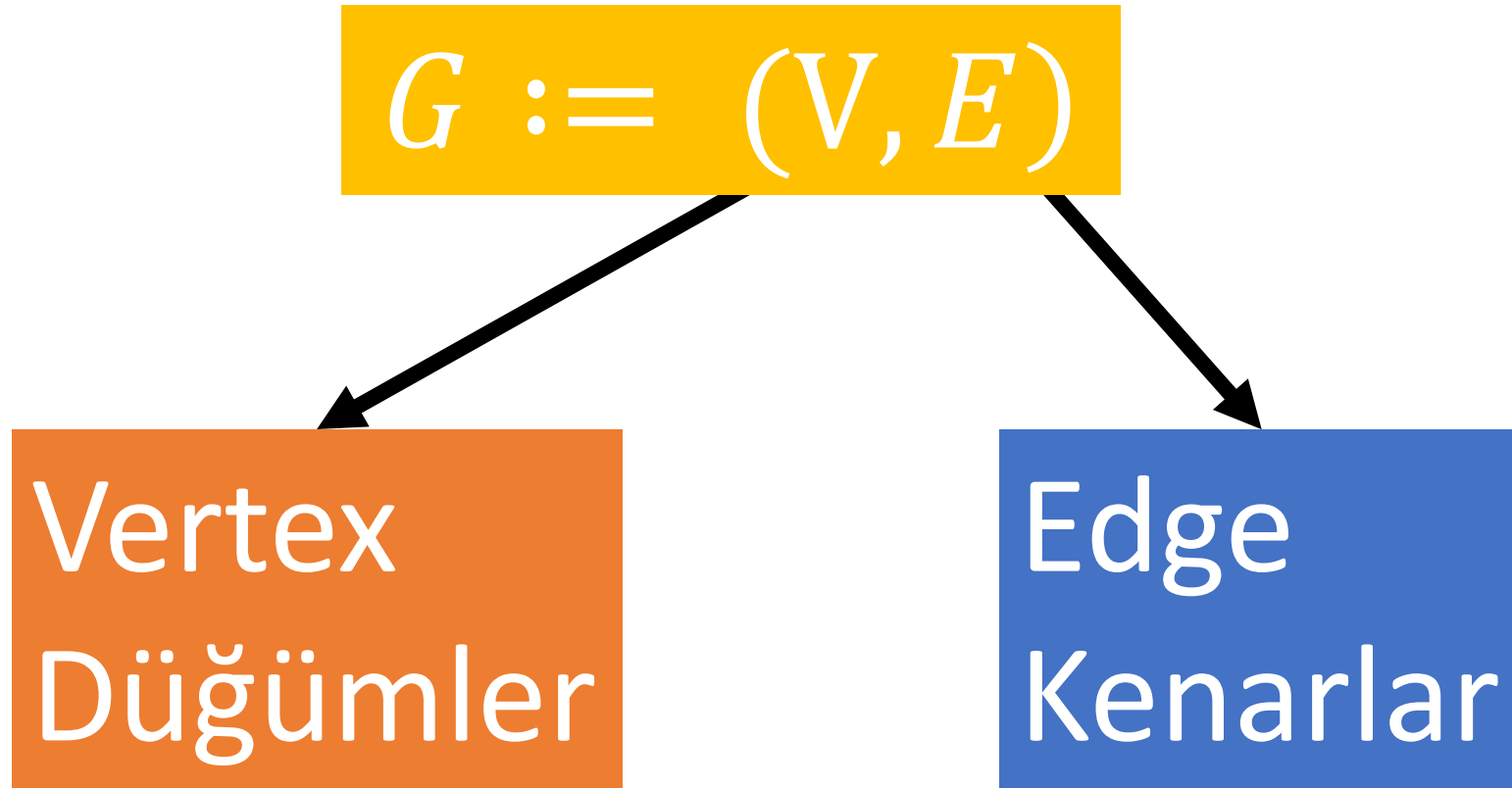


VERİ YAPILARILARI VE ALGORİTMALAR

Graph

Giriş

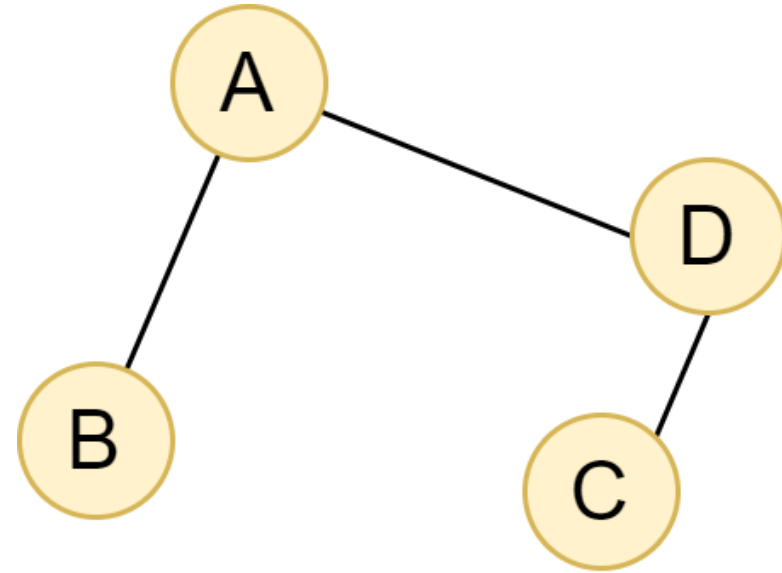
1. Çizge veri yapısı
2. Yönlü ve yönsüz çizgeler
3. Ağırlıklı çizgeler
4. Çizgeler veri yapısı örnekleri
5. Çizgelere ilişkin kavramlar



Çizge

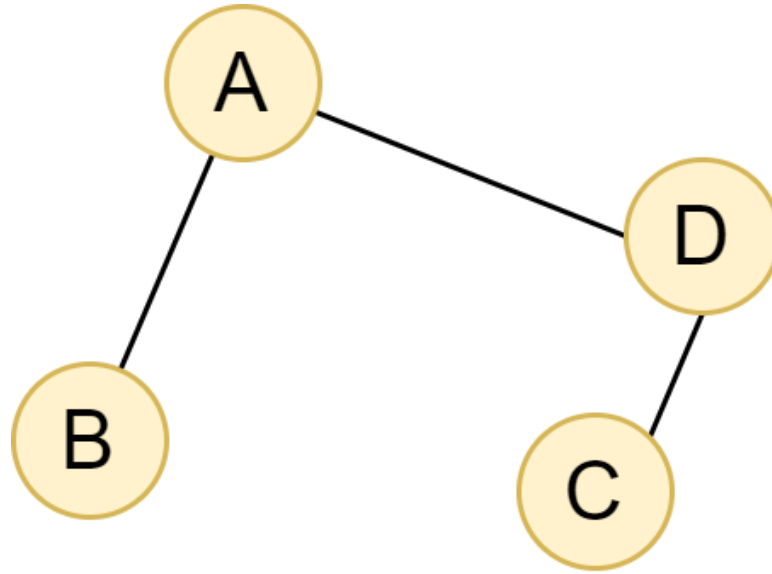
(Graph)

- $V = \{A, B, C, D\}$
- $E = \{(A, B), (A, D), (C, D)\}$



Çizge

(Graph)

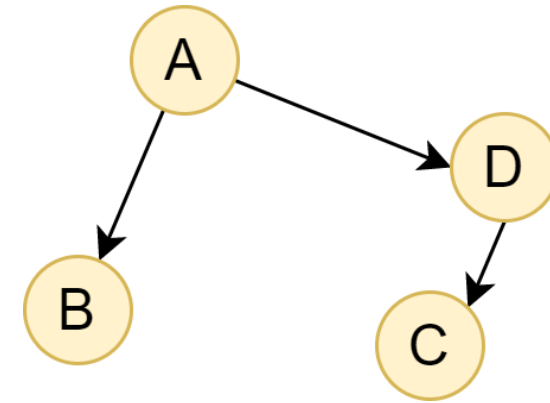


$$G := (\{A, B, C, D\}, \{(A, B), (A, D), (C, D)\})$$

Çizge

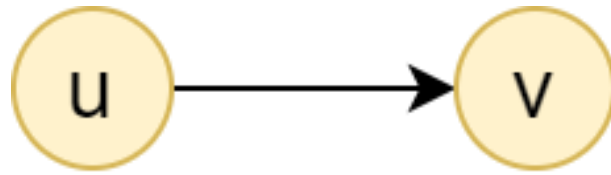
(Graph)

- D ğ mler (vertex) ve kenarları kullanarak baėlantıları temsil etmenin soyut bir yoludur.
- D ğ mler 1 den n 'e kadar etiketlenir.
- m adet kenar bazı d ğ mlere baėlanır.
 - Kenarlar tek y nl  (one-directional, directed) ya da  ift y nl  (bidirectional) olabilir.
- D ğ mler ve kenarlar bazı yardımcı bilgiler i erebilir.



Yönlü Kenar

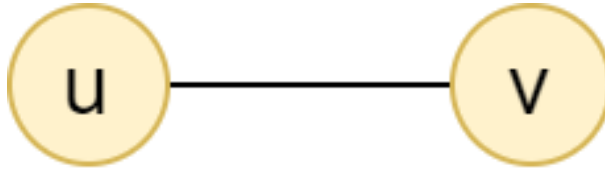
(Directed Edge)



- Bir çift sıralı düğüm (u,v)
- İlk düğüm u , orijini temsil eder.
- İkinci düğüm v , varış noktasını temsil eder.
- Örnek: tek-yön trafik

Yönsüz Kenar

(Undirected Edge)

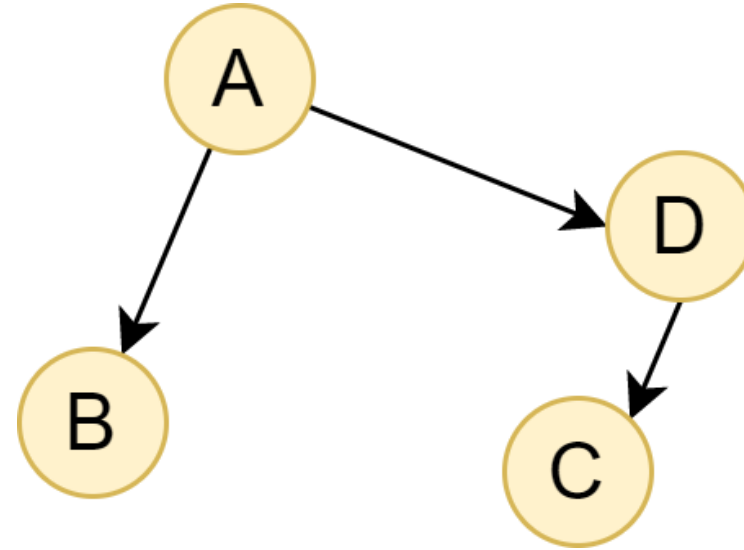


- Bir çift sırasız düğüm (u,v)
- Örnek: Demir yolu

Yönlü Graf

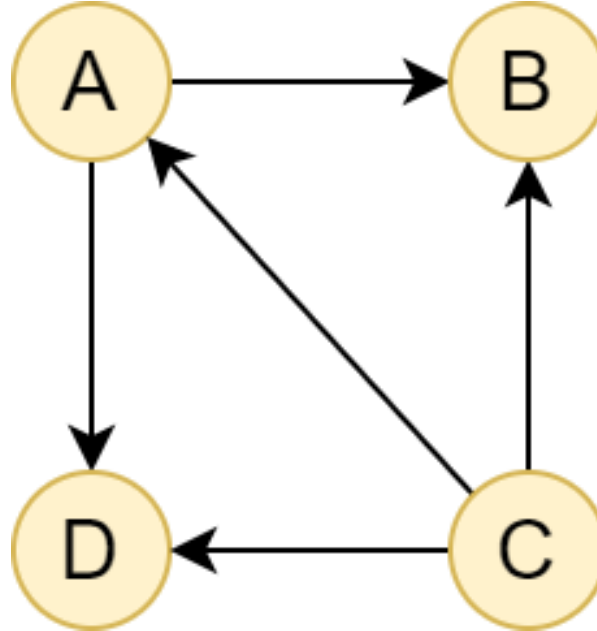
(Directed graph)

- Tüm kenarlar yönlüdür.
- Örnek: Yönlendirme ağı (Root network)



Yönlü çevrimsiz graf

(Directed Acyclic Graph (DAG))

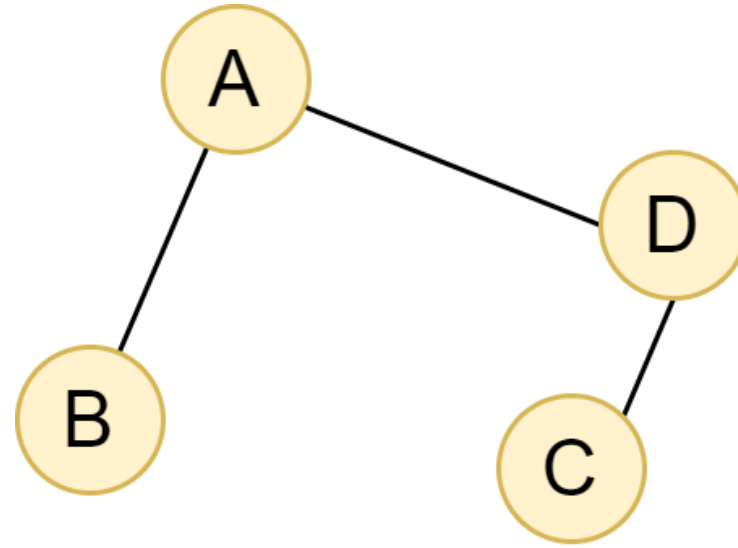


- Bir yönlü çevrimsiz graf (**DAG, directed acyclic graph**) çevrim içermeyen yönlü bir graftır.

Yönsüz Graf

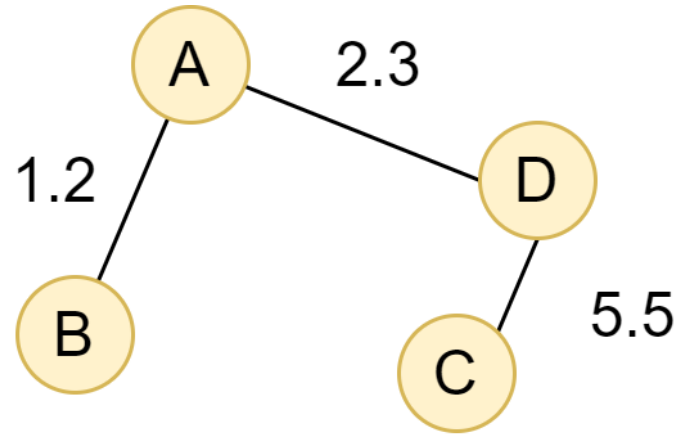
(Directed graph)

- Tüm kenarlar yönsüzdür.
- Örnek: Uçuş ağı (flight network)



Ağırlıklı Graf

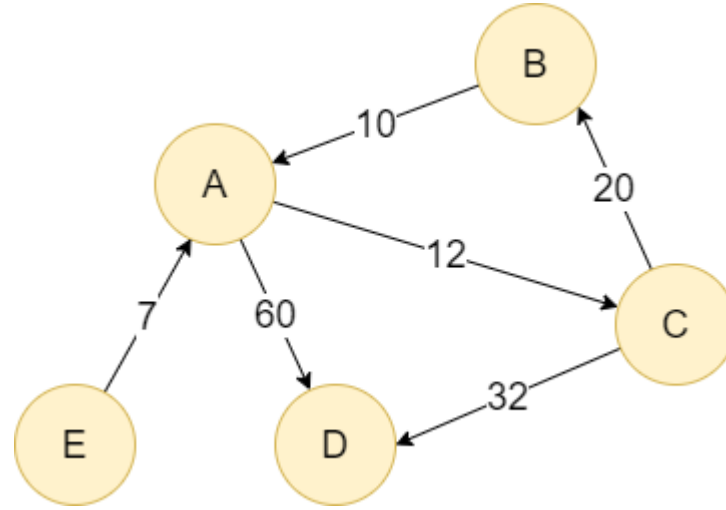
(Weighted Graph)



- Ağırlıklı graflarda iki düğümü birleştiren kenarların belirli bir ağırlığı vardır. Bu ağırlık noktalar arasındaki ilişkiyi tanımlar. Bazen bu ağırlıklar **maliyet** olarak da ifade edilebilir.

Ağırlıklı ve Yönlü Graf

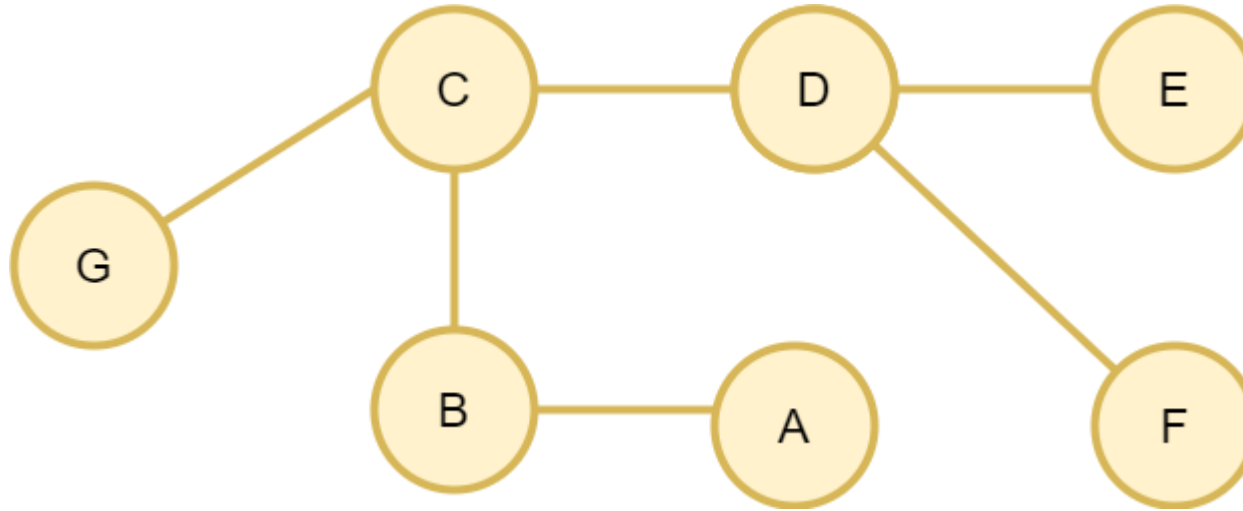
(Weighted DiGraph)



- Ağırlıklı graflarda iki düğümü birleştiren kenarların belirli bir ağırlığı vardır. Bu ağırlık noktalar arasındaki ilişkiyi tanımlar. Bazen bu ağırlıklar **maliyet** olarak da ifade edilebilir. Kenar aynı zamanda **yön** bilgisine de sahipse bu çizge ağırlıklı ve yönlü olarak ifade edilir.

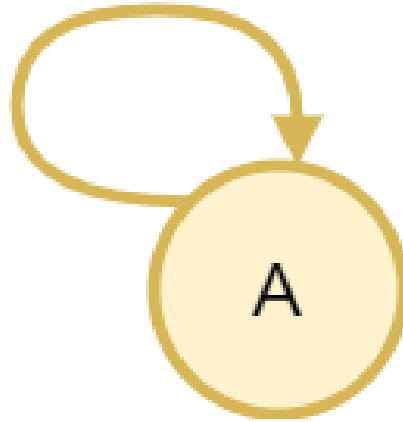
Çizge

- Bir çizgede çevrim yok ise **ağaç (tree)** olarak ifade edilir. Bir ağaç çevrimsel olmayan bağlı bir graftır (**acyclic connected graph**).



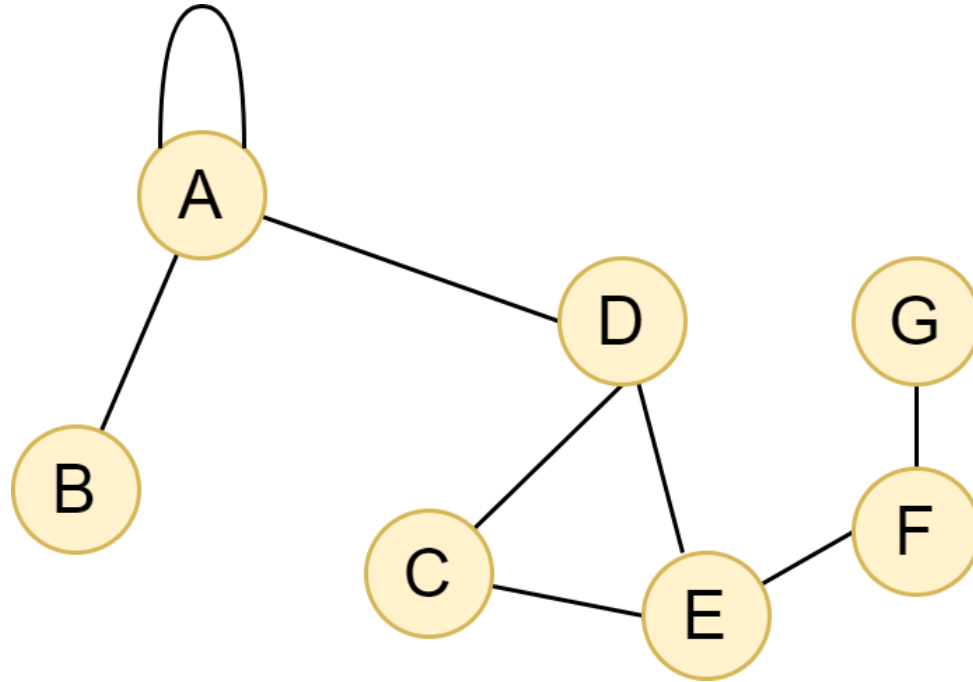
Çizge

- Bir düğümü kendine bağlayan kenara **öz-çevrim (self-loop)** denir.



Döngü

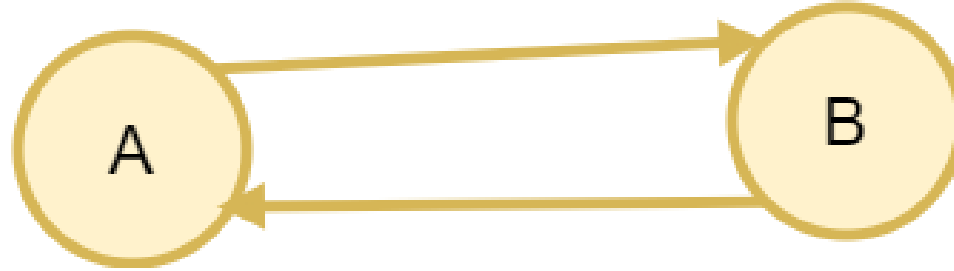
(self-loop)



- Bir yol başladığı düğümden yine başladığı düğüme gidecek şekilde de tanımlanabilir. Bu çoğu zaman bir döngü (self-loop) olarak adlandırılır ve maliyeti genellikle 0 (sıfır) olarak tanımlanır.

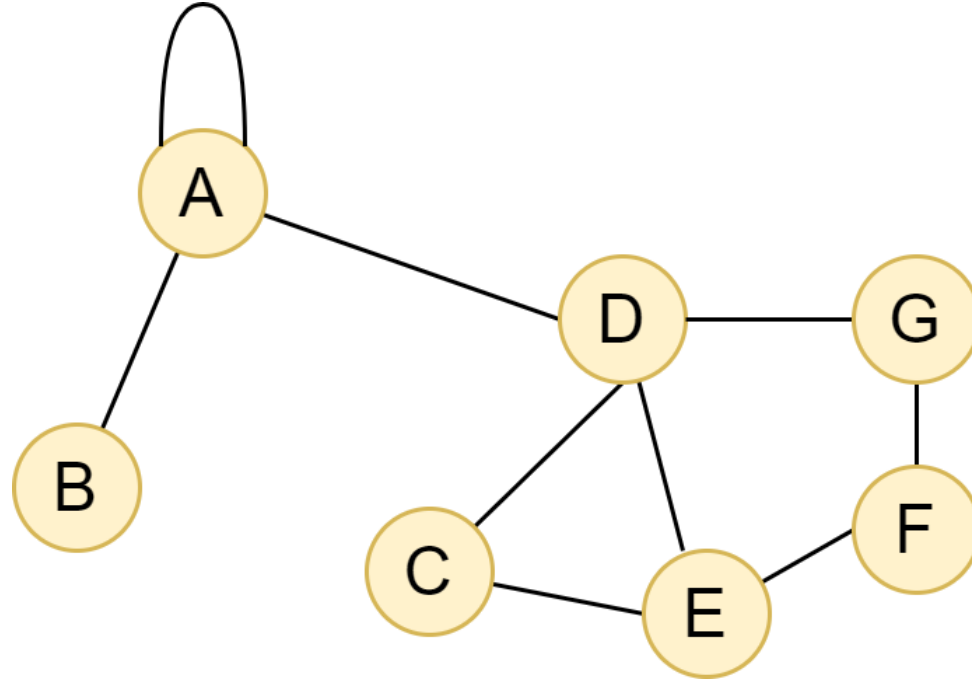
Çizge

- İki kenar aynı düğümleri bağlıyor ise paraleldir.



Çevrim

(cycle)



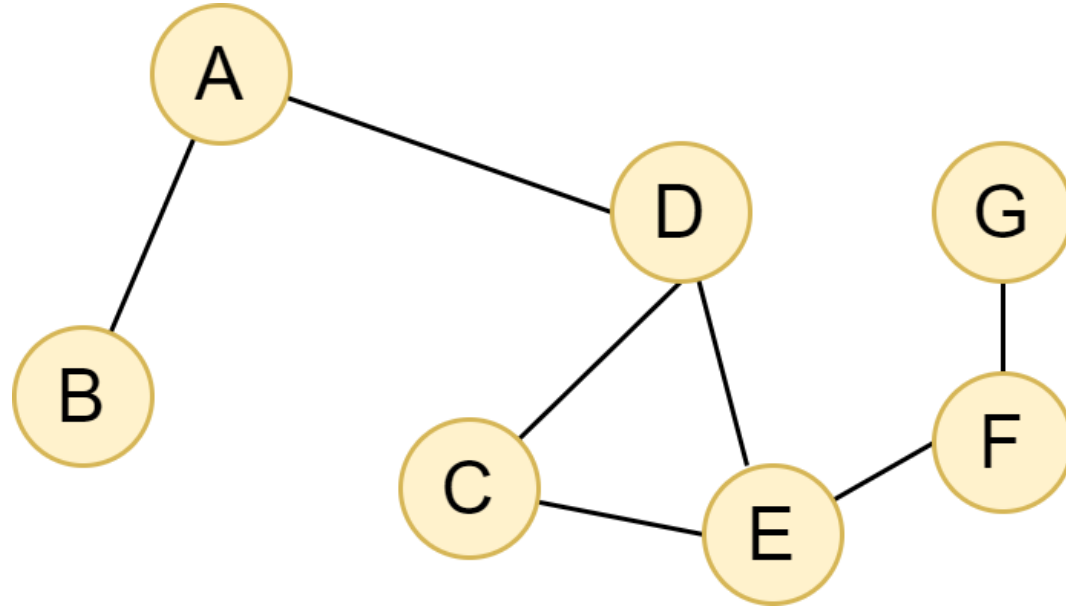
Çevrim

- $D > C > E > D$
- $G > F > E > D > G$
- $C > D > E > C$

- Çevrim yolun başladığı düğümde bitmesidir.
- Çevrim sırasında tekrar eden bir düğüm ya da kenar yok ise ilgili ifade basit çevrim (**simple cycle**) olarak ifade edilir.

Yol

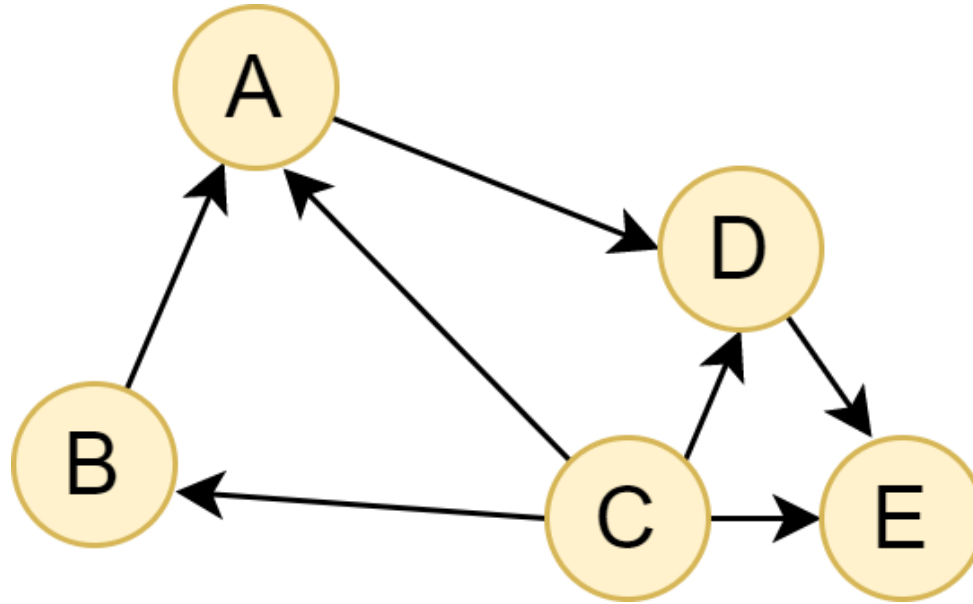
(Path)



- Graf üzerinde bir yol tanımı, ilk düğümden başlamak üzere; yol üzerindeki son düğüme ulaşıncaya kadar olan maliyet şeklinde tanımlanabilir.
- Yol üzerinde tekrar eden düğüm yok ise bu basit yol (**simple path**) şeklinde ifade edilir.

Düğüm derecesi

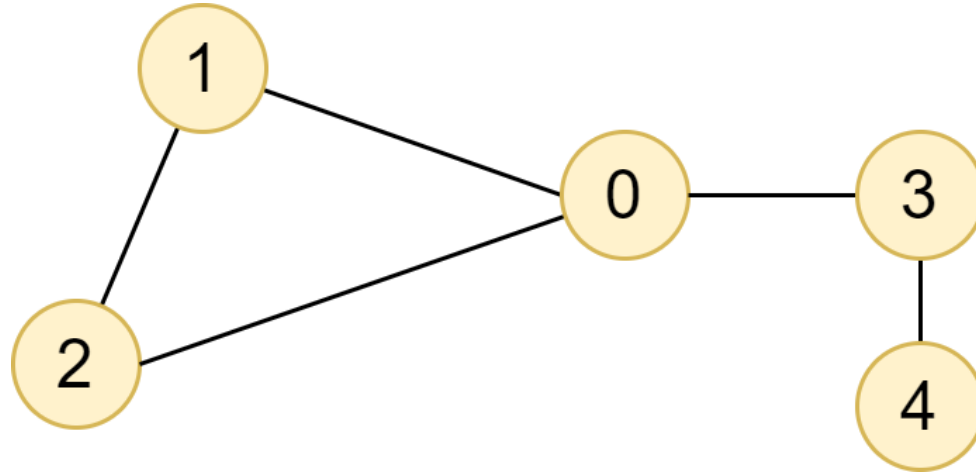
(Degree)



- Düğüm derecesi, düğümün sahip olduğu kenar sayısını ifade etmek üzere kullanılır. Yönlü graflar için bu derece **indeg** (giren kenar sayısı) ve **outdeg** (çıkan kenar sayısı) şeklinde ifade edilebilir.

Güçlü Bağlı Çizge

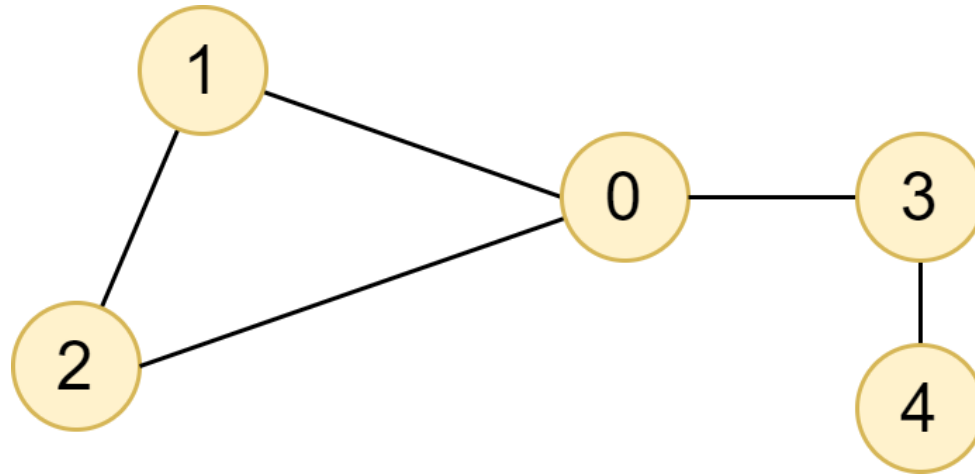
(Strongly Connected Graph)



- Yönsüz bir graf, eğer bir düğümden tüm düğümlere ulaşacak şekilde bir yola sahipse **güçlü bağlı/bağlantılı graf** (strongly connected graph) olarak ifade edilir.

Güçlü Bağlı Çizge

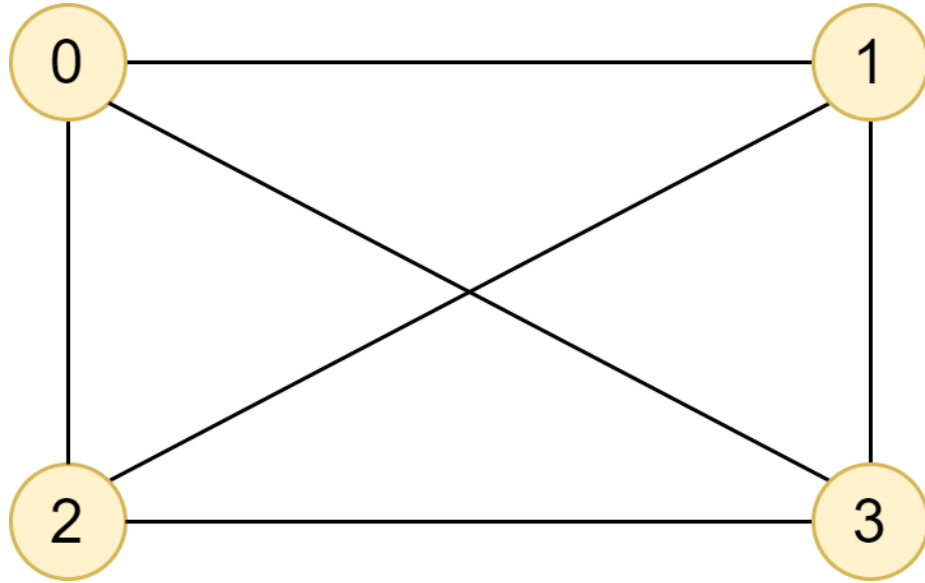
(Strongly Connected Graph)



- Bir graf güçlü bağlantılı değilse bu **zayıf bağlı** (weakly connected graph) olarak ifade edilir.

Tam Çizge

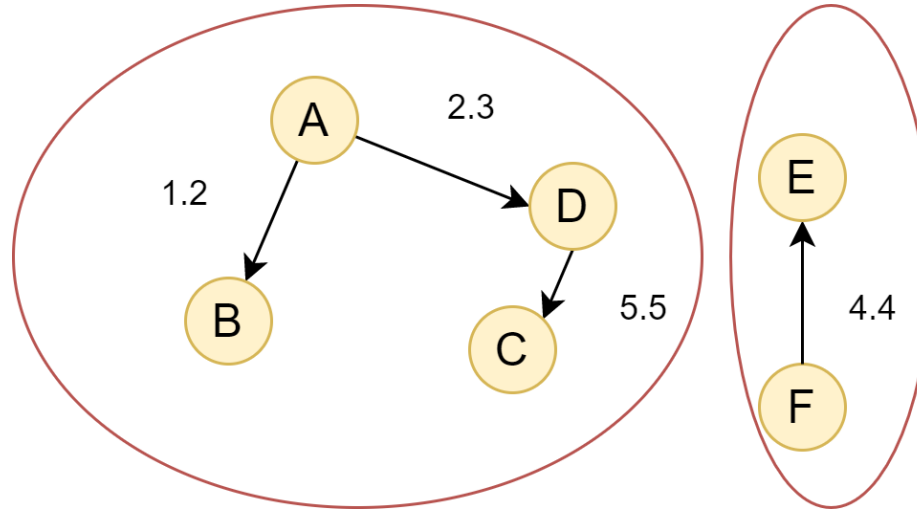
(Complete Graph)



- Her bir düğüm bir diğerine komşu olmalıdır. Tek bir adımda bir düğümden bir başka düğüme gidilebilmelidir.
- n düğüm için $n(n - 1)/2$ kenar bulunur.

Bileşen

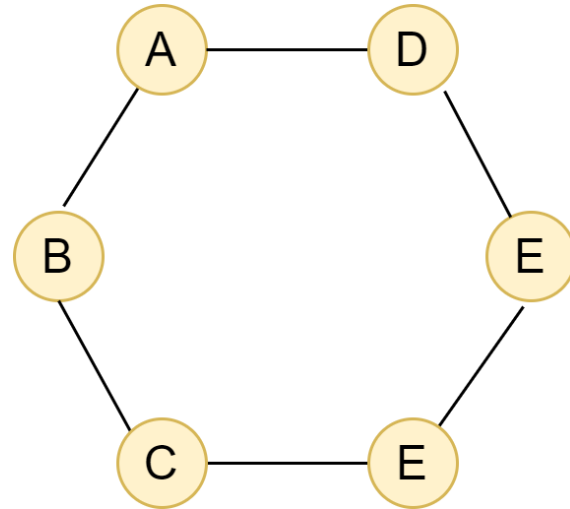
(Component)



- Bağlı olmayan graflardaki her bir ada, ya da ayırık küme, bir bileşen (component) olarak ifade edilir.

Düzenli Çizge

(Regular Graph)

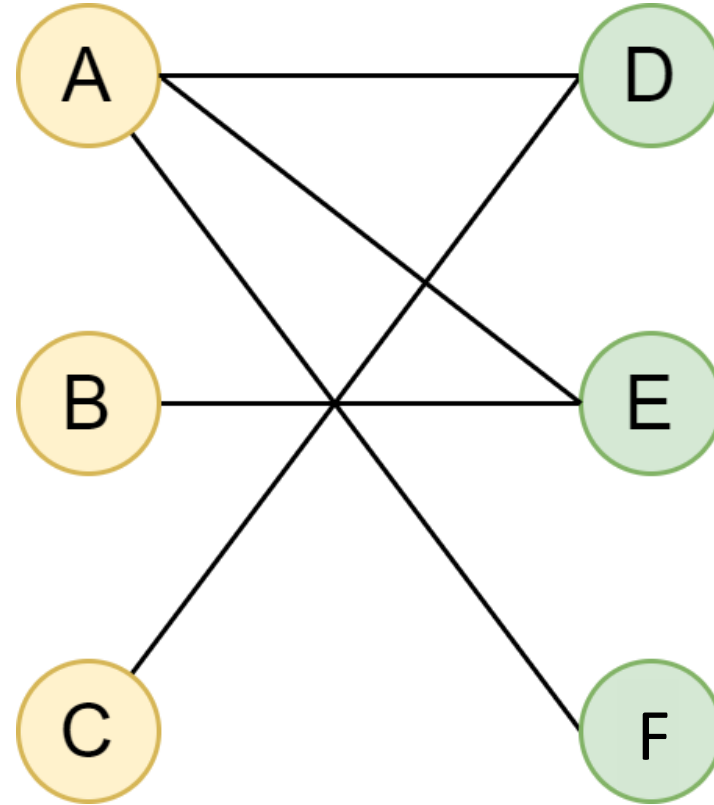


- Bağlı graf yapısındadır.
- Bütün düğümlerin derecesi aynıdır.

İki parçalı çizge

(Bipartite graph)

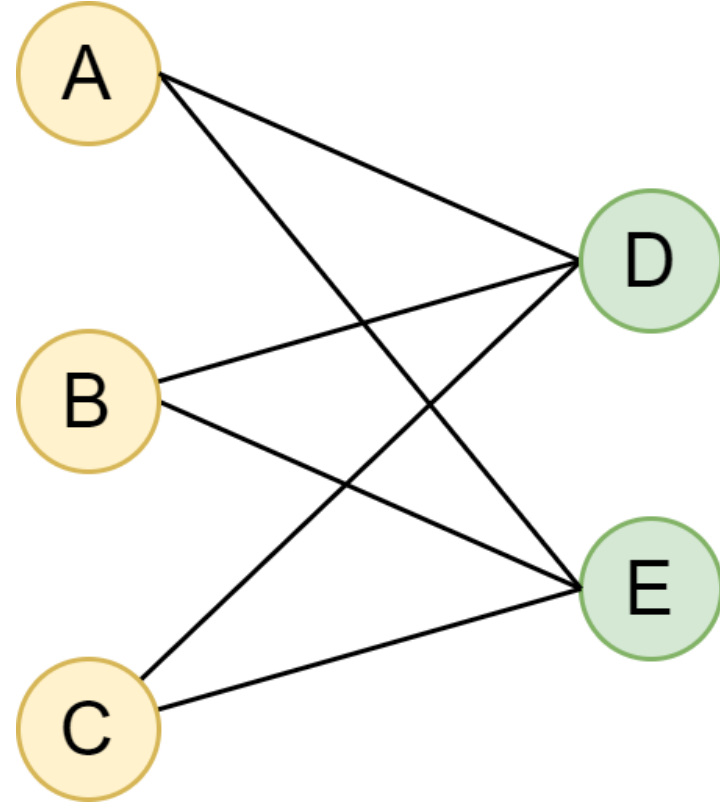
- Ayırık iki küme vardır ve bağlantılar, kenar bu iki küme arasında kurulur.



Tam iki parçalı çizge

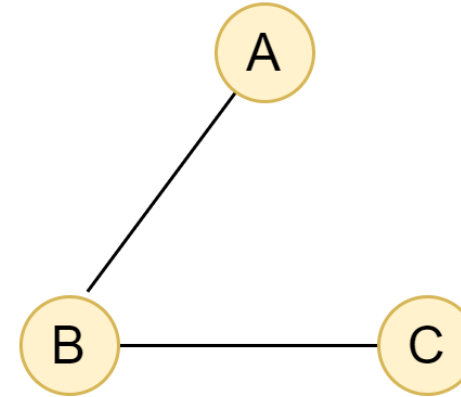
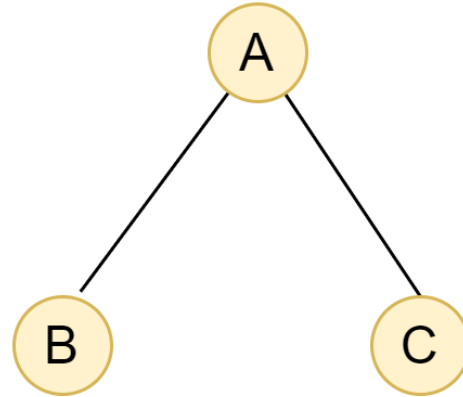
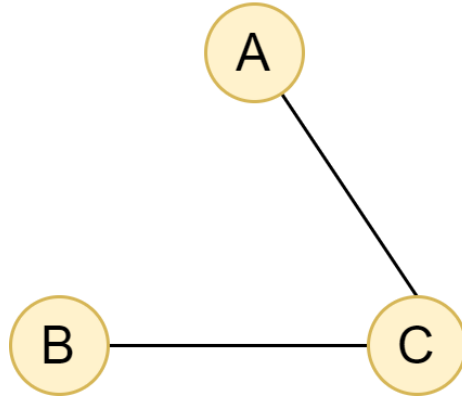
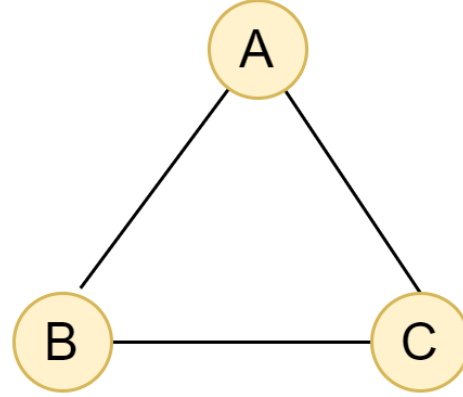
(Complete bipartite graph)

- İki parçalı graf içinde bir düğüm; diğer gruptaki her bir düğüme doğrudan bağlı olmalıdır.



Kapsama Ağacı

(Spanning tree)



- Bütün düğümleri içeren alt graftır. Her bir düğümün en fazla iki bağlantısı vardır.

Çizge Temsili

Graph Representation

- Bitişiklik matrisi (Adjacency matrix)
- Bitişiklik listesi (Adjacency list)
- Bitişiklik kümesi (Adjacency set)

Çizgelerin Depolanması

- V adet düğüm ve E adet kenarın depolanması ihtiyacı:
 - Düğümler dizi de depolanabilir.
 - Kenarlar başka bir şekilde (örneğin komşuluk matrisi ya da komşuluk listesi gibi) saklanmalıdır.
- Gerçekleştirilmek istenen işlevler
 - Belirli bir düğüm ile ilgili tüm kenarları alma
 - İki düğümün doğrudan bağlı olup olmadığını test etme
- Komşuluk/Bitişiklik matrisi ya da listesi kenarları depolamak için kullanılabilir.

Komşuluk Matrisi

(Adjacency Matrix)

- Bağlantı bilgilerini depolanın kolay bir yoludur.
 - İki düğümün bir birine doğrudan bağlı olup/olmama durumu test etmenin maliyeti: $O(1)$
- $n \times n$ matrisi
 - $a_{ij} = 1$ eğer i düğümünden j düğümüne bağlantı varsa
 - $a_{ij} = 0$ diğer durumda
- $\Theta(n^2)$ hafıza kullanır.
 - Yalnızca n birkaç binden az olduğunda kullanın.
 - Ve graf yoğun olduğunda

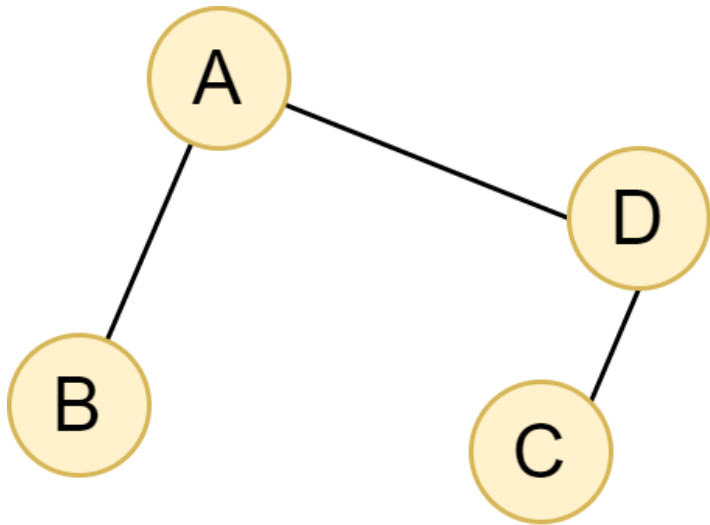
Komşuluk Matrisi

(Adjacency Matrix)

- Grafların komşu liste gösterimi ve komşu matris gösterimi olmak üzere temelde iki farklı gösterim şekli bulunmaktadır.
- Komşuluk matrisinde G grafi $|N| \times |N|$ tane elemandan oluşan bir komşuluk matrisi ile ifade edilir. Matrisi n_{ij} eğer i düğümünden j düğümüne bir kenar var ise 1 değerini alır; aksi durumda 0 değeri ile ifade edilir.

Komşuluk Matrisi

(Adjacency Matrix)



	0	1	2	3
0	0	1	0	1
1	1	0	0	0
2	0	0	0	1
3	1	0	1	0

Komşuluk Matrisi

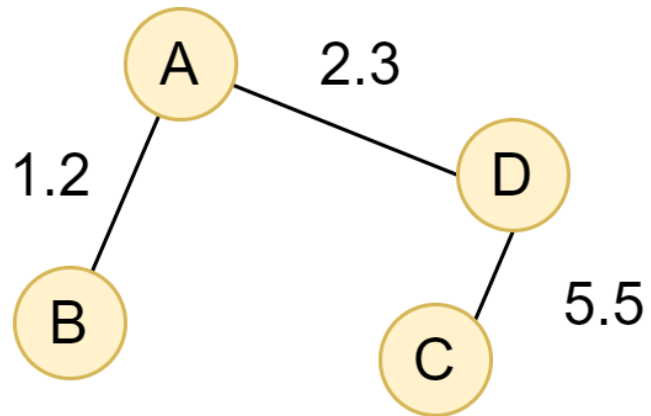
(Adjacency Matrix)

- $V \times V$ boyunda bir matris ile temsil edilir.
- Matris genellikle boolean veri tipindedir.
- Yönsüz graflarda simetriktir.
- Yönlü graflarda sadece bağlı olan düğümler dikkate alınarak oluşturulur.

	0	1	2	3
0	0	1	0	1
1	1	0	0	0
2	0	0	0	1
3	1	0	1	0

Ağırlık Matrisi

(Weighted Matrix)



	0	1	2	3
0	0	1.2	0	2.3
1	1.2	0	0	0
2	0	0	0	5.5
3	2.3	0	5.5	0

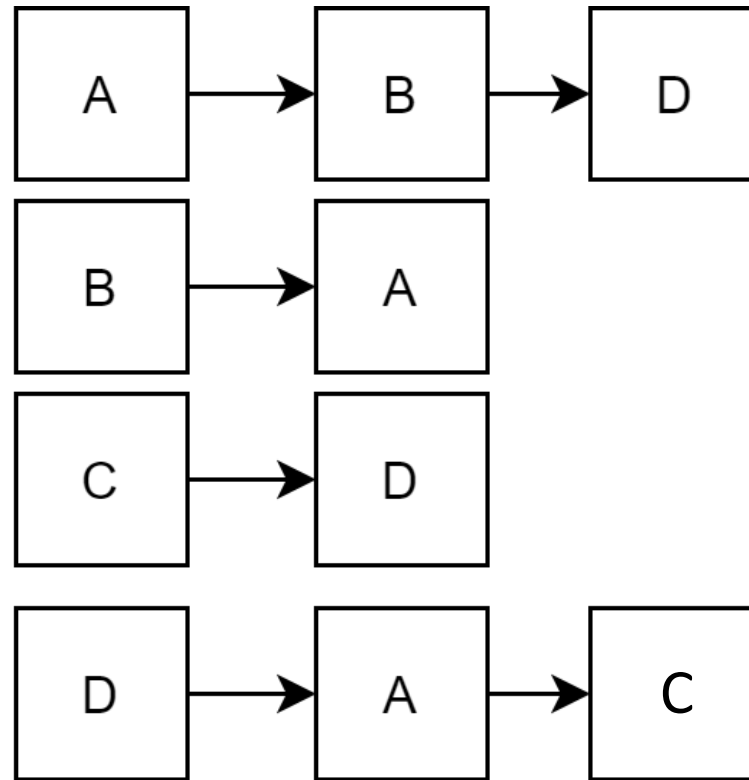
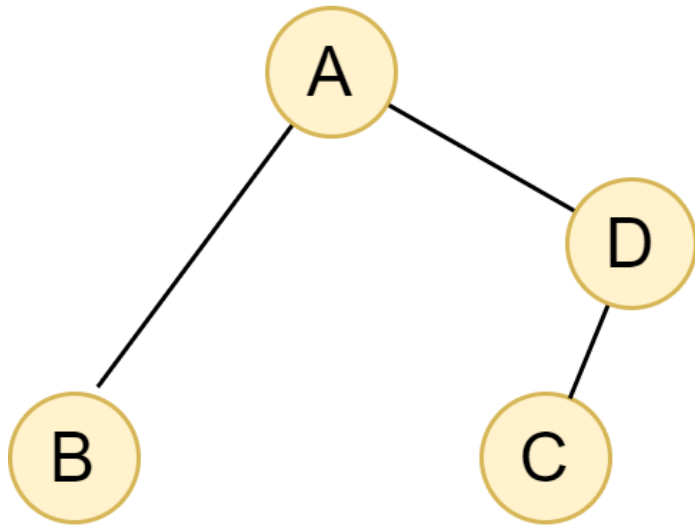
Komşu Liste

(Adjacency List)

- Her düğüm kendisinden çıkan kenarların bir listesine sahiptir.
 - Belirli bir düğüme ilişkin olayda kenarlar üzerinde iteratif işlevleri yürütmek kolaydır.
 - Listenin uzunluğu değişken olabilir.
 - Bellek kullanımı ($\Theta(n + m)$)

Komşu Liste

(Adjacency List)



Komşuluk Listesinin Uygulanması

- **Çözüm 1: Bağlı listeler**

- Çok fazla bellek kullanımı/bellek zaman ek yükü
- Dinamik ayrılmış bellek veya işaretçiler kullanmak kötü

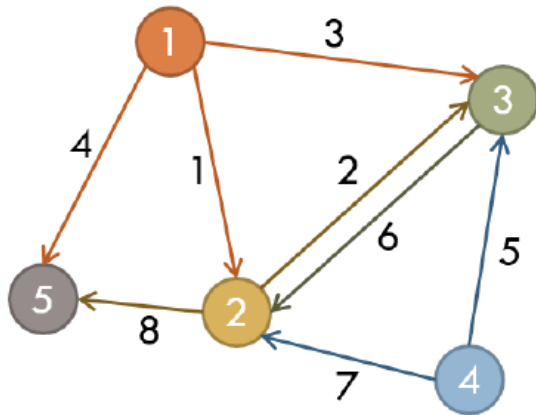
- **Çözüm 2: Vektrölerin Dizileri**

- Kodlama daha kolay, kötü hafıza sorunu yok
- Ancak çok yavaş.

- **Çözüm 3: Diziler**

- Toplam kenar sayısının bilinmesi varsayımı
- Çok hızlı ve bellek-verimli

Dizilerin bir uygulaması



ID	To	Next Edge ID
1	2	-
2	3	-
3	3	1
4	5	3
5	3	-
6	2	-
7	2	5
8	5	2

From	1	2	3	4	5
Last Edge ID	4	8	6	7	-

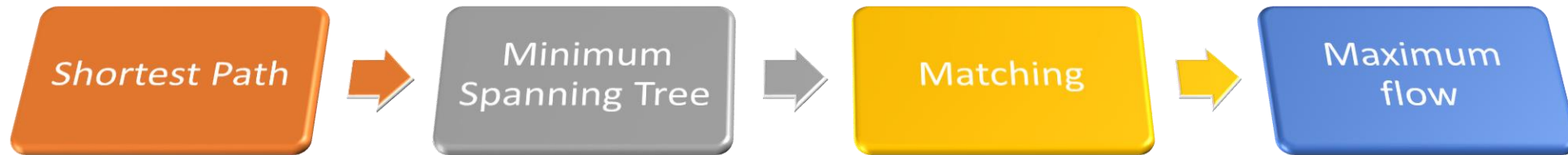
Çizge Uygulamaları

Applications of Graphs

Çizge Uygulamaları (Applications)

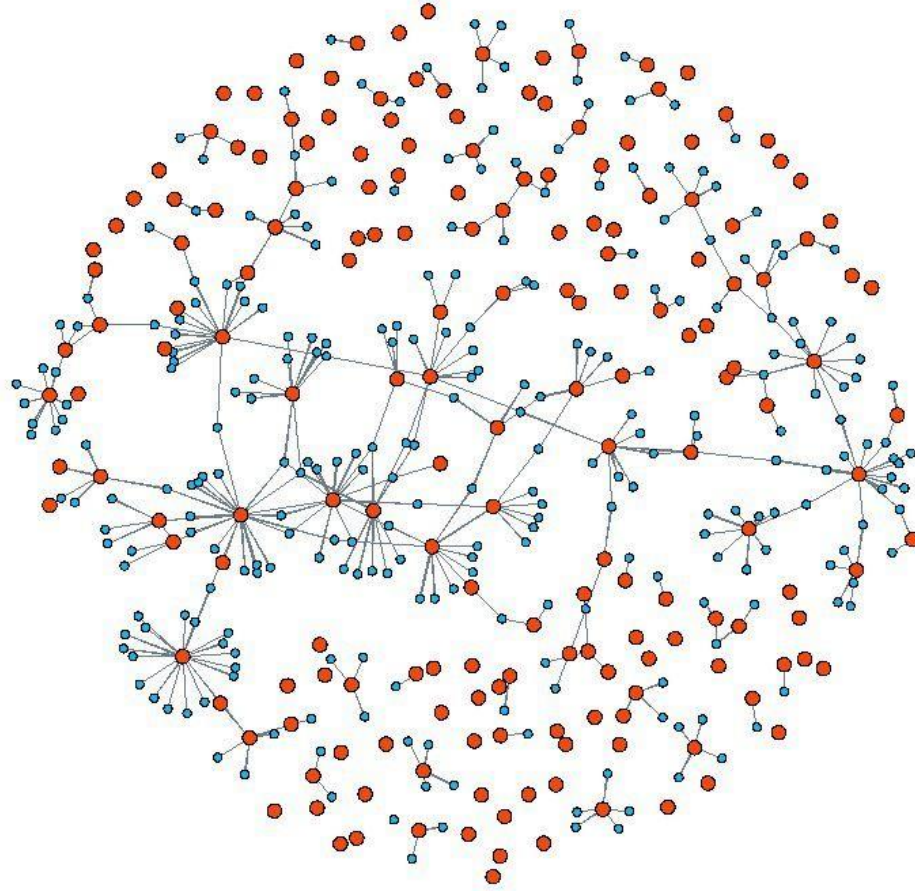
- En kısa yol problemleri
- Taşıma ağlarının modellenmesi (Karayolu ağı, Uçuş ağı)
- Bilgisayar ağlarının modellenmesi (LAN, Internet, Web)
- Eşleşme problemleri
- Gezgin satıcı problemleri
- Çizge renklendirme problemleri
- Veri tabanları: Entity Relationship diyagramları
- Elektronik devre elemanları arasındaki ilişkinin temsil edilmesi
- Rota planlama
- Tedarik zincirleri

Çizge problemlerinin dört sınıfı



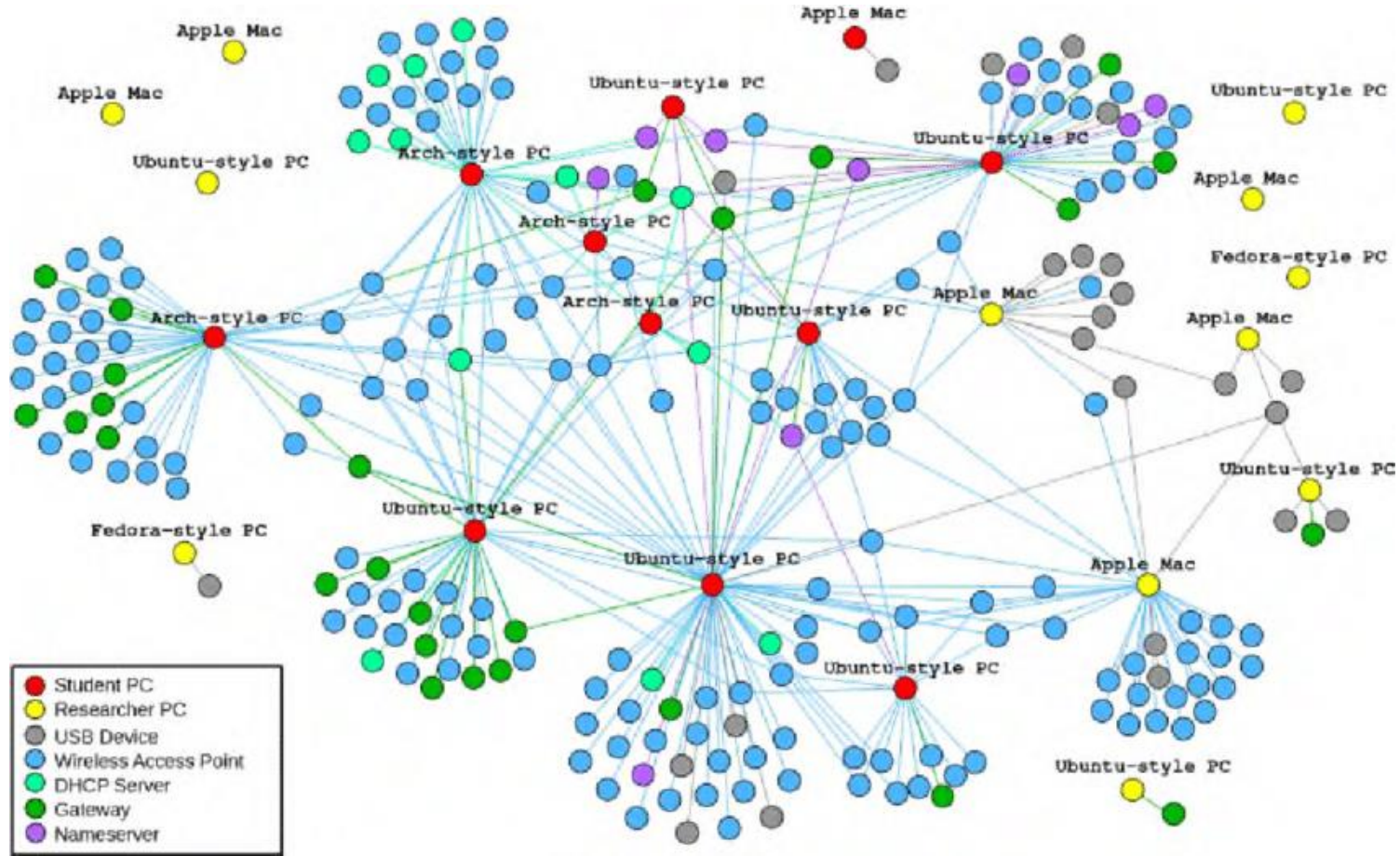
Çizge

(Graph)



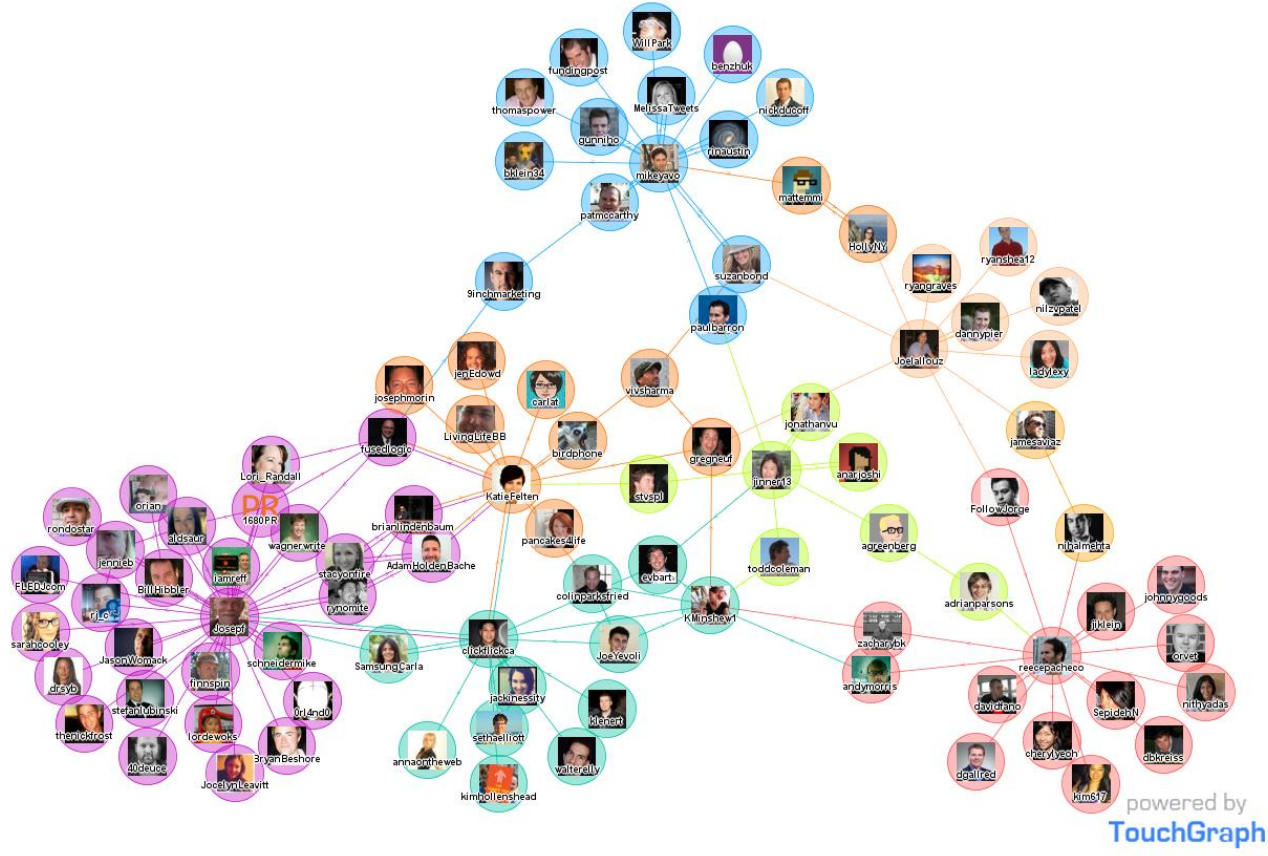
Ağ Çizgesi

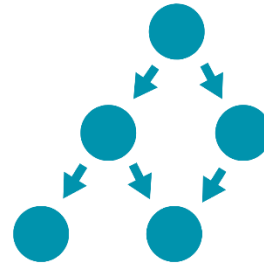
(Network Graph)



Sosyal Ağ Çizgesi

(Social Network Graph)





Veri Yapıları ve Algoritmalar

ZAFER CÖMERT

Öğretim Üyesi