BÖLÜM 10

ÇİZGELER

ANAHTAR KAVRAMLAR
Komşuluk Listesi, Komşuluk Matrisi,
Önce Genişlik Araması, Önce Derinlik Araması

ÖĞRENME HEDEFLERİ

1. Çizge teorisine ait temel kavramları tanımlayabilir
2. Çizge veri yapısını bilgisayar ortamında temsil edebilir
3. Çizge üzerinde dolaşma yöntemlerini uygulayabilir

ONUNCU BÖLÜM

GİRİŞ

Çizge teorisinin ilk örneği 1736'da "Königsberg'in Yedi Köprüsü" problemini çözen Euler tarafından sunulmuştur. Euler, şehri ve köprülerini temsil eden bir çizge veri yapısı oluşturarak ve bir köprüyü iki kez geçmeden bütün şehri dolaşmanın imkânsız olduğunu kanıtlamıştır. Bu çalışma, çizge teorisinin gelişimi ve bilgisayar bilimindeki ilk uygulamalarının temelini oluşturmuştur.

Günümüzde çizge teorisi bilgisayar bilimi ve mühendisliğinin önemli bir parçasıdır ve çizge teorisinin uygulamaları sosyal ağlar, doğal dil işleme, biyoinformatik ve ağ güvenliği gibi birçok alanda bulunabilir. Çizgeler ayrıca bilgisayar biliminde ağaçlar ve bağlantılı listeler gibi veri yapılarını temsil etmek için de kullanılır. Çizge teorisi algoritmaları, ağların yapısını analiz etmek, düğümler arasındaki en kısa yolları belirlemek ve veri aktarımı için en verimli yolları bulmak için kullanılır. Ayrıca sosyal ağlardaki kullanıcıların birbirleriyle olan ilişkileri bir çizge veri yapısı kullanarak temsil edilmekte ve ortaya çıkan yapının özellikleri araştırılmaktır.

Çizge dolaşma yöntemleri, çizgeye ait her bir düğümü bir düzene bağlı kalarak ziyaret etmeyi amaçlamaktadır. Çizge dolaşma algoritmaları, düğümler arasındaki yolları bulmak, döngüleri tespit etmek ve iki düğüm arasındaki en kısa yolu belirlemek için kullanılır. Önce genişlik ve önce derinlik olmak üzere iki farklı çizge dolaşma yöntemi tanımlanmıştır. Her iki yöntemde farklı stratejiler kullanarak çizgede bütün düğümleri ve kenarları keşfetmektedirler.

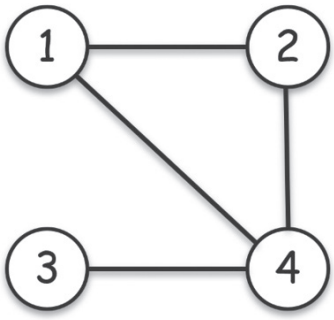
ARAŞTIRALIM

Çizge veri yapılarının farklı alanlarda kullanım yerlerini araştıralım.

1. ÇİZGE TEORİSİ

Çizge nesneler arasındaki karmaşık ilişkileri temsil etmek ve analiz etmek için kullanılan önemli bir araçtır. Çizge düğümler ve her biri bir çift düğümü birbirine bağlayan kenarlar kümesinden oluşur. Bir çizge G = (V, E) iki farklı yapı kullanılarak şekilde temsil edilebilmektedir: komşuluk listesi veya komşuluk matrisi. Komşuluk listesi çizgeye ait kenar sayısının az olması durumunda tercih edilir.

Çizgeler, düğümler arasındaki ilişkiye bağlı olarak kenarlar için yön belirlenip belirlenmeme durumuna göre ikiye ayrılmaktadır: yönsüz çizge ve yönlü çizge. Örneğin, bir ülkedeki farklı şehirler arasındaki bağlantıları gösteren bir çizge, yönsüz bir çizge olacaktır. Bu grafikte, iki şehir arasındaki bağlantı, bu iki şehir arasında bir yol veya başka bir ulaşım olduğunu gösterir, ancak bağlantının yönü önemli olmaz. Yönlü çizgede kenarların yönünün önemi vardır. Bir kenarın hangi düğümden başlayıp hangi düğüme ulaştığı kenarın yönünü belirler. Şehir içindeki bir yolun sadece tek bir yönden trafiğe izin vermesi yönlü bir kenar ile temsil edilir. Bu durumda sadece başlangıç düğümünden diğer düğüm çiftine ulaşmak söz konusudur.



a) Yönsüz bir çizge

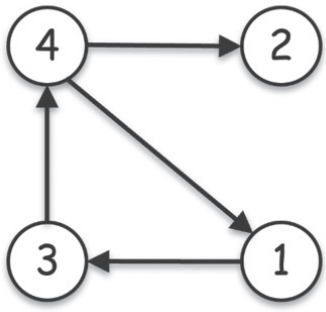
1 - 2,4
2 - 1,4
3 - 4
4 - 1,2,3

b) Komşuluk listesi

	1	2	3	4
1	0	1	0	1
2	1	0	0	1
3	0	0	0	1
4	1	1	1	0

c) Komşuluk matrisi

Şekil 10.1 Yönsüz çizge örneği ve komşuluk listesi ve matrisi gösterimi



a) Yönlü bir çizge

1 - 3
2 -
3 - 4
4 - 1,2

b) Komşuluk listesi

	1	2	3	4
1	0	0	1	0
2	0	0	0	0
3	0	0	0	1
4	1	1	0	0

c) Komşuluk matrisi

Şekil 10.2 Yönlü çizge örneği ve komşuluk listesi ve matrisi gösterimi

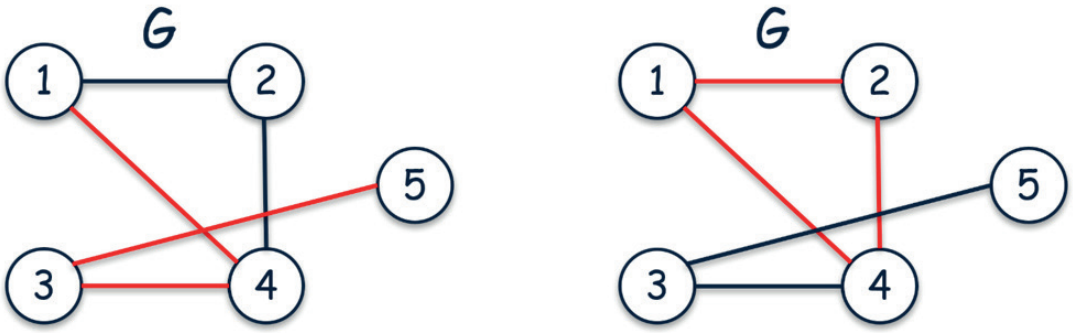
DİKKAT EDELİM

Bir yönlü çizgenin komşuluk matrisi simetrik iken, yönsüz çizgenin komşuluk matrisi simetrik olmayabilir.

Komşuluk listesinde; her bir düğüm için, içinde ilgili düğümden gidilebilecek diğer düğümlerin, yani ilgili düğümün komşularının tutulduğu bir liste oluşturulur. Komşuluk matrisinde ise; n dizinin düğüm sayısı olmak üzere nxn boyutlu bir matris oluşturulur ve eğer i. ve j. düğümler çizgede komşuysa matrisin (i, j) elemanı 1 değerini, değilse 0 değerini alır. Hem satırda hem de sütunda düğümleri göstermektedir.

Çizgelerde düğümler arasındaki ilişkiler farklıysa ve bu çizgeye yansıtılmak istemiyorsa, kenarlara bu ilişkinin yoğunluğuna göre bir ağırlık değeri atanabilmektedir. Böyle oluşturulan çizgeler ağırlıklı çizge olarak tanımlanmaktadır. Diğer yandan çizgelerde düğümlerin derecesi düğümle bağlantılı kenarlar üzerinden belirlenen komşu sayısı olarak ifade edilmektedir. Yönlü çizgede, düğümle bağlantılı kenarlar yönlü olduğundan iki farklı derece tanımlanmaktadır: girdi derecesi (düğümü bitiş düğümü olarak kabul eden kenarların sayısı) ve çıktı derecesi (düğümü başlangıç düğümü olarak kabul eden kenarların sayısı).

Bir çizgede yol, dizideki ardışık her iki düğüm için çizgede bir kenar olması koşuluyla sıralı bir düğümler dizisi olarak tanımlanmaktadır. Herhangi bir yolun uzunluğu ise yoldaki toplam kenar sayısıdır. Dizide bütün düğümlerin farklı olması durumunda yol basit yol olarak tanımlanır. İki farklı düğüm arasında bir yol var ise düğümler bağlı (connected) olarak adlandırılırlar. Bir çizgede bütün düğüm çiftleri arasında bir yol olması durumunda çizgeye bağlı çizge denir. Bir grafikteki döngü, aynı düğümden başlayan ve biten bir yoldur. Şekil 10.3.a'da 1, 4, 3, 5 düğümlerinden oluşan yol basit bir yoldur ama Şekil 10.3.b'de 1, 2, 4, 1 yolu bir döngüdür.

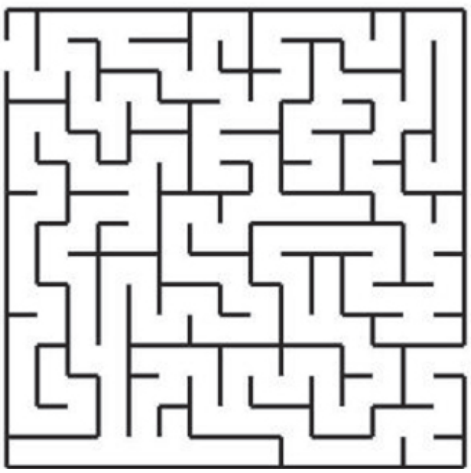


a) Basit yol örneği b) Döngü örneği

Şekil 10.3 Basit yol ve döngü örnekleri

2. ÇİZGE DOLAŞMA YÖNTEMLERİ

En temel çizge problemlerinden biri çizge üzerindeki her kenarı ve düğümü dolaşmaktadır. Buradaki amaç çizgeyi sistematik bir şekilde keşfetmektir. Etkili bir çizge dolaşma metodu Şekil 10.4'te verilen labirent üzerinden daha iyi anlaşılacaktır. Labirentteki her bir kavşağı bir düğümle ve her koridoru da bir kenarla temsil edelim ve çizge dolaşma yöntemi kullanarak çıkışı bulmaya çalışalım. Burada elbette çok zaman kaybedilmeden çıkışı bulurken aynı yerler tekrar tekrar ziyaret edilmemelidir. Yani çizge dolaşma problemlerinde, aynı düğüm ve kenarlar tekrar tekrar ziyaret edilmemelidir.



Şekil 10.4 Labirent örneği

Bir çizge dolaşma yönteminin iki adet anahtar işlemi bulunmaktadır. Bunlardan birincisi ziyaret ettiğinizde her düğümü işaretlemek ve ikincisi ise henüz tam olarak keşfetmediğiniz düğümleri takip etmek. Algoritmanın çalıştırılması esnasında, her düğüm için olası 3 durum bulunmaktadır. Birinci durum *Keşfedilmemiş*, ikinci durum *Keşfedilmiş* ve üçüncü durum *İşlenmiş* olarak ifade edilebilir. Keşfedilmemiş durum bir düğüm için başlangıç durumudur. Keşfedilmiş durumda algoritmanın düğümü ziyaret ettiği ama düğümle işinin henüz bitmediği anlaşılmaktadır. Son olarak İşlenmiş durumda ise, algoritmanın düğümü ziyaret ettiği ve bir daha düğümü ziyaret etmeyecek şekilde işlediği anlaşılmaktadır.

2.1. Önce Genişlik Arama (Breadth First Search)

Önce genişlik arama yönteminin amacı öncelikli olarak bir düğümü ve düğümün bütün komşularını keşfetmektir. Bu yöntemde, aynı seviyedeki düğümlerin tamamı işlendikten sonra ancak bir sonraki seviyedeki düğümlere geçilecektir. Bu yöntemde çizge, atomda çekirdek etrafında elektronların oluşturduğu yörüngeler gibi, merkez kabul edilen bir düğüm etrafındaki farklı katmanlara yerleştirilmiş düğümler olarak ele alınmakta ve düğümler en içteki yörüngeden en dıştaki yörüngeye doğru enlemesine bir şekilde keşfedilmektedir.

Yöntemde, çizge üzerindeki her bir u düğümü mesafe, ebeveyn ve renk olmak üzere üç parametre ile tanımlanır. Mesafe merkez s düğümünden u düğüme olan en kısa yolun uzunluğudur. Ebeveyn ise s'den u'ya giden en kısa yolda u'nun selefi olan düğümü ifade eder. Renk parametresi de u düğümünün yöntem esnasında durumunu gösterir. Beyaz renk düğümün keşfedilmemiş, gri renk keşfedilmiş ve siyah renk ise düğümün işlenmiş olduğunu ifade eder.

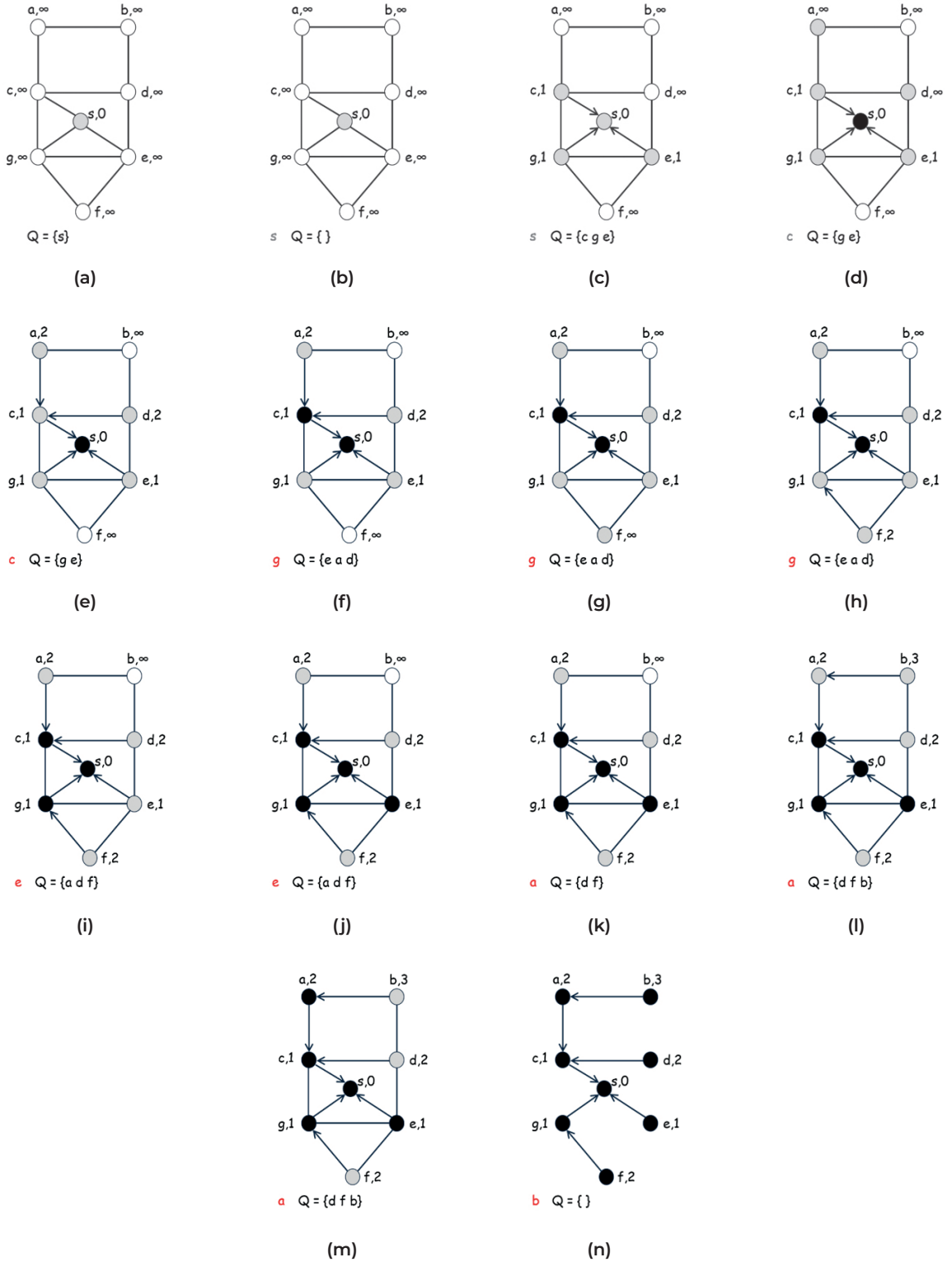
```
1.  BFS(G,s)

2.  for each vertex u of V
3.        u.color = white
4.        u.dis = ∞
5.        u.par = nil
6.  s.color = gray
7.  s.dis = 0
8.  initialize an empty queue Q
9.  Enqueu(Q,s)
10. while Q ≠ ∅
11.       u = Dequeu(Q)
12.       for each v ∈ Adj(u)
13.             if v.color = white
14.                   v.color = gray
15.                   v.dis = u.dis + 1
16.                   v.par = u
17.                   Enqueue(Q,v)
18.       u.color = black
```

Şekil 10.5 Önce Genişlik Arama yöntemine ait sözde kod

Şekil 10.5'te önce genişlik aramasına ait sözde kod verilmiştir. Kod incelendiğinde ilk olarak bütün düğümlerin renk, mesafe ve ebeveyn parametrelerine sırasıyla beyaz, sonsuz ve null başlangıç değerleri atanır. Sonrasında merkez s düğümünün rengi gri ve mesafesi 0 olarak atanır. Daha sonrasında bir boş kuyruk yapısı oluşturarak başlangıç düğümü kuyruğa eklenmiştir. Oluşturulan döngü yapısı kuyruk dolu olduğu sürece aşağıdaki işlemleri gerçekleştirecektir: (i) ilk olarak kuyruğun başındaki düğümü kuyruktan çıkartarak u değişkenine atayacak, (ii) bu düğümün bütün beyaz komşularının rengini gri yapacak, mesafesini bir artıracak ve ebeveynini

u düğümü olarak belirleyecek, (iii) ilgili komşu düğümleri kuyruğun sonuna ekleyecekti. Kuyruktaki bütün düğümler gezilerek işlenecektir. Kuyruktaki bütün düğümlerin gezilmesi ile önce genişlik arama yöntemi tamamlanacaktır.



Şekil 10.6 Önce Genişlik Arama algoritmasının örnek bir çizge üzerinde uygulanması

Önce genişlik arama algoritması örnek bir çizge kullanılarak bütün adımları Şekil 10.6'da gösterilmiştir. Yöntem esnasında:

(a) Başlangıç düğümü olarak s düğümü seçilmiş, rengi gri, ebeveyni null ve mesafe değeri 0 olarak atanmış ve kuyruğa eklenmiştir.

(b) Kuyruğun başındaki tek düğüm olan s çıkartılmıştır.

(c) S düğümünün beyaz komşuları olan c, g ve e düğümleri kuyruğa eklenmiş, mesafe değerleri 1, renkleri gri ve ebeveynleri s düğümü olarak atanmıştır.

(d) C düğümü kuyruktan çıkartılmıştır.

(e) C düğümün beyaz komşuları olan a ve d düğümleri keşfedilmiş ve renkleri gri ve mesafe değerleri 2 olarak atanmıştır.

(f) C düğümü siyaha boyanmıştır ve g düğümü kuyruktan çıkartılmıştır.

(g) G düğümünün beyaz komşusu f düğümü keşfedilmiş ve rengi gri yapılmıştır.

(h) F düğümünün mesafe değeri 2 olarak belirlenmiştir.

(i) G düğümü siyaha boyanmıştır ve e düğümü kuyruktan çıkarılmıştır.

(j) E düğümünün hiçbir komşusu beyaz olmadığı için siyaha boyanmıştır.

(k) Kuyruğun başındaki a düğümü çıkartılmıştır.

(l) B düğümü kuyruğa eklenerek mesafe değeri olarak 3 atanmıştır.

(m) A düğümü siyaha boyanmıştır. Gri renkli olan d, f ve b düğümleri sıra ile kuyruktan çıkartılmış ve çizgede başka beyaz düğüm kalmadığı için siyaha boyanmışlardır.

(n) Düğümlerin mesafe değerleri ve ebeveynlerini gösteren öncelikli arama ağacı elde edilmiştir.

2.2. Önce Derinlik Arama (Depth First Search)

Önce derinlik arama yöntemi, önce genişlik arama algoritmasından farklı olarak çizgenin düğümlerini enlemesine değil derinlemesine keşfeder. Yöntemde bir u düğümünü keşfedildiğinde, u'nun keşfedilmemiş bir v komşusunu seçilerek aramaya v ile devam edilir. Sonrasında u'nun diğer komşularını ziyaret etmek yerine v'nin ziyaret edilmemiş bir komşusu seçilerek aramaya ilgili komşu ile devam edilir. Bu şekilde ancak v komşusundan ulaşılabilecek tüm düğümler ziyaret edildikten sonra v işlenmiş kabul edilir ve u'nun bir diğer ziyaret edilmemiş komşusuna geçilir. İşlem, başlangıç düğümünden ulaşılabilen tüm düğümler keşfedilene kadar devam eder. Çizgede keşfedilmemiş düğümler kalırsa, bunlardan biri yeni başlangıç düğümü olarak seçilir ve benzer adımları tekrarlar.

Çizge üzerindeki her bir u düğümünün keşif ve bitiş zamanı, ebeveyn ve renk olmak üzere dört parametresi vardır. Ebeveyn ve renk parametreleri önce genişlik arama yöntemindeki gibidir. Önce genişlik arama yönteminden farklı olarak burada bir zaman sayacı vardır ve her keşfetme veya işleme işleminde bu sayaç artırılır. Düğümlerin keşif zamanı düğümü ilk keşfettiğimiz zamanı ve bitiş zamanı ise düğümün işlenme zamanını gösterir.

Önce derinlik arama yöntemini iki bölüme ayırmak mümkündür. Birinci bölüm her bir düğümün parametrelerin atandığı ve çizgede beyaz düğüm olduğu sürecek çağrılacak olan "**Visit**" fonksiyonundan oluşmaktadır.

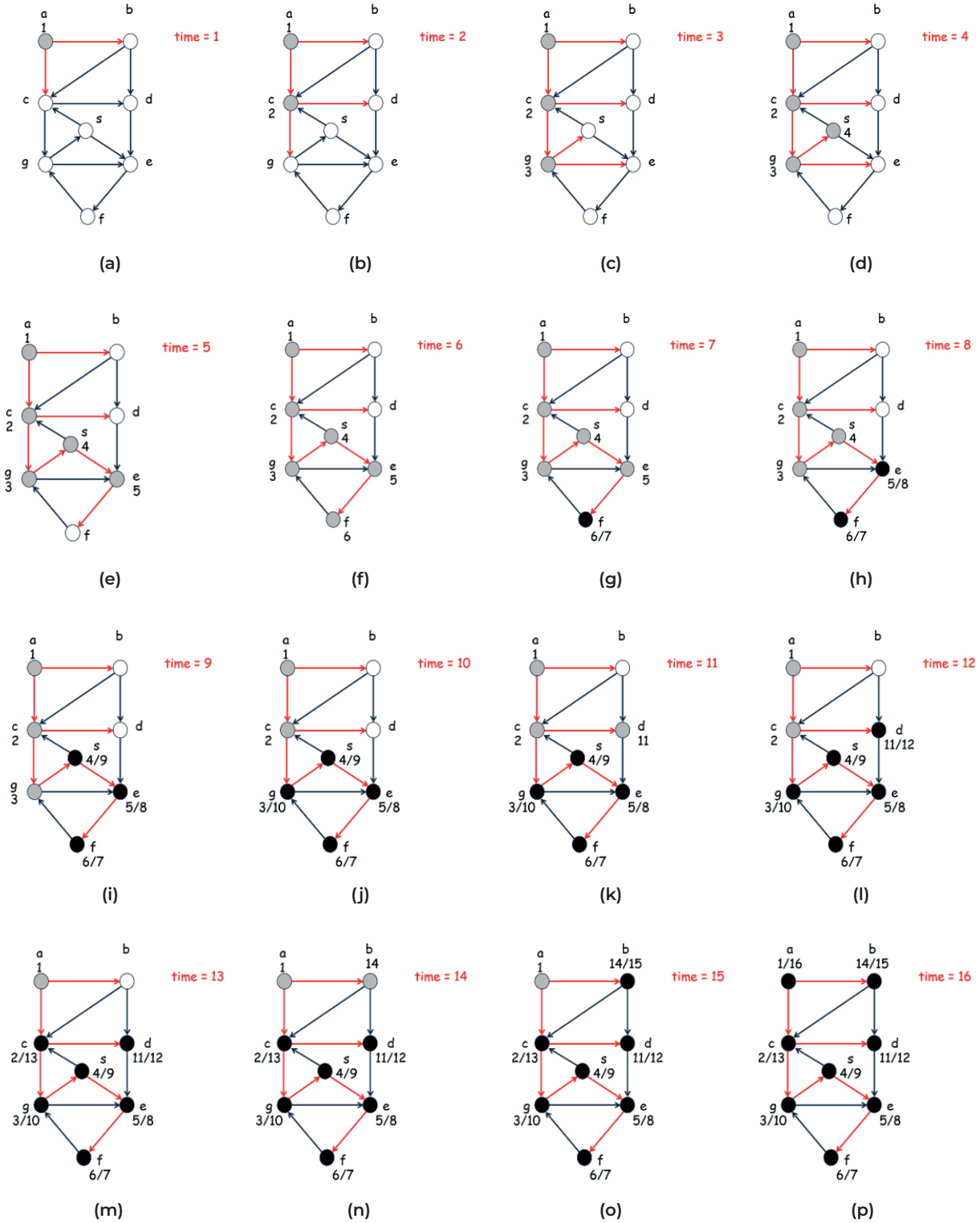
```
1.  DFS_Visit(u)
2.  u.color = gray
3.  time = time + 1
4.  u.dis = time
5.  for each v in Adj(u)
6.        if (v.color = white)
7.              v.par = u
8.              DFS_Visit(v)
9.  u.color = black
10. time = time + 1
11. u.fin = time
```

Şekil 10.7 Visit yöntemine ait sözde kod

İkinci bölüm ziyaret edilen düğümün rengini öncelikli olarak gri yapar ve zaman değişkenini bir artırır ve düğümün keşif zamanı olarak

atar. Sonrasında her bir beyaz komşu için ziyaret fonksiyonu tekrar çağrılır. Keşfedilen düğümden gidilecek beyaz düğüm kalmaz ise rengini siyaha boyar, zamanı bir artırır ve bitiş değerine atar.

Şekil 10.8 Önce derilik arama yöntemine ait sözde kod



Şekil 10.9 Önce Derinlik arama algoritmasının örnek bir çizge üzerinde uygulanması

Önce derinlik arama algoritması örnek bir çizge kullanılarak bütün adımları Şekil 10.9'da gösterilmiştir. Yöntem esnasında:

(a) Başlangıç düğümü olarak a belirlenmiş, keşif zamanı 1 ve rengi gri yapılmıştır.

(b) Beyaz komşulardan biri olan c düğümü seçilerek rengi gri ve keşif zamanı 2 olarak belirlenmiştir.

(c) A düğümü keşfedilerek rengi gri ve keşif zamanı 3 olarak belirlenmiştir.

(d) S düğümü keşfedilmiş, rengi gri ve keşif zamanı olarak 4 atanmıştır.

(e) S düğümünden gidilebilecek ve beyaz komşusu olan e düğümü ziyaret edilmiştir.

(f) F düğümü ziyaret edilerek rengi griye boyanmış ve keşif zamanı olarak 6 atanmıştır.

(g) F düğümünde gidilebilecek başka bir düğüm olmadığı için rengi siyaha boyanarak bitiş zaman değer 7 atanmıştır.

(h) E düğümüne geri dönülerek ve ziyaret edilebilecek başka bir düğüm olmadığı için siyaha boyanmıştır. Bitiş zamanı olarak 8 değeri atanmıştır.

(i) S düğümüne geri dönülmüş ve rengi siyaha boyanarak bitişi zamanı 9 olarak belirlenmiştir.

(j) C düğümüne geri dönülmüş ve d düğümü ziyaret edilmiştir.

(k) D düğümün keşif zamanı olarak 11 belirlenmiştir.

(l) D düğümünden gidilebilecek başka bir düğüm olmadığı için bitiş zamanı 12 olarak atanmıştır.

(m) C düğümüne geri dönülmüş ve başka ziyaret edilebilecek düğüm kalmadığı için bitiş zamanı 13 olarak atanmıştır.

(n) A düğümüne geri dönülmüş ve A'nın bir diğer komşusu B düğümü keşfedilerek rengi griye boyanmış ve keşif zamanı 14 belirlenmiştir.

(o) B düğümünde gidilebilecek başka bir düğüm olmadığı için siyaha boyanmış ve bitiş zaman değeri 15 atanmıştır.

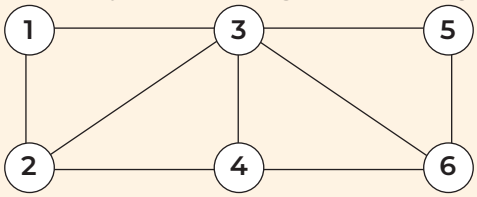
(p) A düğümüne geri dönülmüş ve başka bir düğüm kalmadığı için siyaha boyanmış ve bitiş değeri 16 atanmıştır.

BÖLÜM ÖZETİ

Çizgeler çeşitli nesneler arasındaki ilişkileri temsil etmek için kullanılan bir veri yapısıdır. Bir çizge, düğümler arasındaki bağlantıları temsil eden bir düğümler ve kenarlar kümesidir. Çizgeler, sosyal ağlar, matematiksel denklemler ve dil yapıları dahil olmak üzere çok çeşitli veri türlerini temsil etmek için kullanılabilir. Bu bölümde ilk olarak çizge teorisine ait temel kavramlar açıklanmıştır. Çizgeye ait kenarların yönlerinin önemli olması durumda yönlü çizge ve önemli olmaması durumunda yönsüz çizge türleri açıklanmıştır. Ayrıca bir çizge veri yapısını bilgisayar ortamında temsil edilebilmesi için gerekli komşuluk listesi ve matrisi anlatılmıştır. Çizge dolaşma algoritmaları, bir düğümden başlayarak ve çizgedeki tüm düğümleri belirli bir sırada ziyaret etmek için kullanılır. Bu algoritmalar, çizge veri yapısının aranmasına ve sıralanmasına izin verir. Yaygın çizge dolaşma algoritmaları, önce derinlik ve önce genişlik aramadır. Bu algoritmalar, çizge üzerinde dolaşmak için farklı stratejiler kullanır ve farklı çizge türlerine uygulanabilir. Önce derilik arama algoritması bir kuyruk veri yapısı yardımıyla düğümlerin hangi sırayla dolaşılması gerektiğine karar verir. Buna karşın önce derilik arama algoritması bir yığın veri yapısıyla gezinme işlemini gerçekleştirir.

GÖZDEN GEÇİRELİM

1. Aşağıda verilen çizgeye ait komşuluk matrisi gösterimi hangi seçenekte doğru verilmiştir?



a)

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

b)

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

c)

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

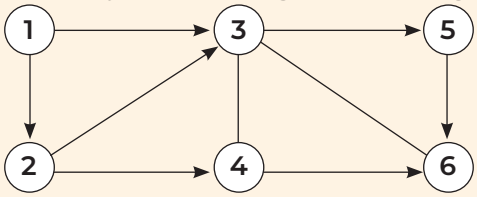
d)

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

e)

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

2. Aşağıda verilen çizgeye ait komşuluk matrisi gösterimi hangi seçenekte doğru verilmiştir?



a)

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

b)

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

c)

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

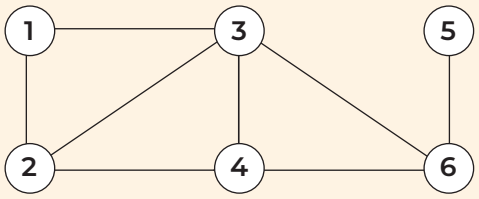
d)

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

e)

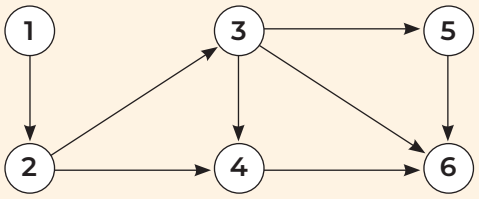
	1	2	3	4	5	6
1	0	1	0	0	1	0
2	0	0	1	1	0	0
3	0	1	0	1	0	1
4	0	0	1	0	1	0
5	0	0	0	0	0	1
6	0	0	0	0	0	0

3. **Aşağıda verilen çizge üzerinde, başlangıç düğümü "1" olacak şekilde önce genişlik arama algoritması çalıştırılacaktır. Bi¬r düğümün birden fazla komşusu olması durumunda komşular küçükten büyüğe doğru ziyaret edilecektir. Buna göre, çizgedeki düğümleri ziyaret etmenin doğru sırası hangi seçenekte verilmiştir?**



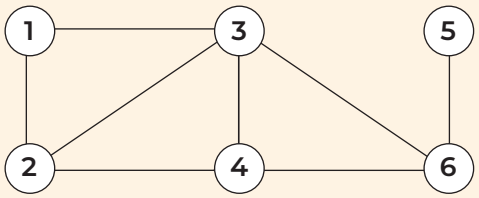
a) 1, 2, 3, 4, 5, 6
b) 1, 2, 3, 4, 6, 5
c) 1, 2, 4, 3, 6, 5
d) 1, 2, 4, 3, 5, 6
e) 1, 2, 3, 6, 4, 5

5. **Aşağıda verilen çizge üzerinde, başlangıç düğümü "1" olacak şekilde önce genişlik arama algoritması çalıştırılacaktır. Bi¬r düğümün birden fazla komşusu olması durumunda komşular küçükten büyüğe doğru ziyaret edilecektir. Buna göre, çizgedeki düğümleri ziyaret etmenin doğru sırası hangi seçenekte verilmiştir?**



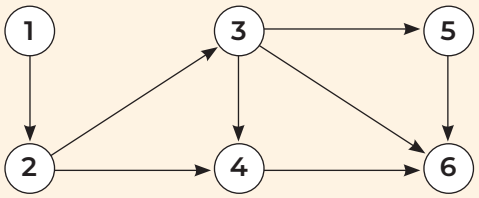
a) 1, 2, 3, 4, 5, 6
b) 1, 2, 3, 4, 6, 5
c) 1, 2, 4, 3, 6, 5
d) 1, 2, 4, 3, 5, 6
e) 1, 2, 3, 6, 4, 5

4. **Aşağıda verilen çizge üzerinde, başlangıç düğümü "1" olacak şekilde önce derinlik arama algoritması çalıştırılacaktır. Bi¬r düğümün birden fazla komşusu olması durumunda komşular küçükten büyüğe doğru ziyaret edilecektir. Buna göre, çizgedeki düğümleri ziyaret etmenin doğru sırası hangi seçenekte verilmiştir?**



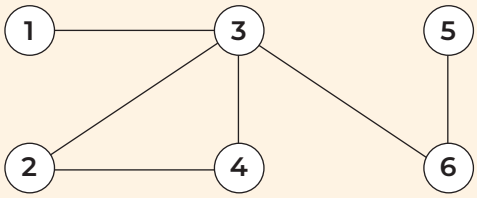
a) 1, 2, 3, 4, 5, 6
b) 1, 2, 3, 4, 6, 5
c) 1, 2, 4, 3, 6, 5
d) 1, 2, 4, 3, 5, 6
e) 1, 2, 3, 6, 4, 5

6. **Aşağıda verilen çizge üzerinde, başlangıç düğümü "1" olacak şekilde önce derinlik arama algoritması çalıştırılacaktır. Bi¬r düğümün birden fazla komşusu olması durumunda komşular küçükten büyüğe doğru ziyaret edilecektir. Buna göre, çizgedeki düğümleri ziyaret etmenin doğru sırası hangi seçenekte verilmiştir?**



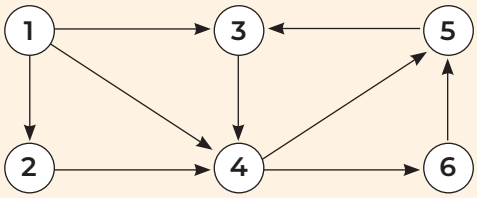
a) 1, 2, 3, 4, 5, 6
b) 1, 2, 3, 4, 6, 5
c) 1, 2, 4, 3, 6, 5
d) 1, 2, 4, 3, 5, 6
e) 1, 2, 3, 6, 4, 5

7. Aşağıda verilen çizge üzerinde, başlangıç düğümü "1" olacak şekilde önce genişlik arama algoritması çalıştırılacaktır. Bi¬r düğümün birden fazla komşusu olması durumunda komşular küçükten büyüğe doğru ziyaret edilecektir. Buna göre, algoritma sonunda 6 düğümünün mesafe parametresinin alacağı değer aşağıdakilerden hangisi olacaktır?



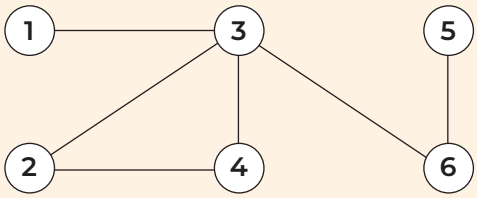
a) 1
b) 2
c) 3
d) 4
e) 5

9. Aşağıda verilen çizge üzerinde, başlangıç düğümü "1" olacak şekilde önce genişlik arama algoritması çalıştırılacaktır. Bi¬r düğümün birden fazla komşusu olması durumunda komşular küçükten büyüğe doğru ziyaret edilecektir. Buna göre, algoritma sonunda 4 düğümünün ebeveyn parametresinin alacağı değer aşağıdakilerden hangisi olacaktır?



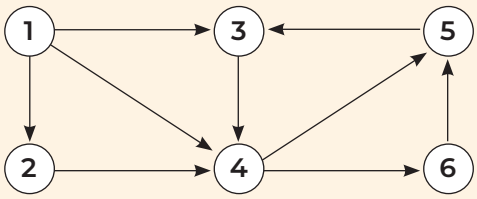
a) 1
b) 2
c) 3
d) 5
e) 6

8. Aşağıda verilen çizge üzerinde, başlangıç düğümü "1" olacak şekilde önce derinlik arama algoritması çalıştırılacaktır. Bi¬r düğümün birden fazla komşusu olması durumunda komşular küçükten büyüğe doğru ziyaret edilecektir. Buna göre, algoritma sonunda 4 düğümünün bitiş zamanı aşağıdakilerden hangisi olacaktır?



a) 3
b) 4
c) 5
d) 6
e) 7

10. Aşağıda verilen çizge üzerinde, başlangıç düğümü "1" olacak şekilde önce derinlik arama algoritması çalıştırılacaktır. Bi¬r düğümün birden fazla komşusu olması durumunda komşular küçükten büyüğe doğru ziyaret edilecektir. Buna göre, algoritma sonunda 3 düğümünün başlangıç zamanı aşağıdakilerden hangisi olacaktır?



a) 4
b) 5
c) 6
d) 7
e) 8

Yanıt Anahtarı: 1-A 2-A 3-B 4-B 5-A 6-B 7-B 8-C 9-A 10-B

Kaynakça

Levitin A. (2012) Introduction to The Design & Analysis of Algorithms, 3rd Edition, Pearson.

Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. (2009) Introduction to Algorithms, 3rd Edition, MIT Press.

Çölkesen T.R. (2019) Veri Yapıları ve Algoritmalar, Papatya Yayıncılık.

Sedgewick R., Wayne K. (2014) Algorithms, 4th Edition, Addison-Wesley

Çamoğlu K. (2021) Algoritma, Kodlab.