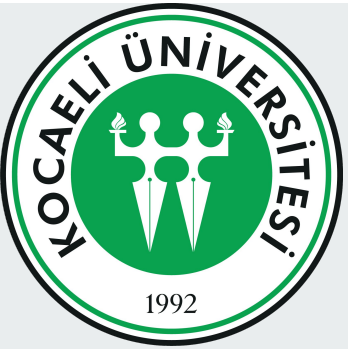


Kesikli Matematik

Graflar (Çizgeler)

Dr. Öğr. Üyesi Mehmet Ali ALTUNCU
Bilgisayar Mühendisliği

Graflar

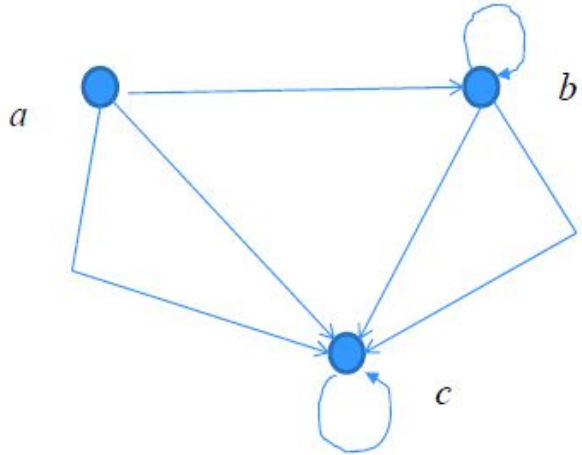
- Graflar köşelerden ve bu köşeleri birleştiren kenarlardan oluşan ayrık yapılardır.
- Graflar, kenarlarının yönlü olup olmamasına, aynı 2 köşeyi birleştiren birden çok kenarın olup olmamasına veya aynı köşeyi kendi kendine birleştiren bir kenarın olup olmamasına bağlı olarak sınıflandırılabilirler.
- Neredeyse akla gelebilen her alanda problemler graf modellerinin yardımıyla çözülebilirler.

Tanım: G = (V, E) grafı, boş olmayan köşeler (veya düğümler) kümesi olan V ve kenarlar kümesi olan E kümelerinden oluşur. Her bir kenarın 2 köşesi vardır, bu köşelere uç noktalar denir. Bir kenar uç noktalarını birleştirmektedir.

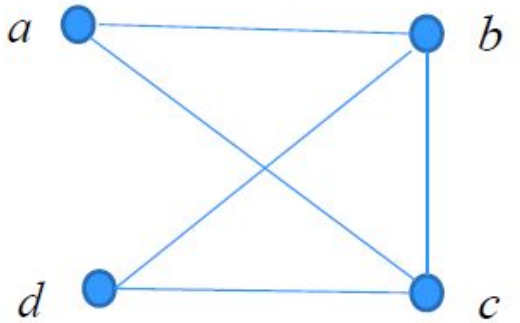
Graflar

- Basit bir grafta, her bir kenar iki farklı köşeyi birbirine bağlar ve aynı köşe çiftini birbirine bağlayan iki kenar yoktur.
- Çoklu graflar, aynı iki köşeyi birleştiren birden fazla kenara sahip olabilir. m farklı kenar u ve v köşelerini birleştirdiğinde, {u,v}'nin m katlı bir kenar olduğunu söyleyebiliriz.
- Bir köşeyi kendisine bağlayan bir kenara ise döngü denir.

Temel Graf Türleri



Yönlü Graf



Yönsüz Graf

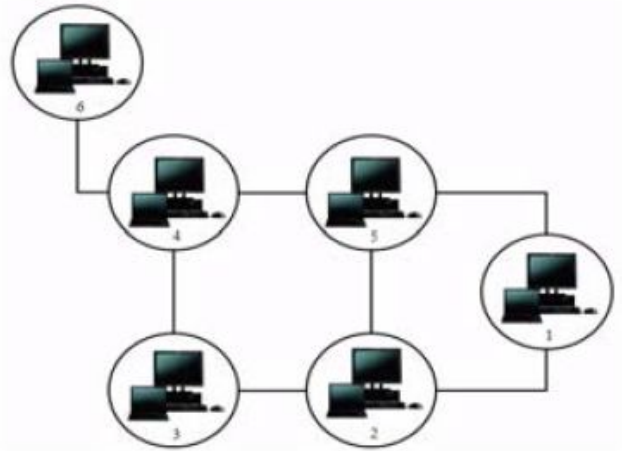
Graf Modelleri

Graflar aşağıdakileri uygulamalar için kullanılabilir:

- Bilgisayar ağları
- Sosyal ağlar
- İletişim ağları
- Bilgi ağları
- Yazılım Tasarımı
- Ulaşım ağları
- Biyolojik ağlar

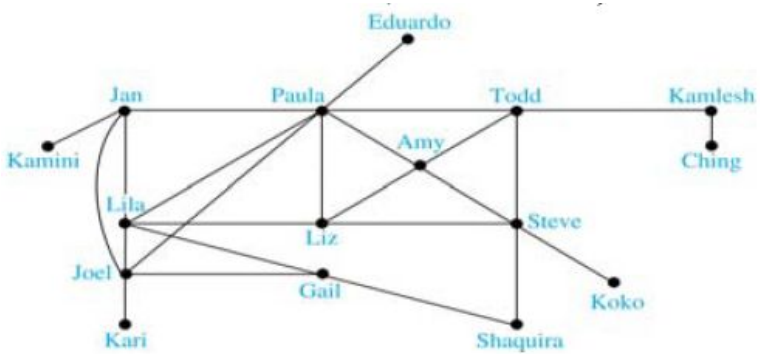
Graf Modelleri

- **Düğümler:** Bilgisayarlar
- **Kenarlar:** Bağlantılar



Graf Modelleri

- Graflar, insanlar veya gruplar arasındaki farklı türde ilişkilere dayalı sosyal ağları modellemek için kullanılabilir.
- Sosyal ağ, köşeler bireyleri veya kuruluşları temsil eder ve kenarlar bunlar arasındaki ilişkileri temsil eder.



Graf Modelleri - Bilgi Ağları

- Örneğin; bir web grafında, her köşe bir web sayfasını gösterecektir. Bir a sayfasından bir b sayfasına bağlantı varsa, a'dan b'ye bir kenar çizilir.
- Ya da farklı bilimsel yayınları, patentleri ve hukuki görüşleri de içeren çeşitli belgelerin alıntılarını ifade etmek için de kullanılabilirler.
- Graflar ulaşım ağlarının incelenmesinde de yaygın olarak kullanılmaktadır.
- Örneğin; yönlü graf kullanılarak modellenen havayolu ağlarında havaalanları köşelerle, her bir uçuş ise kenarlarla temsil edilebilir.
- Ya da otoyol ağlarında, köşeler kavşakları ve kenarlar yolları temsil edebilir.

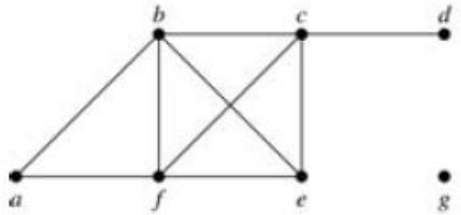
Yönsüz Graf

Tanım 1: Yönsüz bir G grafının u ve v köşeleri, u ve v G'nin e kenarının bitiş noktalarıysa, bu köşeler komşu veya bitişik olarak adlandırılır. Bu şekilde tanımlanan e kenarı ise u ve v köşeleri ile bağlı olarak adlandırılır.

Tanım 2: Yönsüz bir graftaki köşe derecesi o köşeye bağlı olan kenarların sayısıdır, ancak köşedeki bir döngü o köşenin derecesine iki birim katkıda bulunmaktadır. v köşesinin derecesi deg(v) şeklinde gösterilir.

Yönsüz Graf

Örnek: G grafindaki köşelerin dereceleri ve komşulukları nelerdir?



G

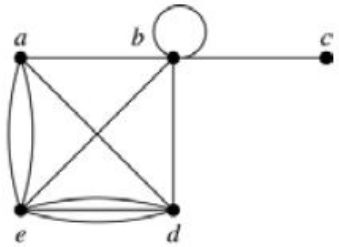
Çözüm: deg(a) = 2, deg(b) = deg(c) = deg(f) = 4, deg(d) = 1, deg(e) = 3, deg(g) = 0.

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) =Ø

Yönsüz Graf

Örnek: H grafındaki köşelerin dereceleri ve komşulukları nelerdir?



H

Çözüm: 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}.

Yönsüz Graf

El Sıkışma Teoremi: G = (V,E) m kenarlı yönsüz bir graf olsun. Bu durumda;

$$2m = \sum_{v \in V} \deg(v)$$

İspat: Her bir kenar, köşe derecelerinin toplamına iki birim katkıda bulunur. Çünkü bir kenar tam olarak iki köşe ile bağlıdır. Bunun anlamı köşelerin derecelerinin toplamı kenar sayısının iki katıdır.

Örnek: Her birinin derecesi 6 olan 10 köşeli bir grafta kaç tane kenar vardır.

Çözüm: 2m = 6*10 ise m= 30 olur.

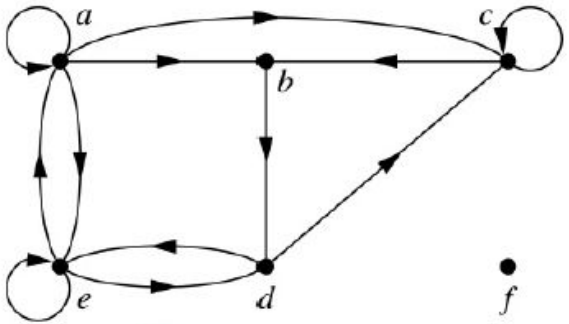
Yönlü Graf

Tanım 1: Yönlü bir G grafı (u,v) ise, u v'nin komşusudur veye v'nin komşusu u'dur denilir. u köşesi (u,v)'nin başlangıç köşesi ve v köşesi (u,v)'nin bitiş köşesi olarak adlandırılır. Bir döngüde başlangıç ve bitiş köşeleri aynıdır.

Tanım 2: Yönlü graflarda v köşesinin iç derecesi, v'yi bitiş olarak alan kenarların sayısıdır ve $\deg^-(v)$ ile gösterilir. v köşesinin dış derecesi ise, v'yi başlangıç köşesi olarak alan kenarların sayısıdır ve $\deg^+(v)$ ile gösterilir.

Yönlü Graf

Örnek: G grafındaki köşelerin iç dereceleri nelerdir?



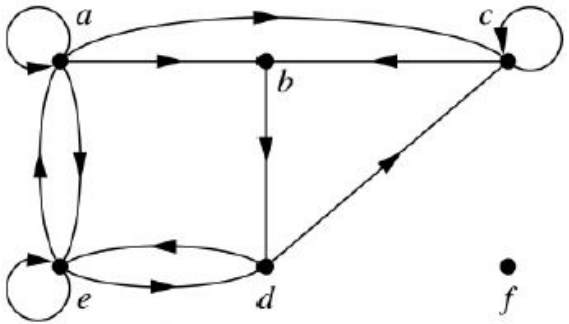
G

Çözüm: $\deg^-(a) = 2$, $\deg^-(b) = 2$, $\deg^-(c) = 3$,

$\deg^-(d) = 2$, $\deg^-(e) = 3$, $\deg^-(f) = 0$

Yönlü Graf

Örnek: G grafindaki köşelerin dış dereceleri nelerdir?



G

Çözüm: $\deg^+(a) = 4$, $\deg^+(b) = 1$, $\deg^+(c) = 2$,

$\deg^+(d) = 2$, $\deg^+(e) = 3$, $\deg^+(f) = 0$

Yönlü Graf

- Her bir kenar birer başlangıç ve bitiş köşelerine sahip olduğu için, yönlü graftaki bütün köşelerin giren derecelerinin toplamı ve çıkan derecelerinin toplamı ile aynıdır. Bu iki toplam da graftaki kenarların sayısına eşittir.

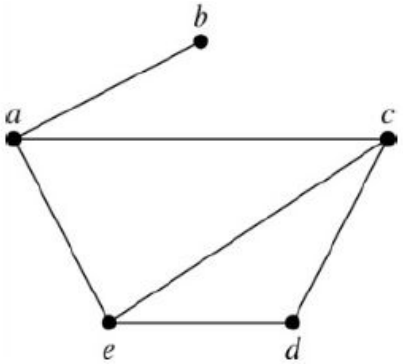
Teorem: G = (V,E) yönlü bir graf olsun. Bu durumda;

$$|E| = \sum_{v \in V} deg^-(v) = \sum_{v \in V} deg^+(v)$$

Grafların Gösterimi

- Çok katlı kenarı olmayan bir grafı göstermenin yolu, bu grafa ait bütün kenarları listelemektir. Diğer yol ise, grafın her köşesi için birleştiği köşeleri belirleyen komşuluk listesi kullanmaktır.

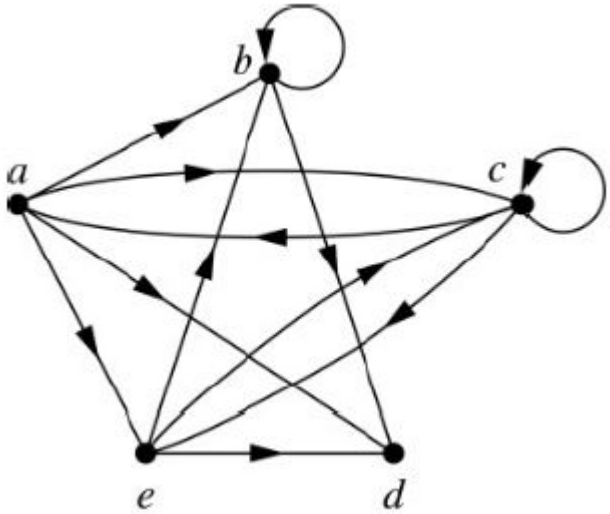
Örnek:



Komşuluk Listesi	
Köşe	**Komşular**
a	b,c,e
b	a
c	a,d,e
d	c,e
e	a,c,d

Grafların Gösterimi

Örnek 2:



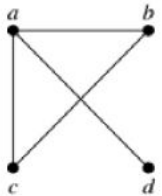
Komşuluk Listesi	
Köşe	**Komşular**
a	b,c,d,e
b	b,d
c	a,c,e
d	-
e	b,c,d

Grafların Gösterimi

- Eğer bir grafta çok sayıda kenar varsa, bu grafı komşuluk listesi kullanan algoritmalara bağlı şekilde göstermek çok zahmetli olabilir. Bu durumda çözüm matris şeklinde gösterimdir. G grafında komşuluk matrisi A = $[a_{ij}]$ ise;

$$a_{ij} = \begin{cases} 1 & \{v_i, v_j\}\ G\text{'nin kenarı ise} \\ 0 & \text{diğer durumlarda} \end{cases}$$

Örnek:



Komşuluk Matrisi

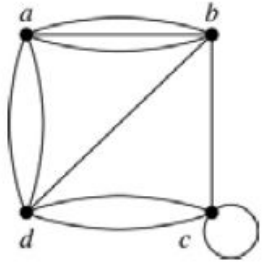
$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Grafların Gösterimi

- Eğer bir grafta çok sayıda kenar varsa, bu grafı komşuluk listesi kullanan algoritmalara bağlı şekilde göstermek çok zahmetli olabilir. Bu durumda çözüm matris şeklinde gösterimdir. G grafında komşuluk matrisi A = $[a_{ij}]$ ise;

$$a_{ij} = \begin{cases} 1 & \{v_i, v_j\}\ G\text{'nin kenarı ise} \\ 0 & \text{diğer durumlarda} \end{cases}$$

Örnek 2:

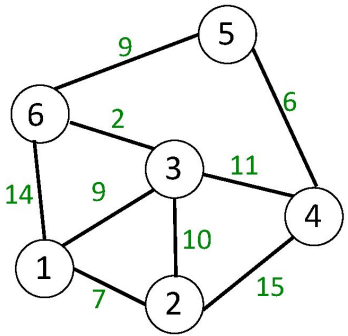


Komşuluk Matrisi

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

Ağırlıklı Graf

- Bir grafın üzerindeki kenarların değerleri eşit değilse ve her biri farklı bir değer alabiliyorsa bu tip graflara **maliyetli yada ağırlıklı graf** denir.
- Şehirlerin arasındaki mesafelerin hatlara değer olarak atandığı yol haritasını temsil eden graflar maliyetli graflar için örnek verilebilir.
- Ya da iş akış şemalarındaki, her işin bitirilme süresini gösteren graflar da yine maliyetli graflar için bir başka örnektir.
- Bir graftaki tüm hatlara ait maliyetlerin toplamı ise o grafın toplam maliyetini verir.



Graf Üzerinde Dolaşma

- Graf üzerinde dolaşma, grafın düğümleri ve kenarları üzerinde istenen bir işi yapacak veya bir problemi çözecek biçimde hareket etmektir.
- Graf üzerinde dolaşma yapan birçok yaklaşım vardır. En önemli iki tanesi şunlardır:
    - Breadth First Search (BFS) Algoritması
    - Depth First Search (DFS) Algoritması

Depth First Search (DFS) Algoritması

Standart bir DFS algoritması, grafın her düğümünü iki kategoriden birine yerleştirir:

- Ziyaret Edilen
- Ziyaret Edilmeyen

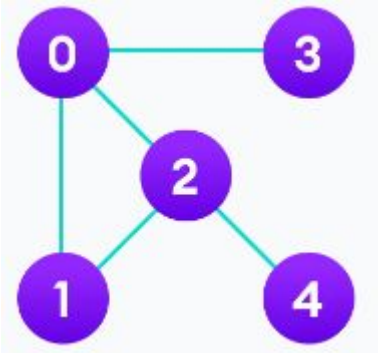
Algoritmanın amacı, döngülerden kaçınırken her düğümü ziyaret edilmiş olarak işaretlemektir.

DFS algoritması şu şekilde çalışır:

1. Grafın köşelerinden herhangi birini bir yığının üstüne koyarak başla.
2. Yığının en üst öğesini al ve ziyaret edilenler listesine ekle.
3. Bu düğümün bitişik düğümlerinin bir listesini oluştur. Ziyaret edilenler listesinde olmayanları yığının en üstüne ekle.
4. Yığın boşalana kadar 2. ve 3. adımları tekrarlamaya devam et.

Depth First Search (DFS) Algoritması

Örnek:

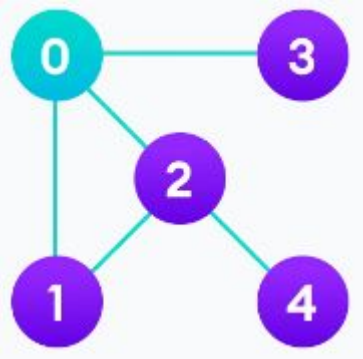


Ziyaret Edilen

Yığın

Depth First Search (DFS) Algoritması

1.Aşama: Düğüm 0'ı başlangıç düğümü seçiyoruz. DFS algoritması onu Ziyaret edilenler listesine koyarak ve komşu tüm düğümlerini yığına koyarak başlar.



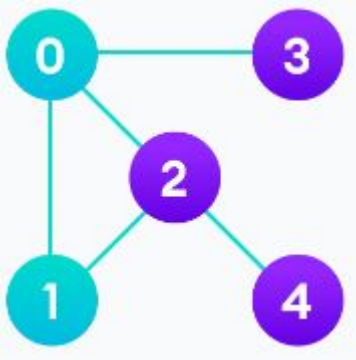
0				
1	2	3		

Ziyaret Edilen

Yığın

Depth First Search (DFS) Algoritması

2.Aşama: Ardından, yığının en üstündeki öğe, yani 1 düğümü ziyaret edilir ve komşu düğümlerine gidilir. 0 zaten ziyaret edildiğinden, bunun yerine 2 düğümü ziyaret edilir.



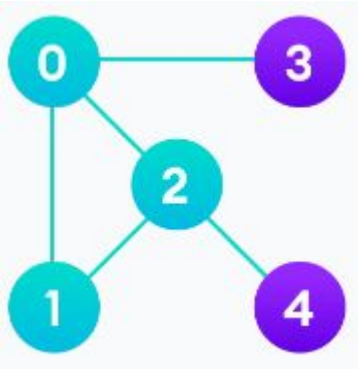
0	1			
2	3			

Ziyaret Edilen

Yığın

Depth First Search (DFS) Algoritması

3.Aşama: Düğüm 2'nin ziyaret edilmemiş komşu bir köşesi var (4 düğümü). Bu yüzden 4 düğümü yığının en üstüne eklenip, o ziyaret edilir.



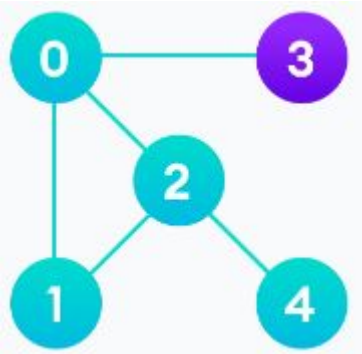
0	1	2		
4	3			

Ziyaret Edilen

Yığın

Depth First Search (DFS) Algoritması

4.Aşama: Yığının en üstündeki 4 düğümü ziyaret edilir. 4 düğümünün ziyaret edilmeyen bir komşusu olmadığı yığına düğüm eklenmez.



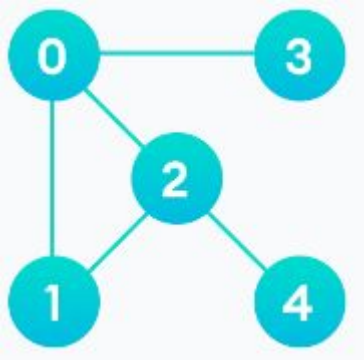
0	1	2	4	
3				

Ziyaret Edilen

Yığın

Depth First Search (DFS) Algoritması

5.Aşama: Son düğüm 3 ziyaret edilir. Ziyaret edilmemiş komşu düğümleri yok. Yığında herhangi bir düğüm kalmadığı için DFS algoritması sonlanır.



0	1	2	4	3	Ziyaret Edilen
					Yığın

Breadth First Search (BFS) Algoritması

Standart bir BFS algoritması DFS'de olduğu gibi, grafın her düğümünü iki kategoriden birine yerleştirir:

- Ziyaret Edilen
- Ziyaret Edilmeyen

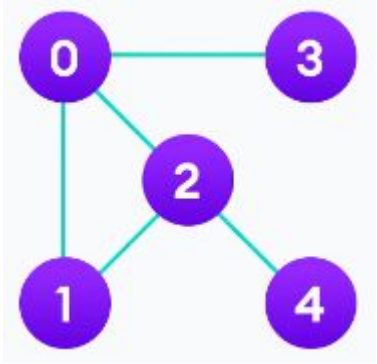
Algoritmanın amacı, döngülerden kaçınırken her düğümü ziyaret edilmiş olarak işaretlemektir.

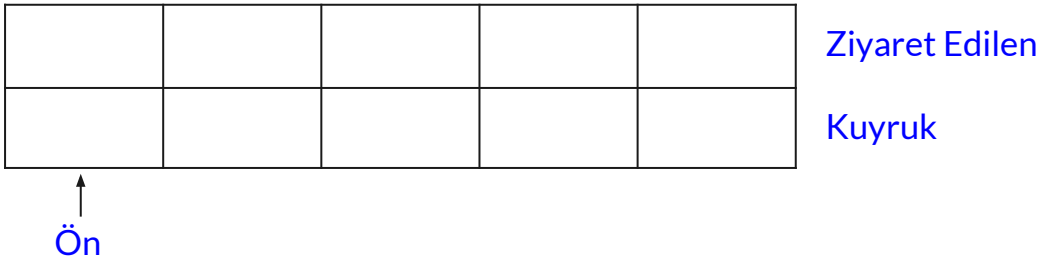
BFS algoritması şu şekilde çalışır:

1. Grafın düğümlerinden herhangi birini sıranın arkasına koyarak başla.
2. Sıranın ön öğesini al ve ziyaret edilenler listesine ekle.
3. Bu düğümün bitişik düğümlerinin bir listesini oluştur. Ziyaret edilenler listesinde olmayanları sıranın en arkasına ekle.
4. Kuyruk boşalana kadar 2. ve 3. adımları tekrarlamaya devam et.

Breadth First Search (BFS) Algoritması

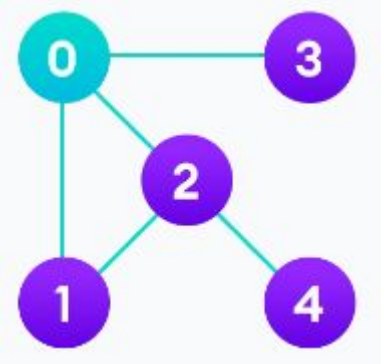
Örnek:





Breadth First Search (BFS) Algoritması

1.Aşama: Düğüm 0'ı başlangıç düğümü seçiyoruz. DFS algoritması onu Ziyaret edilenler listesine koyarak ve komşu tüm düğümlerini kuyruğa koyarak başlar.



0				
1	2	3		

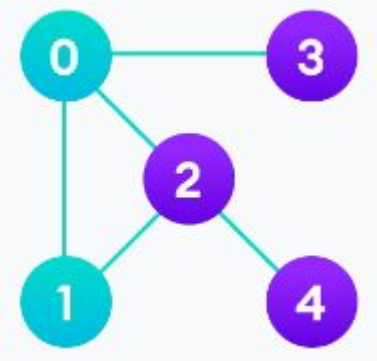
Ziyaret Edilen

Kuyruk

Ön

Breadth First Search (BFS) Algoritması

2.Aşama: Daha sonra, kuyruğun önündeki elemanı, yani 1'i ziyaret edip komşu düğümlerine gidilir. 0 düğümü zaten ziyaret edildiğinden, bunun yerine 2 düğümü ziyaret edilir.



0	1			
2	3			

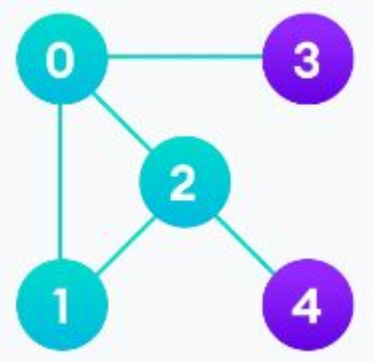
Ziyaret Edilen

Kuyruk

Ön

Breadth First Search (BFS) Algoritması

3.Aşama: Düğüm 2'nin ziyaret edilmemiş komşu bir düğümü var (4). Bu yüzden bu düğüm kuyruğun arkasına eklenir ve sıranın önünde olan 3 düğümü ziyaret edilir.



0	1	2		
3	4			

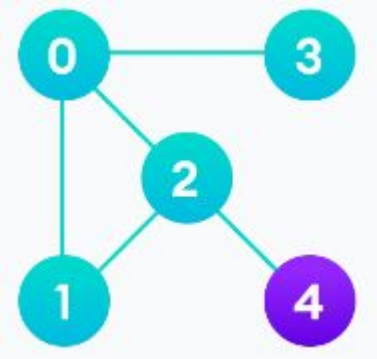
Ziyaret Edilen

Kuyruk

Ön

Breadth First Search (BFS) Algoritması

4.Aşama: 0 düğümünün komşusu olan 3 düğümü ziyaret edildiğinden, kuyrukta yalnızca 4 düğümü kalır.



0	1	2	3	
4				

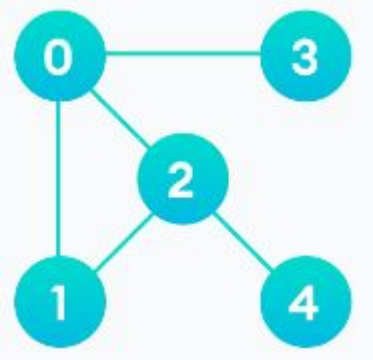
Ziyaret Edilen

Kuyruk

Ön

Breadth First Search (BFS) Algoritması

5.Aşama: Kuyrukta son kalan 4 düğümü ziyaret edilir ve BFS algoritması sonlanır.

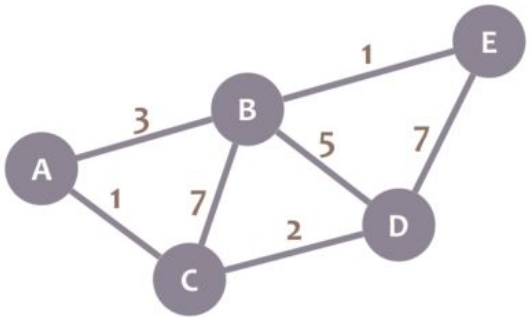


0	1	2	3	4	Ziyaret Edilen
					Kuyruk

Ön

En Kısa Yol Problemi

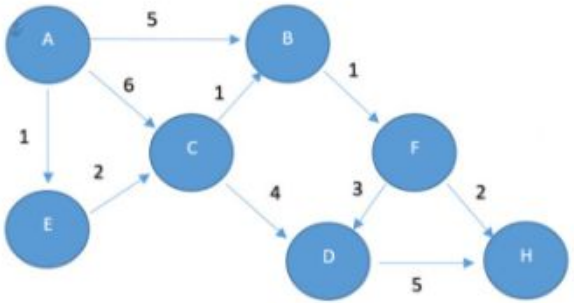
- Ağırlıklandırılmış bir grafta düğümler kümesi V, kenarlar kümesi E, ağırlıklar kümesi C olsun. c_{ij} , i ve j düğümleri arasındaki ağırlığı gösterir.
- Bir düğümden bütün düğümlere olan en kısa yol, verilen kaynak düğümünden (S) graftaki bütün düğümlere olan en kısa yolları ifade etmektedir.
- Ağırlıklı bir grafta iki düğüm arasındaki en kısa yolu bulmanın pek çok farklı algoritmaları mevcuttur.



Dijkstra Algoritması

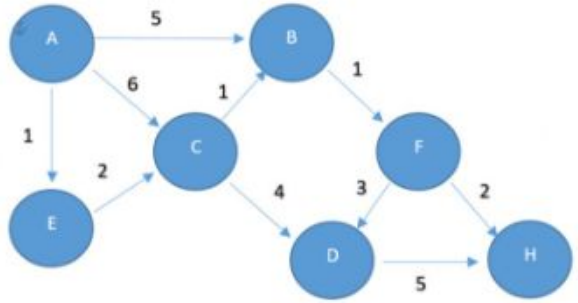
- Günümüzde oldukça popüler olan Dijkstra Algoritması, Google Maps, oyun programlama ulaşım ağlarında sıklıkla kullanılmaktadır.
- Algoritmanın temel amacı graf üzerinde bir düğümden başka bir düğüme giderken en ucuz maliyetle nasıl gidilebileceğini hesaplamaktır.

Örnek: Dijkstra algoritması ile A düğümünden H düğümüne en kısa yolun maliyeti nedir?



Dijkstra Algoritması

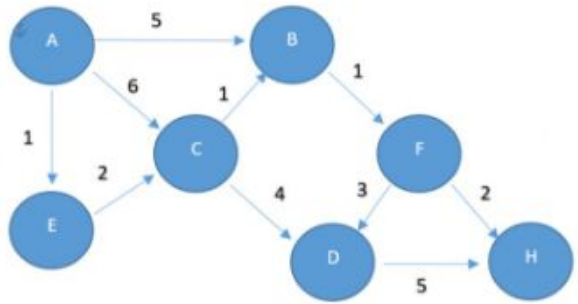
1.Aşama : Başlangıç düğümü olarak A düğümünden, diğer düğümlere erişim imkanı olmadığı için uzaklık değerlerine sonsuz değeri atılır.



Durum	A	B	C	D	E	F	H
Başlangıç	0	∞	∞	∞	∞	∞	∞

Dijkstra Algoritması

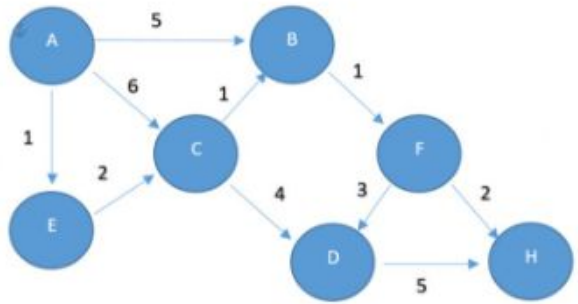
2.Aşama: Başlangıç düğümünden başlayarak komşu olan düğümlerine olan uzaklıklar, durum tablosunda gösterilir ve en kısa olan yolu olan düğüm seçilerek bir sonraki düğüme geçilir. Şekilde A düğümünün 3 düğüme komşuluğu bulunmaktadır ve en az maliyetli olan E düğümü seçilir.



Durum	A	B	C	D	E	F	H
Başlangıç	0	∞	∞	∞	∞	∞	∞
A	-	5	6	∞	1	∞	∞

Dijkstra Algoritması

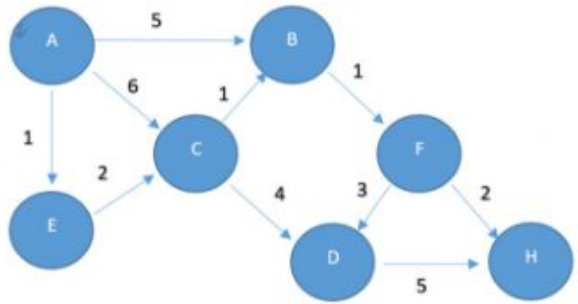
3.Aşama : E düğümünün C düğümüne komşuluğu bulunmakta. C değerine E üzerinden gitmek daha az maliyetli olduğu için C değeri üzerinde güncelleme yapılır. C düğümünün yeni değeri 1+2=3 olur. Bir sonraki düğüme geçmek için en küçük değere sahip ve daha önceden ziyaret edilmemiş olan C'den devam edilir.



Durum	A	B	C	D	E	F	H
Başlangıç	0	∞	∞	∞	∞	∞	∞
A(0)	-	5	6	∞	1	∞	∞
E(1)	-	5	3	∞	-	∞	∞

Dijkstra Algoritması

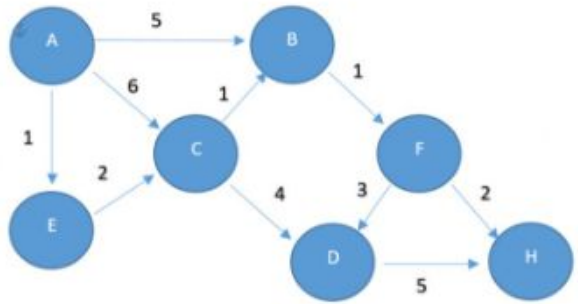
4.Aşama : C'den B ve D düğümlerine komşuluk bulunmakta. B' nin yeni değeri 3+1=4<5 olduğu için B düğümün değerini güncellenir. D'nin yeni değeri 3+4=7 oldu. Durum tablosunda B düğümünün değeri en düşük olduğu için B düğümünden devam edilir.



Durum	A	B	C	D	E	F	H
Başlangıç	0	∞	∞	∞	∞	∞	∞
A(0)	-	5	6	∞	1	∞	∞
E(1)	-	5	3	∞	-	∞	∞
C(3)	-	4	-	7	-	∞	∞

Dijkstra Algoritması

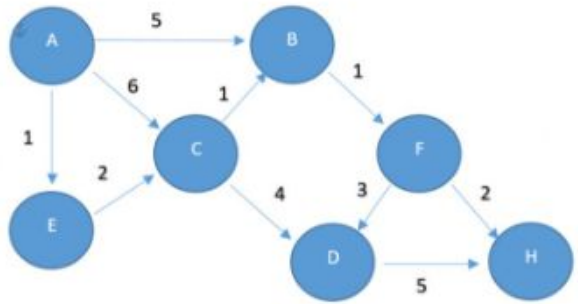
5.Aşama : B düğümünün F düğümüne komşuluğu bulunmakta ve 4+1=5'ten F'nin yeni değeri güncellenir.



Durum	A	B	C	D	E	F	H
Başlangıç	0	∞	∞	∞	∞	∞	∞
A(0)	-	5	6	∞	1	∞	∞
E(1)	-	5	3	∞	-	∞	∞
C(3)	-	4	-	7	-	∞	∞
B(4)	-	-	-	7	-	5	∞

Dijkstra Algoritması

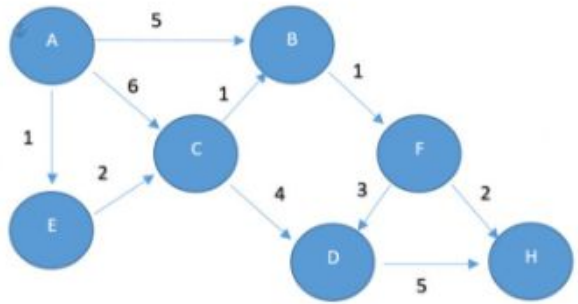
6.Aşama : F düğümü D ve H düğümüne komşuluğu bulunmakta. H düğümünün değeri 5+2=7 olur. D düğümünün değeri 7 iken 5+3=8 olması gerekmekteydi ama 8>7 olduğundan mevcut değerini korur.



Durum	A	B	C	D	E	F	H
Başlangıç	0	∞	∞	∞	∞	∞	∞
A(0)	-	5	6	∞	1	∞	∞
E(1)	-	5	3	∞	-	∞	∞
C(3)	-	4	-	7	-	∞	∞
B(4)	-	-	-	-	-	5	∞
F(5)	-	-	-	7	-	-	7

Dijkstra Algoritması

Sonuç: Tüm düğümlere ulaşıldığı için A düğümünden H düğümüne en kısa yol **A->E->C->B->F->H** şeklindedir. **Maliyeti ise 1+2+1+1+2=7'dir.**



Durum	A	B	C	D	E	F	H
Başlangıç	0	∞	∞	∞	∞	∞	∞
A(0)	-	5	6	∞	1	∞	∞
E(1)	-	5	3	∞	-	∞	∞
C(3)	-	4	-	7	-	∞	∞
B(4)	-	-	-	-	-	5	∞
F(5)	-	-	-	7	-	-	7

Kaynaklar

- **Kenneth Rosen,** “Discrete Mathematics and Its Applications”, 7th Edition , McGraw Hill Publishing Co., 2012.
- **Milos Hauskrecht,** “Discrete Mathematics for Computer Science”, University of Pittsburgh, Ders Notları.
- https://bm.erciyes.edu.tr/mcelik/bz205/En_Kisa_Yol_Problemi.pptx
- https://mertmekatronik.com/dijkstra-algoritmasi
- https://www.programiz.com/dsa/graph-dfs
- https://www.programiz.com/dsa/graph-bfs