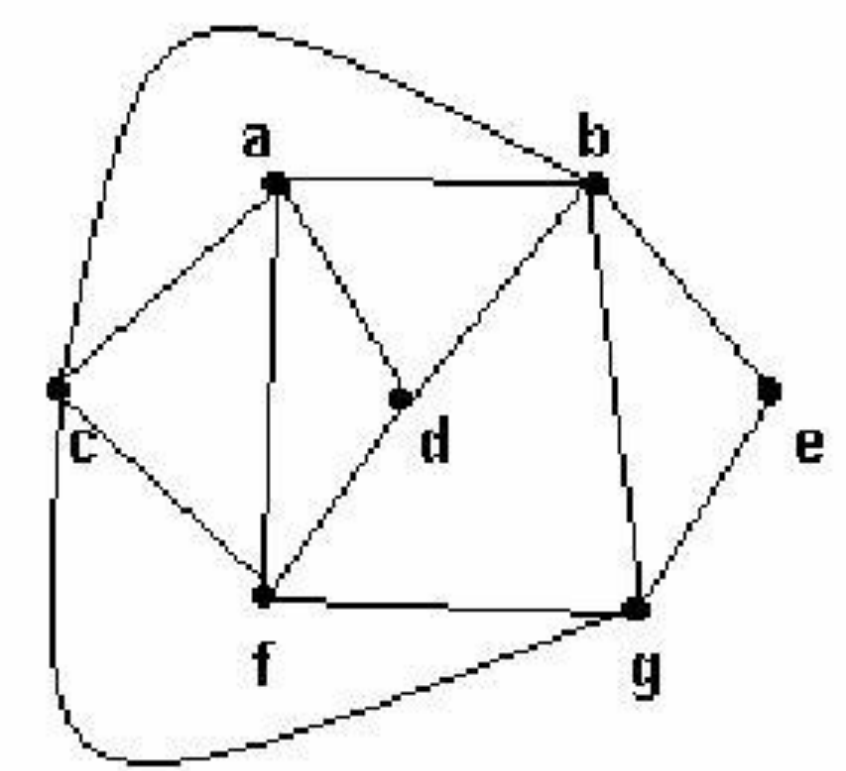
Precedence Graf: Bir işlemin sonucu, kendisinden önce gelen işlemin sonucuna bağlı olarak değişen sistemleri modellemede kullanılır.

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**



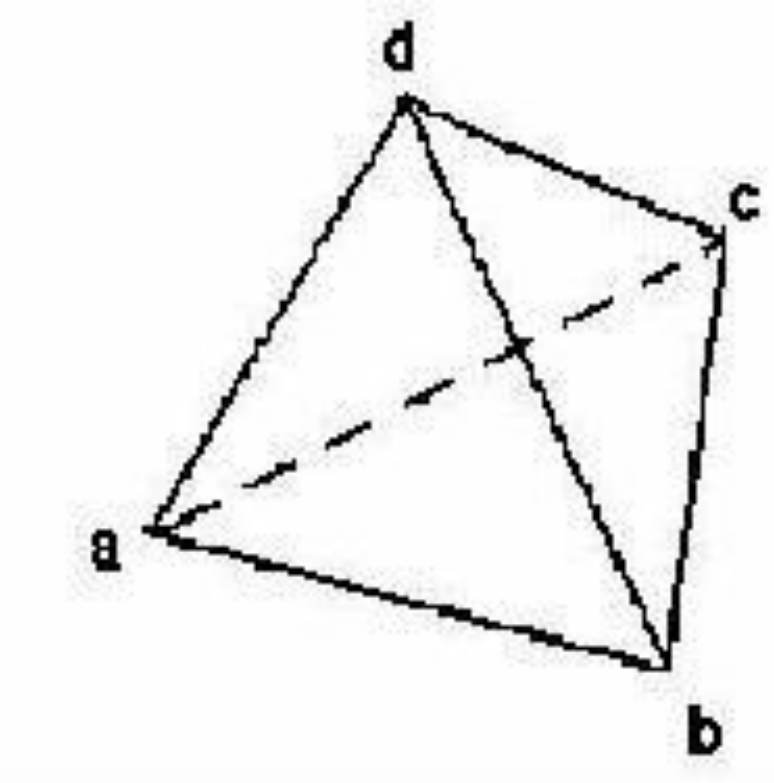
Planar Graflar



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

Euler’in formülü

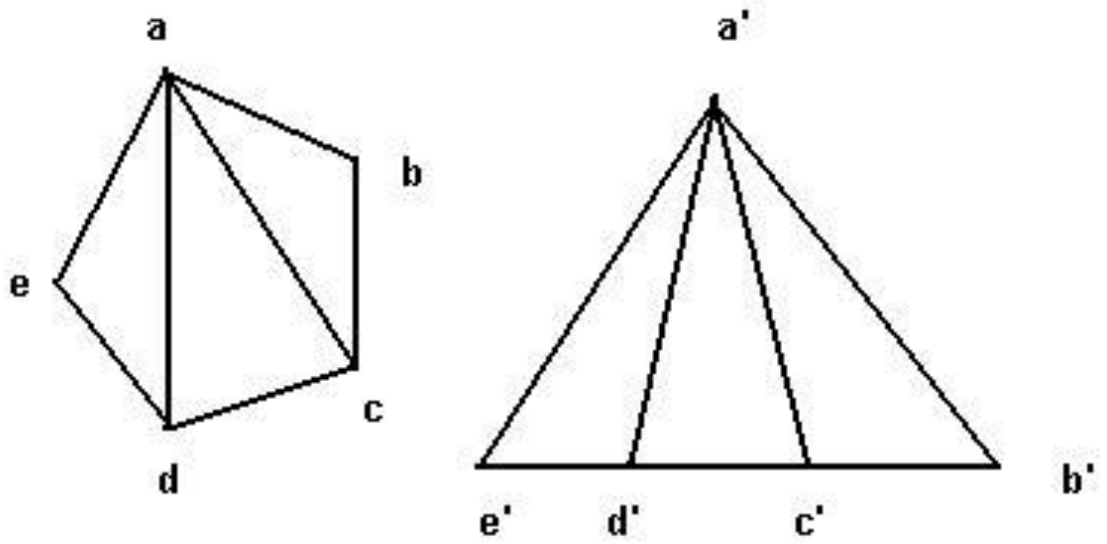
- Eğer G bir *planar* graph is
  - v = düğüm sayısı
  - e = kenar sayısı
  - f = yüzey sayısı
- Öyleyse v – e + f = 2



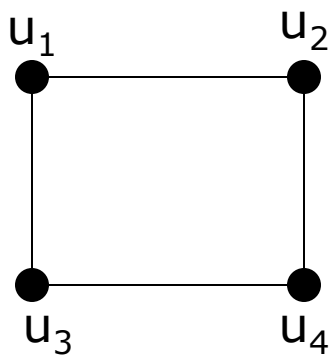
İzomorfik (Isomorphic) Graflar

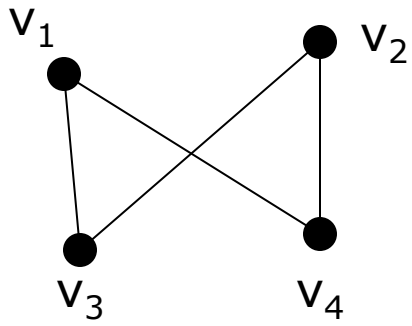
İki grafın izomorfik olup olmadığı nasıl kontrol edilir ?

- 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?

EVET

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

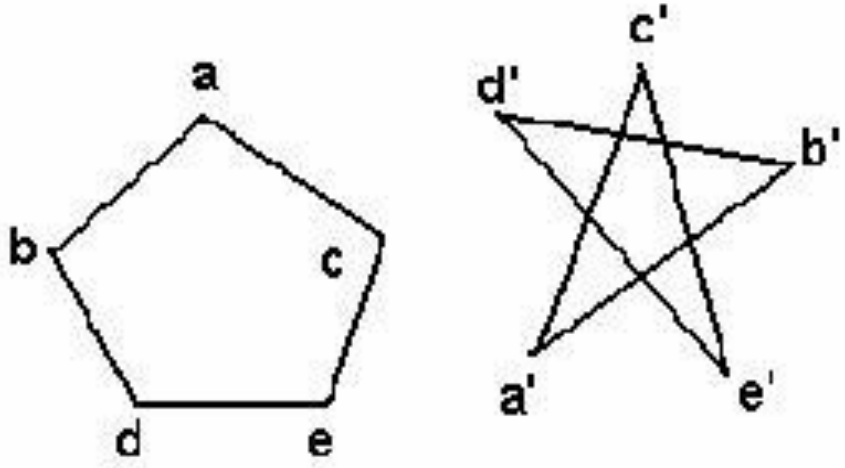
	u1	u2	u3	u4
u1	0	1	1	0
u2	1	0	0	1
u3	1	0	0	1
u4	0	1	1	0

	v1	v2	v3	v4
v1	0	0	1	1
v2	0	0	1	1
v3	1	1	0	0
v4	1	1	0	0

u_2 ve u_4 satır ve sütunlar yerdeğiştiriyor

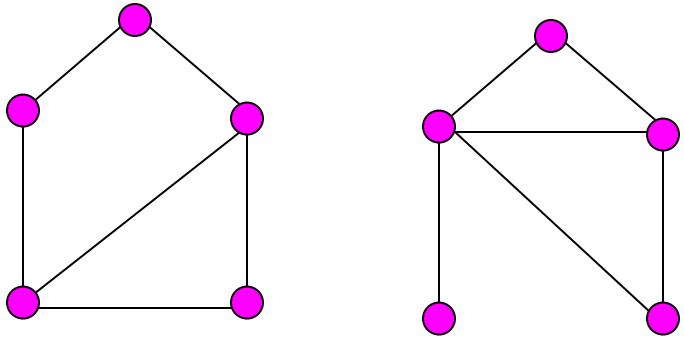
Örnek

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



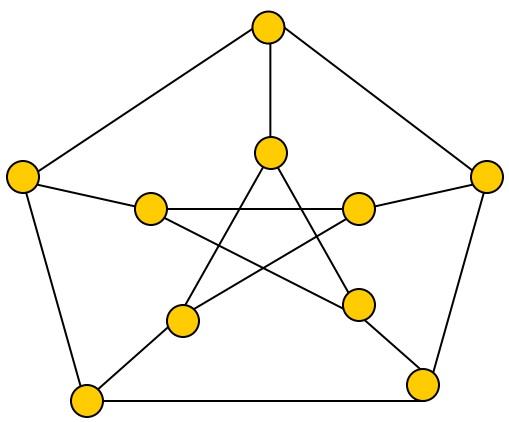
	a	b	c	d	e
a	0	1	1	0	0
b	1	0	0	1	0
c	1	0	0	0	1
d	0	1	0	0	1
e	0	0	1	1	0

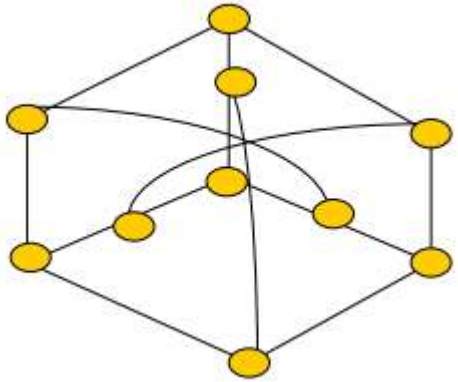
Örnek



Bu iki graf izomorfik midir ?

HAYIR





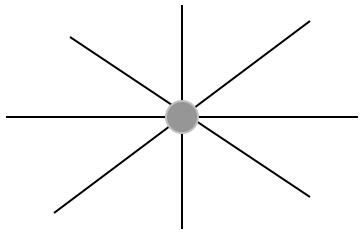
Bu iki graf izomorfik midir ? EVET

Özel Tip Graflar

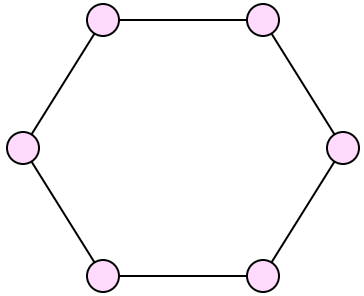
- Özel tip graflar genellikle veri iletişimi ve paralel veri işleme uygulamalarında kullanılır.

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

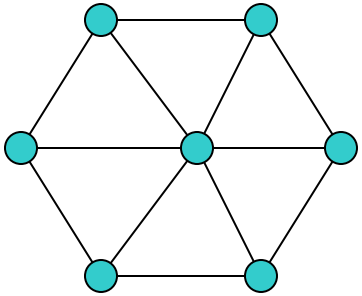
« **Star Topology :** Bütün cihazlar, merkezdeki cihaz üzerinden birbirlerine bağlanırlar. $K_{1,n}$ complete Bipartite Graf kullanılır.



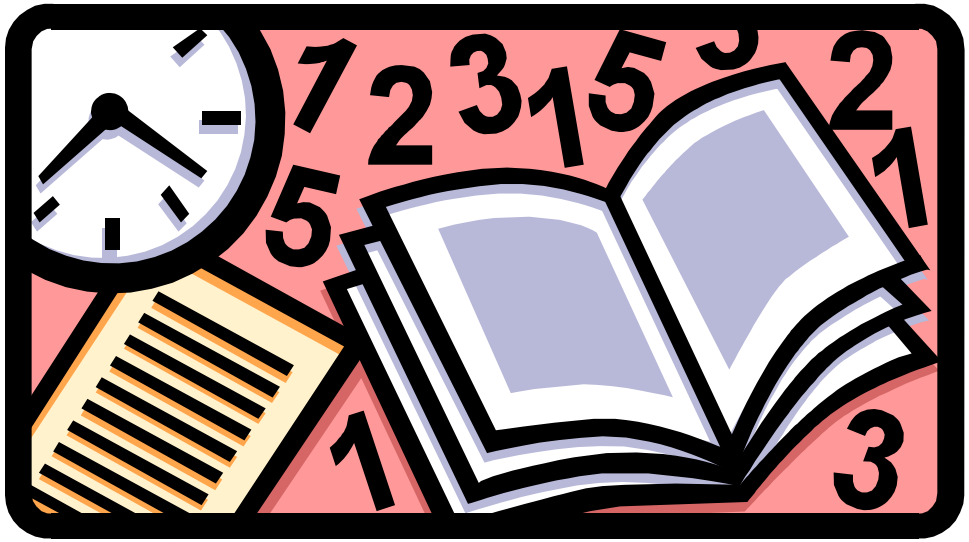
« **Ring Topology :** Bu modelde her cihaz diğer iki farklı cihaz ile birbirine bağlıdır. n-cycles C_n modelidir.



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



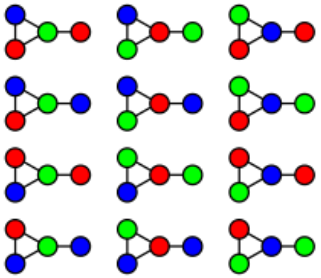
Graf Boyama (Grap Coloring)
Kromatic Polinomlar
(Choromatic Polinomial)

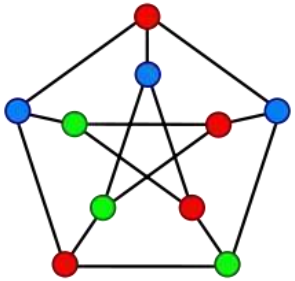


Graf Boyama

- **Düğüm / Köşe / Vertex / Node Boyama**

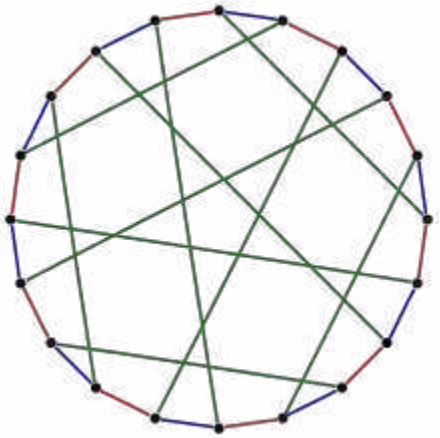
Düğüm boyama birbirine komşu iki düğümün farklı renklerle boyanması problemidir.





- **Kenar / Edge Boyama**

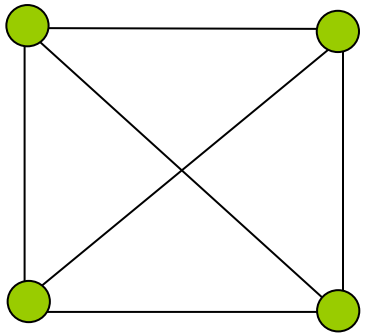
Kenar boyama birbirine komşu iki kenarın Farklı renklerle boyanması problemidir.



Tanım

- Bir **G** grafının herhangi iki komşu düğümüne aynı renk atanmayacak şekilde, grafın her bir düğümüne bir renk atanmasına bir grafın renklendirilmesi denir.

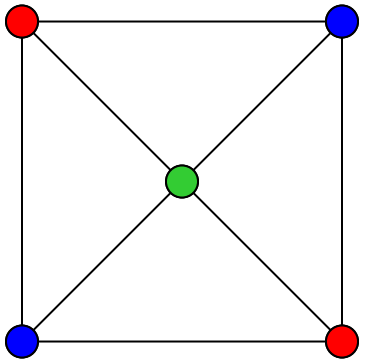
- Bir grafın renk (kromatik) sayısı, grafın renklendirilmesi için gerekli olan **en az renk** sayısıdır. Bir G grafının renk (kromatik) sayısı X(G) ile gösterilir.



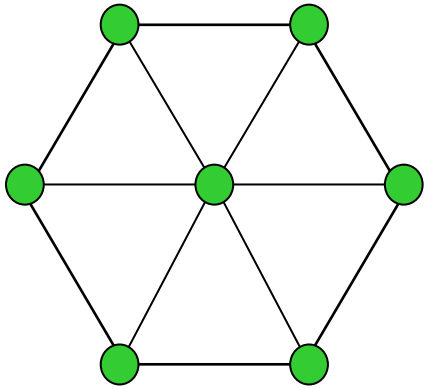
Grafın renk (kromatik) sayısı kaçtır?

X(G) = 3 renk mi?

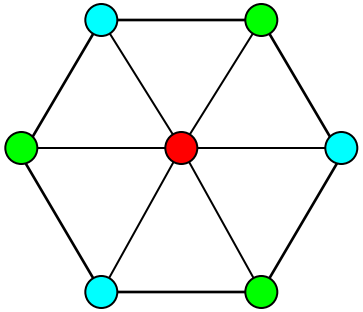
X(G) = 4 renk mi?



X(G) = 3



Grafın renk (kromatik) sayısı kaçtır?



X(G) = 3

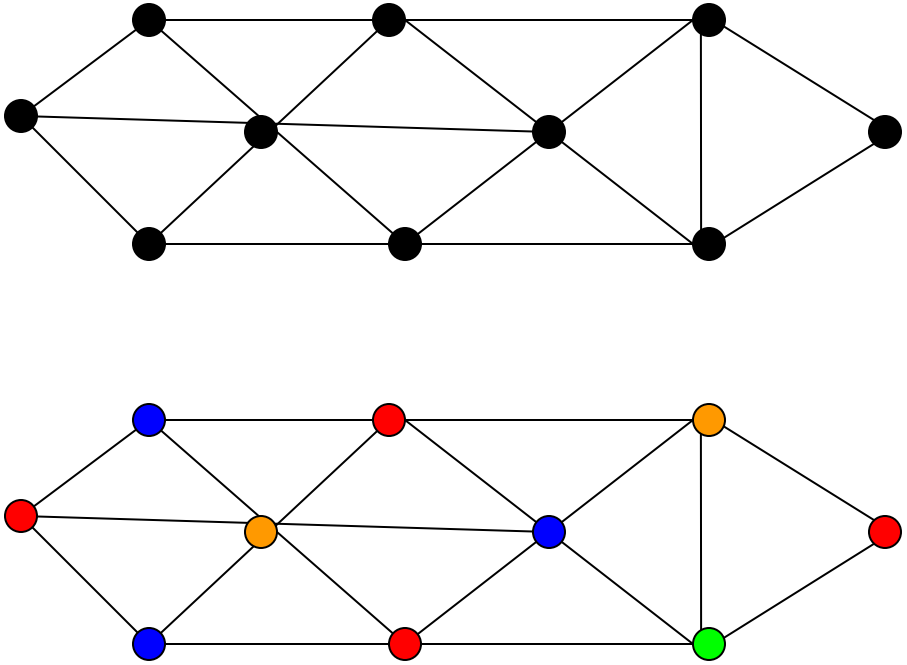
Örnek

- Bir üniversite içerisinde profesör ünvanlı akademisyenlerden oluşan 10 tane kurul olsun
- Bu kurullar haftada bir kez toplanmaktadır
- Bir akademisyen birden fazla kurulda görev alabilir
- Tüm toplantıların en kısa sürede tamamlanması ve akademisyenlerin katılacağı toplantılarda çakışma olmaması istenmektedir

Kaç farklı toplantı oluşturulmalıdır?

Düğümler = Kurullar
Kenarlar = Çakışan akademisyenler
Renkler = Farklı toplantı zamanları

Grafımız aşağıdaki şekildeki gibi olsun.



X(G)=4

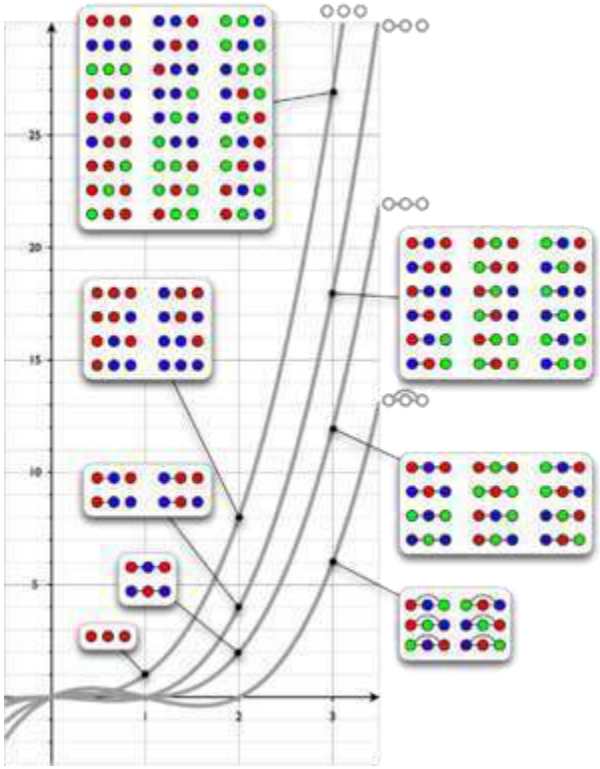
Kromatik Polinomlar

- **Bir G grafına ait köşelerin mümkün olan en az sayıda renkle boyanması için kullanılan analitik çözümlerdir.**

- Herhangi bir k ≥ 0 tamsayısında değerlendirilen bir P (G, x) grafnın polinomu olan P (G, k); G.'nin kromatik polinomu olarak adlandırılır.

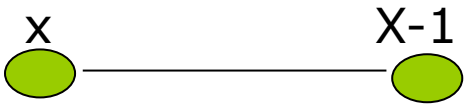
- Elde edilen kromatik polinoma göre k değeri P(G,k) polinomunun değerini P(G,k)>0 yapmalıdır.

Bu değerlere göre;
X(G) = min { k is N: P (G,k) > 0 }
İçin
X(G), en küçük k değeri için elde dilecek sonuca göre boyamada kullanılacak en az renk sayısını verecektir.

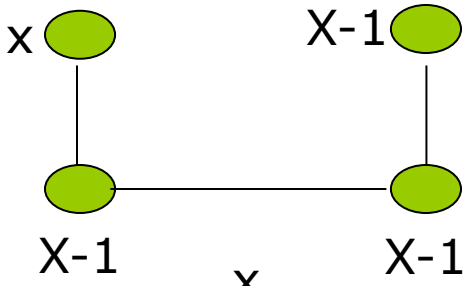


Tanımlar...

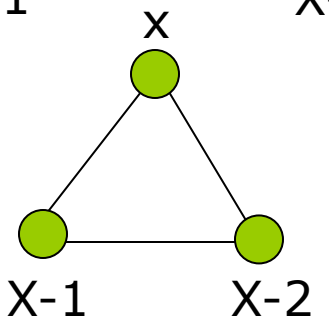
Noktasal Graf **P(G,x)= x**



Çizgisel Graf **P(G,x)= x . (x-1)**



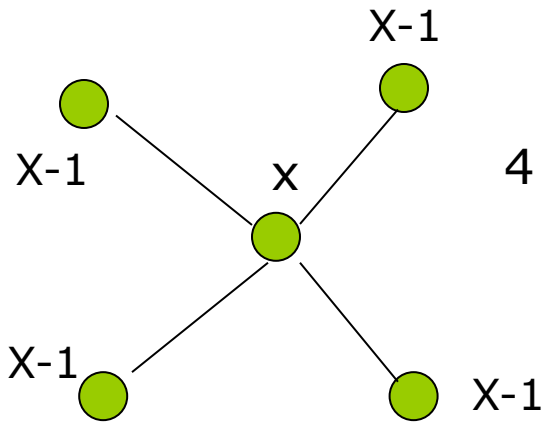
U grafı $\longrightarrow$ $U_n = x(x-1)^{n-1}$



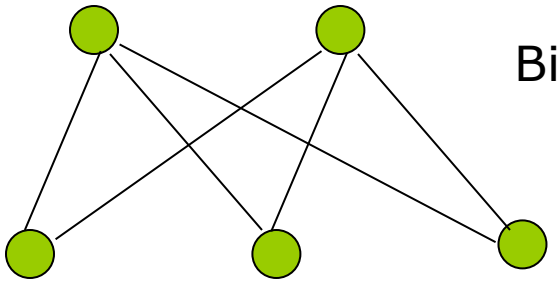
Z Grafı veya K Grafı

$$K_n = \prod_{k=0}^{n-1} (x-k)$$

Tanımlar...



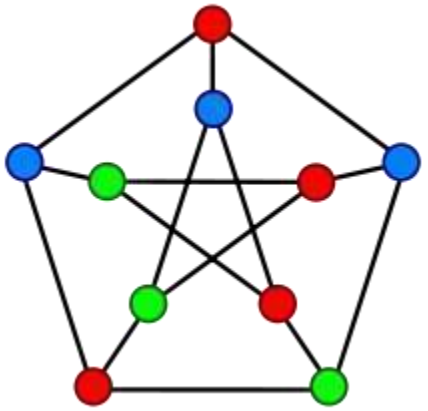
4 - Star Graf → **$P(G,x) = x \cdot (x-1)^4$**



Bipartite Graf → **x =2 renk**

Tanımlar...

Petersen Graph



Chromatic polynomials for certain graphs

Triangle K_3	$x(x-1)(x-2)$
Complete graph K_n	$x(x-1)(x-2)\cdots(x-(n-1))$
Edgeless graph $\overline{K}_n$	x^n
Path graph P_n	$x(x-1)^{n-1}$
Any tree on n vertices	$x(x-1)^{n-1}$
Cycle C_n	$(x-1)^n+(-1)^n(x-1)$
Petersen graph	$x(x-1)(x-2)\left(x^7-12x^6+67x^5-230x^4+529x^3-814x^2+775x-352\right)$

Deletion and Contraction

Bir G grafının kromatik polinomu bulunurken temel graflar cinsinden yazmak karmaşık graflarda çok kullanılan bir çözümdür. Bu işlemi yaparken kenar silme veya ortak düğüm bulma işlemleri gerçekleştirilir. Bir grafta u ile v köşeleri için;

G/uv : Bir graftan u ve v köşelerinin birleştirilmesi veya aralarındaki kenarın çıkarılması ile elde edilen yeni bir graftır.
G –uv: Bir graftan uv kenarının çıkarılması ile elde edilen yeni graftır.
Bu duruma göre

u ve v G grafında bitişikse, uv kenarını kaldırarak elde edilen grafiği kromatik polinomu aşağıdaki gibi hesaplanır

$$P(G, k) = P(G - uv, k) - P(G / uv, k)$$

Kromatik Polinomlar

Bir G grafının kromatik polinomu **P(G)**, G grafını minimum ***k*** renkle renklendirmenin kaç farklı şekilde yapılacağının sayısını verir.

Deletion Contraction Method

$$P_k(G) = P_k(G-e) - P_k(G \setminus e)$$

kenarı silme

Silinen kenara ait düğümleri birleştirme

Kromatik Polinom özellikleri

Elde edilen kromatik polinom aşağıdaki özellikleri sağlamalıdır.

$$P(x) = a_0X^n - a_1X^{n-1} + a_2X^{n-2} + \ldots a_nX^0$$

1- En yüksek dereceli terim a_0=1 olmalıdır

2- Polinomun derecesi olan n düğüm sayısını vermelidir

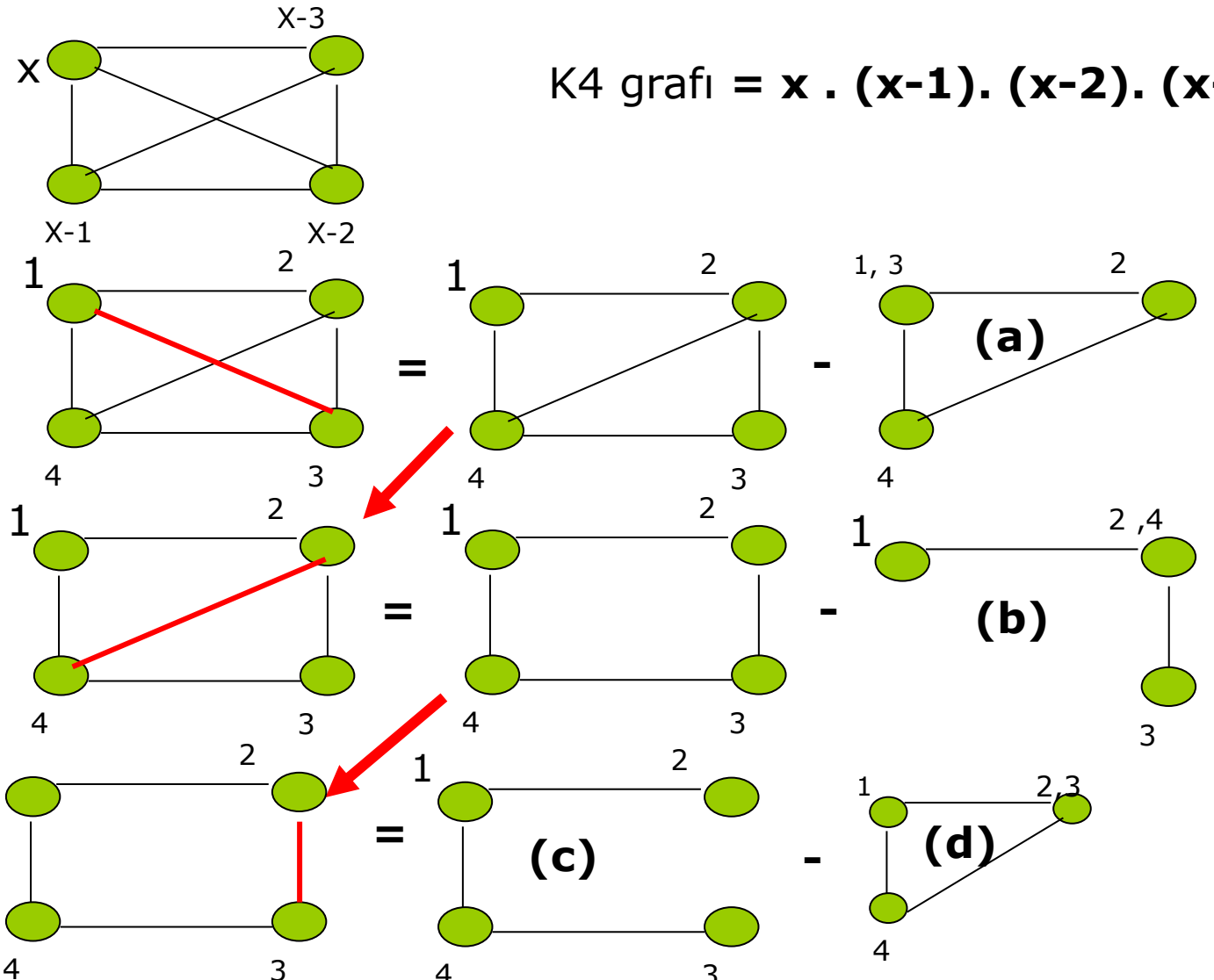
3- Polinomun 2. büyük teriminin katsayısını kenar sayısını vermelidir

4- Polinomda katsayılar toplamı 0 olmalıdır , P(1)=0

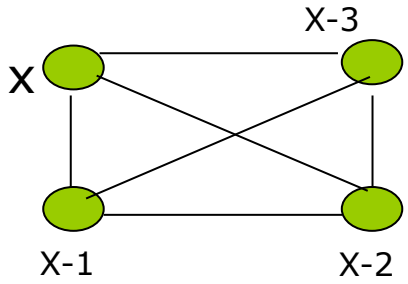
5- Polinom katsayıları +, - , + ., - düzeninde pozifitf ve negatif şeklinde olmalıdır

6- Polinomda sabit terim olmamalıdır. P(0) =0

Örnek Soru:


X-3
X
X-1
X-2
K4 grafı = x . (x-1). (x-2). (x-3)
1
2
4
3
=
1
2
4
3
-
1, 3
2
4
(a)
1
2
4
3
=
1
2
4
3
-
1
2 ,4
3
(b)
1
2
4
3
=
1
2
4
3
(c)
-
1
2,3
4
(d)

Örnek Soru-devam:

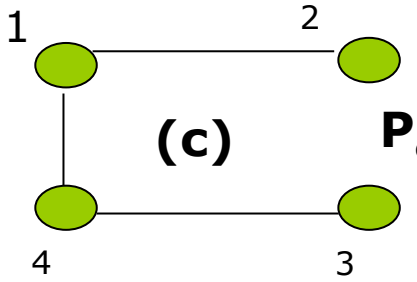


K4 grafı = **x . (x-1). (x-2). (x-3)**

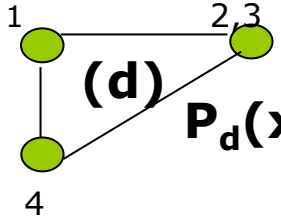
P(x) = $P_c(x)$– $P_d(x)$– $P_b(x)$– $P_a(x)$

$P(x) = x^4 - 6x^3 + 11x^2 - 6x$

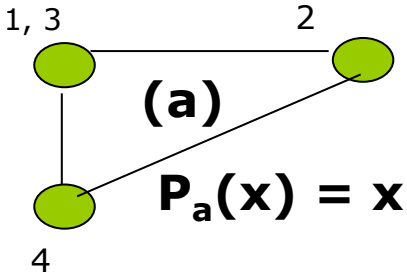
Polinom özelliklerini kontrol edelim!



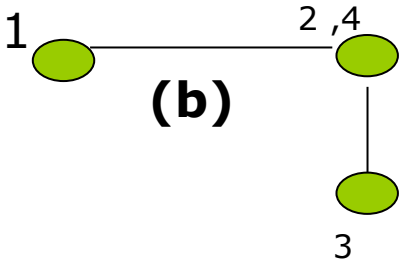
$P_c(x) = x . (x-1)^3$



$P_d(x) = x . (x-1) . (x-2)$



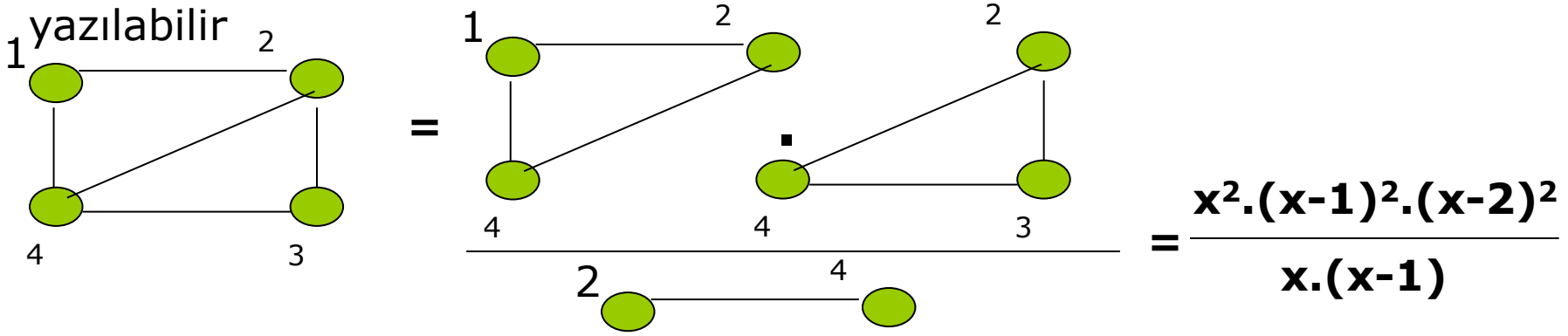
$P_a(x) = x . (x-1) . (x-2)$



$P_b(x) = x . (x-1)^2$

Örnek Soru-2 (Ortak Nokta)

Eğer bir graf,noktasal veya çizgisel bir graf ile ayrılırsa, kromatik polinom çarpım ve bölüm cinsinden bu ortak noktaya göre yazılabilir



$G(x) = x^4 - 5x^3+8x^2 - 4x$

$G(x) = x^4$ (4 düğüm) , $5x^3$(5 kenar) , katsayılar doğru, sabit terim yok

En az kaç renk? En az renkle kaç farklı şekilde boyama yapılır?

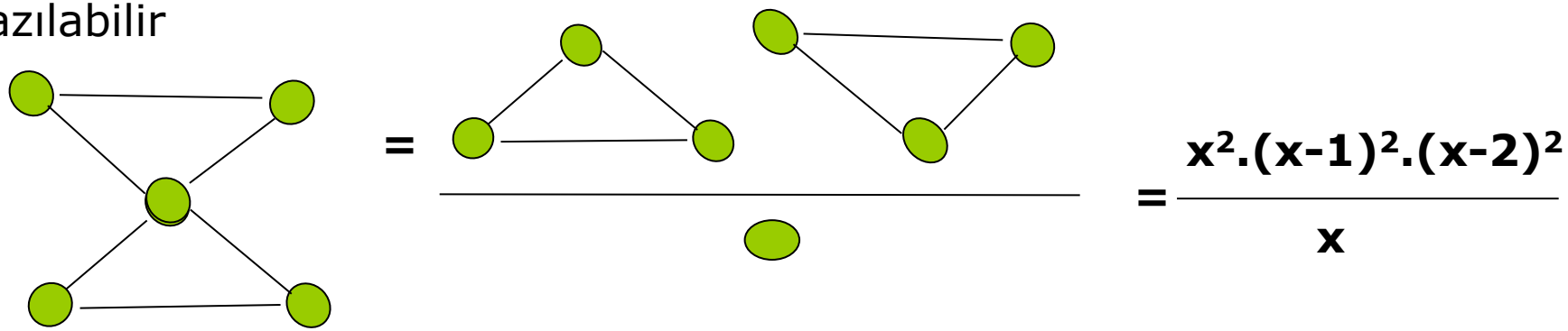
G(P, k) > 0 şartını k bulalım,

G(0) = G(1) = G(2) = 0, G(3) = 6

∴ min 3 renkle, 6 farklı şekilde boyama yapılabilir

Örnek Soru-3 (Ortak Nokta)

Eğer bir graf,noktasal veya çizgisel bir graf ile ayrılırsa, kromatik polinom çarpım ve bölüm cinsinden bu ortak noktaya göre yazılabilir



$$= \frac{}{} = \frac{x^2.(x-1)^2.(x-2)^2}{x}$$

$$G(x) = x^5 - 6x^4 + 13x^3 - 12x^2 + 4x$$

$G(x) = x^5$ (5 düğüm) , $6x^3$(6 kenar) , katsayılar doğru, sabit terim yok

En az kaç renk? En az renkle kaç farklı şekilde boyama yapılır?

G(P, k) > 0 şartını k bulalım,

G(0) = G(1) = G(2) = 0, G(3) = 12

∴ min 3 renkle, 12 farklı şekilde boyama yapılabilir