

BÖLÜM 6

ÖZETLEME (HASH) TABLoları

ANAHTAR KAVRAMLAR

Özetleme Tabloları, Çatışma,
Açık Adresleme, Ayırık Zincirleme



ÖĞRENME HEDEFLERİ

- 1 Özetleme tablolarının kullanılma amacını bilir.
- 2 Özetleme tablolarını farklı yöntemler kullanarak oluşturabilir.
- 3 Özetleme tablolarında oluşabilecek çatışmaların farkında olur.
- 4 Çatışma problemini çözmek için açık adresleme kullanabilir.
- 5 Ayırık zincirleme kullanarak çatışma problemini çözebilir.

GİRİŞ

Arama, bir veri yapısı üzerinde en fazla kullanılan işlemlerden birisidir. Arama işlemi ile beraber ekleme ve silme işlemlerini de bu bölüme kadar işlenen tüm veri yapılarından daha hızlı bir şekilde gerçekleştirmek için özetleme (hash) tabloları geliştirilmiştir. Bu veri yapısında hedeflenen çalışma süresi, yukarıda ifade edilen temel işlemler için $O(1)$ ile ifade edilen sabit bir zamandır. Hash tablosuna bir anahtar değeri eklemek için öncelikli olarak bu değerden bir indis üretilir. Bu indis değeri hash fonksiyonları yardımıyla hesaplanır. Hash fonksiyonları üretilen indis değerlerini tüm tablo üzerinde dengeli bir şekilde dağıtmaya çalışır. Bu nedenle hash fonksiyonları kullanarak, bir anahtar değeri hash tablosuna ekleme işlemi kararlı bir süreçtir. Ekleme için kullanılan hash fonksiyonu arama işlemi için de kullanılacaktır ve bu nedenle fonksiyonun aynı anahtar değer için aynı indis değerini üretmesi beklenir. Hash tablosuna yeni bir değer ekleme işleminde bazen ilgili hash tablosu alanı dolu olabilir. Bu durum çatışma olarak değerlendirilir. Bu bölümde farklı hash fonksiyonları ve çatışma durumunda kullanılacak çözüm yöntemleri anlatılacaktır.



1. HASH TABLOLARI

Doğrusal arama işleminin çalışma süresi $O(n)$ ile ifade edilebilir. İkili arama doğrusal aramadan daha hızlı çalışır ve $O(\log n)$ çalışma süresine sahiptir. İkili arama yöntemini kullanabilmek için listedeki elemanların sıralı olması gerekir. $O(n)$ ve $O(\log n)$ çalışma süresinden daha başarılı bir arama işlemi gerçekleştirebilmek amacıyla hash tabloları geliştirilmiştir. Hash tabloları kullanarak $O(1)$ çalışma süresi hedeflenmektedir. Bu yöntemin temeli, her bir aranan değer için sabit zamanda bir indis üretmek ve doğrudan tablodaki bu indisin gösterdiği alana erişmektir. Bir örnekle durumu açıklamak gerekirse, bir dersi alan 100 öğrencinin bilgilerini, boyutu 100 olan bir dizide tutmak mümkündür. Her bir öğrencinin dizide yer aldığı indis değeri öğrenci numarasından elde edilebilir. Örneğin öğrenci numarasının son iki hanesi, öğrencinin dizide yer aldığı alanı gösterebilir. Burada önemli olan verilen anahtar değerden hash tablosundaki bir alanı gösterecek bir indis üretebilmektir. Bu işlemi gerçekleştirmek için kullanılan fonksiyonlara hash fonksiyonları adı verilir. Bir hash fonksiyonunun bazı temel özelliklere sahip olması beklenir:

1. Hash fonksiyonlarının hesaplanması kolay olmalıdır.

2. Bu fonksiyonlar ile oluşan sürecin kararlı olması gerekir. Aynı girdi değerinin her zaman aynı çıktı değerini üretmesi beklenir. Tabloya eklenen bir değere daha sonra erişmek için aynı hash fonksiyonu kullanılacaktır.

3. Ayrıca bu fonksiyonların ürettiği değerlerin hash tablosu üzerindeki dağılımları dengeli olmalıdır. Bir başka ifade ile her bir hash tablosu indisinin eşit olasılıkla üretilmesi, bu fonksiyonun anahtar değerleri tablo üzerine yerleştirmesini kolaylaştıracaktır.

Hash tablolarında kullanılan fonksiyonlar için farklı yöntemler vardır. Bu yöntemler aşağıdaki gibi listelenebilir:

1. Bölme Yöntemi
2. Çarpma Yöntemi

3. Orta-Kare Yöntemi

4. Katlama Yöntemi

Bölme yönteminde kullanılan fonksiyon basit bir şekilde $\text{hash(anahtar)} = \text{anahtar} \bmod m$ şeklinde ifade edilebilir. Bu yöntemde anahtar değeri belirlenen bir m değerine bölünür ve kalan değer fonksiyonun ürettiği indis olarak değerlendirilir. m değeri tablonun boyutuna yakın bir asal sayı olarak belirlenir. Bu değerın asal olması anahtar değerlerin tablo üzerindeki dağılımının daha başarılı olmasını sağlar. Bu yöntemin en önemli dezavantajı birbirini takip eden anahtar değerlerin, hash tablosunda birbirini takip eden alanlara yerleştirilmesidir. Bu da hash tablosunun performansını düşürür. Ancak bu yöntemin avantajı anahtar değerleri farklı dizi alanlarına yerleştirmedeki başarısıdır. Örnek olarak boyutu 100 olan bir hash tablosunda m değerini 97 olarak seçelim. Anahtar değerın 62432 olduğunu varsayalım.

$\text{hash}(62432) = 62432 \bmod 97$ ifadesi indis olarak 61 değerini üretir.

Bir diğer yaklaşım olan çarpma yönteminde, 0 ile 1 arasında bir sabit değeri seçilir. Anahtar değeri seçilen bu sabit değeri ile çarpılır ve elde edilen değerin kesirli kısmı alınır. Bu değeri tablonun boyutu ile çarpılarak tabloda kullanılacak indis değeri hesaplanır. k , 0 ile 1 arasında seçilen bir değeri olmak üzere, burada ifade edilen işlemlerin formül ile ifadesi aşağıdaki gibidir:

$$\text{hash(anahtar)} = \text{boyut} * (k * \text{anahtar} \bmod 1)$$

k için en uygun değeri 0.618 olduğu önerilmiştir.

5634 anahtar değerini boyutu 100 olan bir hash tablosuna çarpma yöntemi ile yerleştirelim.

$$\begin{aligned} \text{hash}(5634) &= 100 * (0.618 * 5634 \bmod 1) \\ &= 100 * (3481.812 \bmod 1) \\ &= 100 * 0.812 \\ &= 81 \end{aligned}$$

81, çarpma yönteminin 5634 anahtar değeri için ürettiği indistir.



ARAŞTIRALIM

Çarpma yönteminde kullanılan 0 ile 1 arasındaki sabit için 0.618 değerinin önerilme nedenini araştır-
alım.

Orta-kare yönteminde ise anahtar değerin karesi alınır. Elde edilen değerin ortada yer alan r hanesi hash fonksiyonun sonucu olarak değeri-
lendirilir. Anahtar değeri oluşturan tüm rakam-
lar r rakamı oluşturmada etki eder. Bu nedenle
bu yöntem de başarılı bir fonksiyon olarak düşü-
nülebilir. Bu yöntemde tüm anahtarlar için aynı
 r hane kullanılmalıdır.

Şimdi de orta-kare yöntemini kullanarak 6723
anahtar değerinin hash tablosundaki indisini he-
saplayalım. Sağdan 3. ve 4. hanelerin indis olarak
kullanılacağını düşünelim.

$$6723^2 = 45198729$$

Bu durumda 87 değeri 6723 anahtar değerinin
hash tablosundaki indis değeri olacaktır.

Son olarak katlama yönteminde ise anahtar
değer rakamsal olarak daha önceden belirlenmiş
uzunlukta parçalara ayrılır. Daha sonra bu parça-
lar toplanır. Her bir parçada yer alacak hane sayı-
sı hash tablosuna göre belirlenir. En son parçada
daha az hane bulunabilir.

32421 anahtar değerini boyutu 100 olan bir
hash tablosuna yerleştirmeye çalışalım. Değeri-
mizi 2 haneden oluşan sayılara parçalayalım. Bu
durumda elde edilen sayılar 32, 42 ve 1 olacaktır.
Buradaki örnekte de olduğu gibi son parça daha
az sayıda hane içerebilir. Bu rakamlar toplandı-
ğında 75 değeri indis değeri olarak üretilir.



ARAŞTIRALIM

anahtar mod m hash fonksiyonunda kullanılacak m değerinin asal sayı olmasının getirdiği avantaj-
ları araştıralım.



ARAŞTIRALIM

Zamanı veya benzer harici parametreleri kullanan hash fonksiyonlarını araştıralım.

Kullanılan hash fonksiyonunun yapısına bağ-
lı olarak, hash tablosuna yeni bir eleman ekleme
işleminde fonksiyon farklı anahtarlar için aynı
değeri üretebilir. Bir başka ifade ile farklı değer-
ler için tablo üzerinde kullanılacak aynı indis de-
ğerini hesaplayabilir. Oluşan bu duruma çatışma

(collision) denir. Tablodaki bir alan birden fazla
değer için kullanılamayacağı için bu probleme
bir çözüm üretmek zorunludur. Çatışma proble-
mini çözmek için temel olarak iki farklı yöntem
kullanılır.



DİKKAT EDELİM

Çatışma problemini mümkün olduğunca azaltmak için anahtar değerleri tablo üzerinde dengeli bir şekilde dağıtan hash fonksiyonlarını kullanalım.

2. AÇIK ADRESLEME (OPEN ADDRESSING)

Hash fonksiyonunun aşağıdaki gibi verildiğini varsayalım.

$$\text{hash(anahtar)} = \text{anahtar mod } 10$$

Anahtar değerlerimiz 1 ile 100 arasında verilsin. Aşağıda verilen değerleri hash tablosuna yerleştirmeye çalışalım.

82 27 35 47 64

Verilen bu anahtar değerlerden 82 yukarıda verilen fonksiyonun sonucu olarak tabloda 2 indisine, 27 7 indisine ve 35 5 indisine yerleştirilir. 47 değeri fonksiyona verildiğinde 7 indisi üretilir. Ancak 7 indisinde daha önce yerleştirilmiş 27 değeri vardır. Bu durumda çatışma meydana gelmiş olur. Bu durum Şekil 6.1' de gösterilmektedir.

0	
1	
2	82
3	
4	
5	35
6	
7	27 47 ?
8	
9	

Şekil 6.1. Hash tablosuna eklenme sürecinde çatışma durumunun oluşması

Çatışmaya sebep olan sorunu çözmek için kullanılan yöntemlerden birisi açık adreslemedir. Bu yöntemde belirli kurallar çerçevesinde çatışmaya sebep olan yeni anahtar tablonun farklı bir alanına yerleştirilir. Bu yerleştirme işlemini farklı şekillerde gerçekleştiren yöntemler vardır.

1. Doğrusal Sınama
2. Karesel Sınama
3. İkili Özetleme

Doğrusal sınama yönteminde çatışma olduğu anda bir sonraki boş alana bakılır. Bu işlem ilk boş alan bulununcaya kadar devam eder. Bir başka ifade ile hash değeri birer birer artırılır. Bu artırma işlemi aşağıdaki fonksiyon ile gösterilebilir.

$$\text{hash(anahtar)} = ((\text{anahtar mod } m) + i) \text{ mod } m$$

Bu formülde yer alan i değişkeni deneme sayısını göstermektedir. Bir anahtarı ilk defa hash tablosuna yerleştirmeye çalıştığımızda bu değişkenin değeri 0 olarak kabul edilir. İlk çatışma olduğunda i 1 olur. Aynı anahtar için her çatışmada i değeri 1 artırılır. Bu şekilde tablodaki ilk boş alana erişilinceye kadar artırma işlemi devam eder. Örneğin, 47 değeri eklenmek istendiği zaman ilk denemede 7 indisi üretilir. Bu alan hash tablosunda doludur. 0 olan i değeri 1 artırılır. Doğrusal sınama formülümüz kullanıldığında, yeni üretilen hash tablosu indisi 8 olur (Şekil 6.2).

0	
1	
2	82
3	
4	64
5	35
6	
7	27
8	47
9	

Şekil 6.2. 5 farklı değer doğrusal sına ile hash tablosuna eklenmesi

Karesel sına yönteminde çatışma olduğunda farklı bir formül ile boş alan bulunmaya çalışılır. Buradaki temel amaç kümeleme oluşmasını engellemektir. Çatışmadan sonra ilk boş alanı kullanmak yerine farklı bir değer hesaplanarak ilerlenir ve yerleştirme işlemi yapılır. Bu ilerleme miktarı, deneme sayısı ve bu sayının karesi daha önceden belirlenmiş sabit değerler ile çarpılarak elde edilir. Ancak yeni hesaplanan alanda da çatışma olursa deneme sayısını gösteren i değişkeni bir artırılır ve hesaplama işlemi tekrarlanır. Karesel sına yaklaşımında çatışma sonunda yeni yerleştirme alanı hesaplayan formül aşağıda verilmiştir.

$$\text{hash(anahtar)} = ((\text{anahtar mod } m) + s_1i + s_2i^2) \text{ mod } m$$

82 27 35 47 64

değerlerini karesel sına ile aynı hash tablosuna yerleştirelim. s_1 değerinin 1, s_2 değerinin 3 olarak belirlendiğini varsayalım. Hash fonksiyonu 47 değeri için ilk denemede ($i=0$) 7 değerini üretir. 7 indisine 27 anahtarı daha önce yerleştirildiği için 2. deneme ($i=1$) gerçekleştirilir. Fonksiyon, 47 anahtarı için 2. denemede 1 değeri üretir $((47 \text{ mod } 10) + 1*1 + 3*1) \text{ mod } 10$. Elde edilen yeni hash tablosu Şekil 6.3'te verilmiştir.

0	
1	47
2	82
3	
4	64
5	35
6	
7	27
8	
9	

Şekil 6.3. 5 farklı değer karesel sına ile hash tablosuna eklenmesi

İkili özetleme yönteminde 2 farklı hash fonksiyonu kullanılır.

$$\text{hash(anahtar)} = (\text{hash1(anahtar)} + i*\text{hash2(anahtar)}) \text{ mod } m$$

$$\text{hash1(anahtar)} = \text{anahtar mod } m$$

$$\text{hash2(anahtar)} = \text{anahtar mod } m-1$$

Deneme sayısını gösteren i değişkeni diğer yöntemlerde olduğu gibi ilk denemede 0 değerine sahiptir. Dolayısı ile ilk deneme için 2. hash fonksiyonunun bir etkisi yoktur. 2. denemeden

itibaren iki fonksiyonun birlikte ürettiği indis değeri kullanılarak hash tablosuna yerleştirme işlemi gerçekleştirilir.

82 27 35 47 64

değerlerinin ikili özetleme yaklaşımıyla boyutu 10 olan bir hash tablosuna yerleştirilme işlemi Şekil 6.4'te gösterilmiştir.

0	
1	
2	82
3	
4	64
5	35
6	
7	27
8	
9	47

Şekil 6.4. 5 farklı değer için ikili özetleme ile hash tablosuna eklenmesi

Tüm açık adresleme yöntemlerinde bazı anahtar değerler ilk denemede hash tablosuna yerleştirilemeyebilirler. Bu nedenle bir hash tablosunda bir anahtar değeri arama işleminde fonksiyonunun ürettiği ilk alanda değer bulunmazsa bu ilgili anahtar değerinin tablodaki olmadığı anlamına gelmez. Aynen tabloya ekleme işleminde olduğu gibi kullanılan fonksiyonda *i* (deneme sayısı) değişkeninin değerini bir artırarak arama işlemine devam etmek gerekir. Tabii ki bu süreç anahtar değeri bulunmaya veya boş bir tablo alanına rastlayıncaya kadar devam eder. Burada dikkat edilmesi gereken en önemli nokta silme işleminde, silinen değer bulduğu tablo

indisine farklı bir yapay değer yerleştirilerek bu alanda daha önce bir anahtar değeri olduğunu göstermektir. Bir başka ifade ile eğer silinme işleminde ilgili alan, bu alana hiç değer eklenmemiş gibi bir değer alır veya boş bırakılırsa bazı arama işlemlerinin tamamlanmadan kesilmesine yol açabilir. Arama işlemine devam edilebilmesi için herhangi bir denemede aradığımız anahtar değere sahip olmayan bir tablo alanının dolu olması gerekir. Silme işleminde, boş yerine “silindi” ifadesini temsil eden farklı bir sayısal değeri yerleştirmek, arama işlemine devam edilmesini sağlar.



DİKKAT EDELİM

Hash tablosundan bir anahtar değeri silindiğinde bu alanı boş alanlardan farklı bir ifade ile göstermeye dikkat edelim.

3. AYRIK ZİNCİRLEME (SEPARATE CHAINING)

Ayrık zincirleme yönteminde hash tablosunun her bir indisinde bir bağlı listenin adresi yer alır. Hash tablosuna ekleme işleminde anahtar değeri hash fonksiyonunun ürettiği alanda yer alan

bağlı listeye yerleştirilir. Eğer herhangi bir alana bir anahtar eklenmemişse o indisin değeri NULL olarak kalır. Bu yöntemde kümeleme problemi söz konusu olmaz. Anahtarların sayısının hash

tablosundaki alanlardan daha fazla olduğu durumlarda da rahatlıkla kullanılabilir. Anahtar sayısının artması ile birlikte hash tablosundaki bazı bağlı listelerde yer alan eleman sayısı artabilir ve bu durum performansın düşmesine sebep olabilir. Arama işleminde öncelikli olarak hash fonksiyonu kullanılarak tablonun ilgili alanı bulunur. Daha sonra bu alanın gösterdiği bağlı listede doğrusal arama gerçekleştirilir. En kötü durumda, eğer tüm anahtarlar (n tane anahtar olduğunu varsayalım) aynı bağlı listeye eklenmişse arama işleminin maliyeti $O(n)$ kadar olabilir. Doğru seçilmiş bir hash fonksiyonunun bu duruma izin vermesi mümkün değildir.

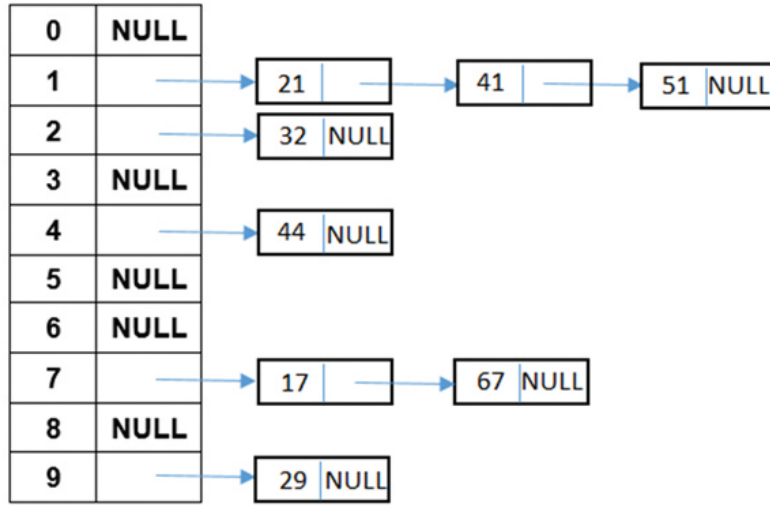
Hash fonksiyonunun aşağıdaki şekilde verildiğini düşünelim.

$$\text{hash(anahtar)} = \text{anahtar mod } m$$

Aşağıda verilen listeyi ayrı zincirleme kullanarak bir hash tablosuna ekleyelim.

32 44 21 17 41 29 67 51

Tüm değerler eklendiğinde Şekil 6.5'te yer alan hash tablosu elde edilir. Tabloda görüldüğü gibi kümeleme sorunu ortaya çıkmamıştır.



Şekil 6.5. 8 anahtarın boyutu 10 olan bir hash tablosuna ayrı zincirleme ile eklenmesi

BÖLÜM ÖZETİ

Bu bölümde bir veri yapısında temel olarak en fazla kullanılan ekleme ve arama işlemlerini $O(1)$ çalışma süresinde gerçekleştirmeyi hedefleyen hash tabloları anlatılmıştır. Bir anahtar değerinin bir hash tablosuna yerleştirilebilmesi için bu değerden tablonun boyutuna bağlı olarak geçerli bir indis değeri üretmek gerekir. Bu işlem için kullanılan farklı yöntemler ve fonksiyonlar söz konusudur. Bölme, çarpma, orta-kare ve katlama yöntemleri mevcuttur. En fazla kullanılan hash fonksiyonu bölme yöntemini temsil eden mod işleminin kullanıldığı fonksiyondur. Hash fonksiyonlarından beklenen kolay hesaplanabilme, kararlı olma ve anahtar değerleri tablo üzerinde dengeli bir şekilde dağıtma gibi özellikler vardır. Her ne kadar bir fonksiyon bu özelliklere de sahipse bazı durumlarda aynı tablo alanına birden fazla anahtarın hash değeri isabet edebilir. Bu durum çatışma olarak isimlendirilir ve bu duruma çözüm getirecek farklı yöntemler vardır. Açık adresleme yönteminde hash tablosundaki boş alanlar belirli bir düzen çerçevesinde çatışmaya neden olan anahtar değerler için kullanılır. Ayrık zincirleme yöntemi hash tablosunun indislerinin her birinde farklı bir bağlı listenin adresini tutar. Bu şekilde hash tablosunun boyutundan daha fazla sayıda anahtar değer bu veri yapısında yer almış olur.

GÖZDEN GEÇİRELİM

1. Aşağıdakilerden hangis hash tablolarında hedeflenen çalışma süresidir?

- a) $O(n^2)$
- b) $O(\log n)$
- c) $O(n)$
- d) $O(n \log n)$
- e) $O(1)$

2. Aşağıdakilerden hangisi iyi bir hash fonksiyonunun özelliklerinden değildir?

- a) Tüm hash değerlerini yaklaşık olarak eşit ihtimalle üretmesi
- b) Aynı girdi değerinin aynı çıktı değerini üretmesi
- c) Bazı değer aralıklarında daha fazla çıktı vermesi
- d) Kolay hesaplanması
- e) Hash tablosunun dışında değer üretmemesi

3. Aşağıdakilerden hangisi anahtar mod m hash fonksiyonunda m in alabileceği en iyi değerdir?

- a) m değerinin asal olması
- b) m değerinin çift olması
- c) m değerinin tek olması
- d) m değerinin tablonun boyutu olması
- e) m değerinin 5'in katı olması

4. Aşağıdaki hash fonksiyonu üretme yöntemlerinden hangisi 0 ile 1 arasında bir sabit değer kullanır?

- a) Bölme yöntemi
- b) Çarpma yöntemi
- c) Orta-Kare yöntemi
- d) Katlama yöntemi
- e) Hiçbir fonksiyonda kullanılmaz

5.

0	
1	
2	72
3	
4	82
5	
6	
7	
8	
9	

Aşağıdakilerden hangisi doğrusal sına-
ma kullanılan yukarıdaki hash tablosu
için kesinlikle söylenemez (hash(anahtar) = anahtar mod 10)?

- a) Tabloda 2 anahtar değer yer almaktadır.
- b) Tabloya en az 3 değer eklenmiştir.
- c) Tabloya 52 değeri eklenmek istenirse 3 nolu indise eklenir.
- d) Tablodan herhangi bir değer silinmiştir.
- e) Tabloda yeni anahtar değerler için yer vardır.

6. Aşağıdaki hangi açık adresleme yönteminde deneme sayısı dışında, önceden belirlenen sabit değerler kullanılmaktadır?

- a) Katlama yöntemi
- b) Doğrusal sına-
ma
- c) Karesel sına-
ma
- d) İkili özetleme
- e) Ayrık zincirleme

7. Aşağıdakilerden hangisi, 43, 64 ve 33 anahtar değerlerinin doğrusal sına kullanılarak boyutu 5 olan bir hash tablosuna eklenmiş durumudur ($\text{hash(anahtar)} = \text{anahtar} \bmod 5$)?

a)

0	33
1	
2	
3	43
4	64

b)

0	64
1	
2	
3	33
4	43

c)

0	64
1	
2	
3	43
4	33

d)

0	
1	
2	33
3	43
4	64

e)

0	
1	33
2	
3	43
4	64

8.

0	64
---	----

1	51
2	62
3	43
4	21

Yukarıda verilen hash tablosunda yer alan değerlerin, doğrusal sına kullanılarak tabloya eklenme sırası aşağıdakilerden hangisi olabilir ($\text{hash(anahtar)} = \text{anahtar} \bmod 5$)?

a) 64 51 62 43 21

b) 51 43 62 21 64

c) 21 43 62 51 64

d) 51 43 62 64 21

e) 51 64 62 21 43

9. Doğrusal sına yönteminin en önemli dezavantajı aşağıdakilerden hangisidir?

a) Hesaplama zamanı

b) Kararlı bir yaklaşım olmaması

c) Kümeleme oluşmasına sebep olması

d) Çatışma sorununu çözmemesi

e) Bazı dizi indislerine çok fazla yerleştirme yapması

10. Aşağıdaki yöntemlerden hangisinde tabloya, tablonun boyutundan daha fazla sayıda anahtar yerleştirilebilir?

a) Doğrusal sına

b) Karesel sına

c) İkili özetleme

d) Ayrık zincirleme

e) Basit özetleme fonksiyonu

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

Kaynakça

Deitel, P., Deitel, H. (2017). C++ How to Program. 10th Edition. Pearson.

Thareja R. (2014) Data Structures Using C, 2nd Edition, Oxford University Press.

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.