

BÖLÜM 9

SIRALAMA ALGORİTMALARI

ANAHTAR KAVRAMLAR

Seçmeli Sıralama, Kabarcık Sıralama, Birleşmeli Sıralama, Hızlı Sıralama

ÖĞRENME HEDEFLERİ

- 1 Kaba kuvvet ve böl-ve-fethet yöntemleri arasındaki farkları anlayabilme
- 2 Çeşitli sıralama algoritmalarının çalışma prensiplerini anlayabilme
- 3 Sıralama algoritmalarının performanslarını karşılaştırabilme

GİRİŞ

Sıralama, verilen elemanları artan veya azalan şeklinde, düzenlemek anlamına gelir. Geçmişten günümüze birçok bilim insanı, sıralama algoritmaları ile ilgilenmişlerdir. Bilgisayar bilimlerinde en çok çalışılan problemlerden biridir. Bazı problem çözümleri, sıralama algoritmalarını, alt bir fonksiyon olarak kullanır. Örneğin n elemanlı bir dizi içerisinde, bir öğenin var olup olmadığını tespit etmek veya n elemanlı bir dizi verildiğinde, öğelerin frekanslarını tespit etmek problemleri, sıralama algoritmaları yardımı ile çözülebilir.

Sıralama algoritmaları en genel haliyle tam sayı ve karşılaştırma temelli yöntemler olmak üzere ikiye ayrılırlar. Karşılaştırma sıralamaları, bir öğenin başka bir öğenin solunda mı yoksa sağında mı olması gerektiğini belirlemek için algoritmanın her adımındaki öğeleri karşılaştırır. Tam sayı ve sayma sıralamaları dizi içerisinde bir öğeden küçük kaç adet öğe olduğu hesap ederek sıralama işlemini gerçekleştirir. Bu yöntemlerin çalışabilmesi için bazı şartların sağlanması gerekmektedir aksi takdirde bu yöntemler uygulanamaz. Bu nedenle bu bölümde herhangi bir şarta bağlı kalmaksızın uygulanabilecekleri için kaba kuvvet ve böl-fethet şeklinde iki başlık altında karşılaştırma temelli sıralama algoritmaları anlatılacaktır.



ARAŞTIRALIM

Bucket ve Radix sıralama algoritmalarının hangi şartlar altında uygulanabildiği ve çalışma prensiplerini öğrenelim.



1. KABA KUVVET SIRALAMA ALGORİTMALARI

Sıralama problemini kaba kuvvet tasarım tekniği kullanarak çözmek için izlenebilecek iki strateji vardır: dizinin küçük öğelerini dizinin başındaki konumlara taşımak, ya da dizinin büyük öğeleri-

ni dizinin sonuna taşımaktır. Bu stratejiler takip edilerek oluşturulmuş algoritmalar sırasıyla Seçmeli sıralama ve Kabarcık sıralaması ismiyle literatürde yer bulmuştur.

1.1. Seçmeli Sıralama

En basit sıralama algoritmalarından biridir. Bu sıralama algoritmasının ilk adımında, dizi içerisindeki en küçük değere sahip eleman bulunur ve dizinin ilk elemanı ile yer değiştirilir. İkinci adımda ise benzer şekilde dizi içerisindeki ikinci en küçük değere sahip eleman bulunur ve dizinin ikinci elemanı ile yer değiştirilir. Bu işlem $(n-1)$ defa uygulanarak dizi elemanlarının tamamının küçükten büyüğe doğru sıralanması sağlanır. Bu

yöntem, kalan en küçük elemanı tekrar tekrar seçerek çalıştığı için seçmeli sıralama olarak adlandırılır. Eğer dizi içerisindeki elemanların büyükten küçüğe doğru bir sıralanması istenirse, her bir adımda kalan en büyük değer bulunması ve yer değiştirilmesi yeterli olacaktır. Seçmeli sıralama algoritmasının sözde kodu Şekil 9.1'de verilmiştir.

```

1.  SelectionSort([ $a_1, a_2, \dots, a_n$ ])
2.  for  $i = 1$  to  $n - 1$ 
3.       $\text{min} \leftarrow i$ 
4.      for  $j = i + 1$  to  $n$ 
5.          if  $a_j < a_{\text{min}}$ 
6.               $\text{min} \leftarrow j$ 
7.          swap  $a_i$  and  $a_{\text{min}}$ 

```

Şekil 9.1 Seçmeli Sıralama Algoritması

Seçmeli sıralama algoritması karışık sıradaki bir diziyi girdi olarak alır ve çıktı olarak küçükten büyüğe sıralı bir şekilde diziyi döndürür. Şekil 9.1'de de görüleceği gibi, algoritma iç içe iki adet for döngüsü ile sıralama işlemini gerçekleştirir. Birinci for döngüsü i değişkenine 1 değerini atar ve $n-1$ değerine ulaşınca kadar çalışır. İkinci for döngüsü j değişkenine $i + 1$ değerini atar ve n değerine ulaşınca çalışmasını tamamlar. İkinci for döngüsü j değişkenin gösterdiği dizi elemanı ile güncel minimum değerini karşılaştırır. Eğer şu an inceleme altındaki değer küçük ise minimum değeri gösteren indeks güncellenir. Değilse işleme bir sonraki dizi elemanı ile devam edilir. İkinci for döngüsünün tamamlanmasından sonra i indeksinin gösterdiği dizi elemanı ile minimum değer yer değiştirilir.

Tablo 9.1'de sıralı olmayan 8 elemanlı örneklem üzerinde Seçmeli sıralama algoritmasının çalışma adımları gösterilmiştir. Tablonun en sol sütunlarında yer alan i ve min değişken değerleri yöntemin daha kolay anlaşılması için verilmiştir. Kırmızı renkli değerler dizideki en küçük değerleri ve mavi renkli olanlar ise yer değiştirme sonucunda sıralanması tamamlamış değerleri göstermektedir. Yani bu değerler dizi içerisindeki olmaları gereken konumlara yerleştirilmişlerdir. Bundan sonra konumları kati suretle değişmeyecektir.

Tablo 9.1 Seçmeli Sıralama Algoritmasının Çalışma Adımları

i	min	1	2	3	4	5	6	7	8
1	7	8	4	6	9	2	3	1	5
2	5	1	4	6	9	2	3	8	5
3	6	1	2	6	9	4	3	8	5
4	5	1	2	3	9	4	6	8	5
5	8	1	2	3	4	9	6	8	5
6	6	1	2	3	4	5	6	8	9
7	7	1	2	3	4	5	6	8	9
8	8	1	2	3	4	5	6	8	9
		1	2	3	4	5	6	8	9

1.2. Kabarcık Sıralama

Kaba kuvvet tasarım tekniği kullanılarak geliştirilmiş bir diğer sıralama algoritması da Sabarcık sıralama algoritmasıdır. Kabarcık sıralama algoritmasının ilk adımında birinci elemandan başlayarak sonuncu elemana kadar dizinin sırasıyla komşu elemanları karşılaştırılır ve her karşılaştırmada büyük elemanlar dizinin sonuna doğru taşınırlar. Böylece, ilk adımın tamamlanmasıyla en büyük değere sahip eleman dizinin sonuna yerleştirilir. İkinci adımda birinci elemandan

başlayarak sondan bir önceki elemana kadar benzer şekilde dizinin komşuları karşılaştırılır ve yine her karşılaştırmada büyük elemanlar dizinin sonuna doğru taşınırlar. Böylece, ikinci büyük değere sahip eleman dizinin sondan bir önceki konumuna yerleştirilir. Benzer adımları dizide tek bir eleman kalıncaya kadar uygulanır. Kabarcık sıralama algoritmasının sözde kodu Şekil 9.2’de verilmiştir.

```

1. BubbleSort([a1, a2, ..., an])
2. for i = 1 to n - 1
3.   for j = 1 to n - i
4.     if aj+1 < aj
5.       swap aj+1 and aj

```

Şekil 9.2 Kabarcık Sıralama Algoritması

Şekil 9.2’de de görüleceği gibi; Kabarcık sıralama, Seçmeli sıralama algoritmasına benzer bir biçimde iç içe iki adet for döngüsü ile gerçekleştirilmektedir. Dıştaki for döngüsü 1’den n-1’e kadar çalışmaktadır. İçteki for döngüsü 1’den başla-

yıp n - i değeri kadar çalışmaktadır ve if ifadesi ile komşu elemanlar karşılaştırılarak, sırası doğru olmayan elemanların yerlerinin değiştirilmesi sağlanır.

Tablo 9.2 Kabarcık sıralama algoritmasının çalışma adımları

i	j	1	2	3	4	5	6	7	8
		8	4	6	9	2	3	1	5
1	1	4	8	6	9	2	3	1	5
1	2	4	6	8	9	2	3	1	5
1	3	4	6	8	9	2	3	1	5
1	4	4	6	8	2	9	3	1	5

1	5	4	6	8	2	3	9	1	5
1	6	4	6	8	2	3	1	9	5
1	7	4	6	8	2	3	1	5	9
2	1	4	6	8	2	3	1	5	9
2	2	4	6	8	2	3	1	5	9
2	3	4	6	2	8	3	1	5	9
2	4	4	6	2	3	8	1	5	9
2	5	4	6	2	3	1	8	5	9
2	6	4	6	2	3	1	5	8	9
3	1	4	6	2	3	1	5	8	9
3	2	4	2	6	3	1	5	8	9
3	3	4	2	3	6	1	5	8	9
3	4	4	2	3	1	6	5	8	9
3	5	4	2	3	1	5	6	8	9
4	1	2	4	3	1	5	6	8	9
4	2	2	3	4	1	5	6	8	9
4	3	2	3	1	4	5	6	8	9
4	4	2	3	1	4	5	6	8	9
5	1	2	3	1	4	5	6	8	9
5	2	2	1	3	4	5	6	8	9
5	3	2	1	3	4	5	6	8	9
6	1	1	2	3	4	5	6	8	9
6	2	1	2	3	4	5	6	8	9
7	1	1	2	3	4	5	6	8	9
		1	2	3	4	5	6	8	9

Tablo 9.2’de sıralı olmayan 8 elemanlı [8, 4, 6, 9, 2, 3, 1, 5] örneklemini üzerinde Kabarcık sıralama algoritmasının çalışma adımları gösterilmiştir. tam sayı dizisini kabarcık sıralama algoritma ile küçükten büyüğe sıralanması durumundaki bütün adımları gösterilmiştir. Kırmızı değerler karşılaştırılan komşuları ve mavi değerler konumu belirlenen dizi elemanlarını göstermektedir. İçteki for döngüsünün ilk iterasyonunun tamam-

lanmasıyla, dizi içerisindeki en büyük değer olan 9 dizinin en sonuna yerleştirilmiştir. İkinci iterasyonda ikinci en büyük değer 8 dizideki olması gereken konuma taşınmıştır. Sonraki adımlarda dizinin ön kısımlarındaki komşularına göre büyük olan değerlerin dizinin sonlarına doğru taşınmaları sağlanarak dizi küçükten büyüğe sıralı hale getirilmiştir.

2. BÖL - FETHET SIRALAMA ALGORİTMALARI

2.1. Birleşmeli Sıralama

Bir böl-ve-fethet yöntemi olan Birleşmeli sıralama algoritması 3 adımdan oluşur. Birinci adım n boyutlu $X[1...n]$ dizisini $X[1...n/2]$ ve $X[n/2+1...n]$ olmak üzere $n/2$ uzunluğunda iki parçaya böler.

Sonra her bir parça özyinelemeli olarak sıralanır. En son, sıralı olan iki parça birleştirilir. Böylece ana problem çözülmüş olur.

8	4	6	9	2	3	1	5	Sıralı olmayan tam sayı dizisi
4	6	8	9					Sıralı sol parça
				1	2	3	5	Sıralı sağ parça
1	2	3	4	5	6	8	9	Birleştirir metodu ile elde edilen sıralı dizi

Şekil 9.3 Birleşmeli Sıralama Algoritması

Şekil 9.3'te de görüleceği gibi, Birleşmeli sıralama metodu bir dizi ve iki adet tam sayı değerini girdi olarak alır. Burada p parametresi dizinin başlangıç ve r parametresi dizinin bitiş indekslerini göstermektedir. r değeri p den büyük olmak şartıyla q değişkeninin değeri p ile r değerlerinin toplamının ikiye bölünmesi ile elde edilen değer olarak belirlenir. q değeri belirlendikten sonra

Birleşmeli sıralama algoritması dizinin iki parçası üzerinde, yani dizinin p indeksinden q indeksine kadar olan kısmında ve $q + 1$ indeksinden r indeksine kadar olan kısmında iki defa özyinelemeli olarak çağrılır. Son adımda ise birleştirir metodu çağrılarak sıralı halde olan iki dizi parçası birleştirilir.

```

1. Merge( $X[1,n]$ ,  $p$ ,  $q$ ,  $r$ )
2.  $a \leftarrow q - p + 1$ 
3.  $b \leftarrow r - q$ 
4. let  $L[1, a + 1]$  and  $R[1, b + 1]$  be new arrays
5. copy  $X[p, q]$  to  $L[1, a]$ 
6. copy  $X[q+1, r]$  to  $R[1, b]$ 
7.  $L[a+1] \leftarrow \infty$ ;  $R[b+1] \leftarrow \infty$ 
8.  $i \leftarrow 1$ ;  $j \leftarrow 1$ 
9. for  $k = p$  to  $r$ 
10.   if  $L[i] \leq R[j]$ 
11.      $X[k] \leftarrow L[i]$ 
12.      $i \leftarrow i + 1$ 
13.   else
14.      $X[k] \leftarrow R[j]$ 
15.      $j \leftarrow j + 1$ 

```

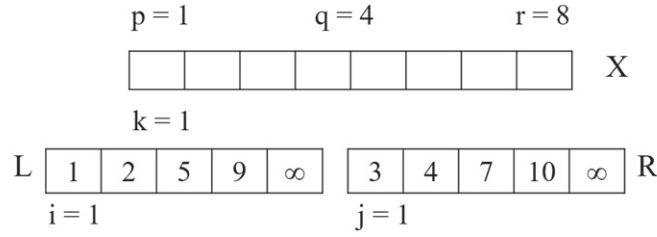
Şekil 9.4 Birleştirir Metodunun Sözde Kodu

**DİKKAT EDELİM**

Özyinelemeli (recursive) fonksiyonların tekrarlamalı(iterative) fonksiyonlara göre farklılıkları vardır. Özyinelemeli fonksiyonların çalışmasını durduracak bir koşulun muhakkak olması gerektiğine dikkat edelim.

Birleşmeli sıralama algoritmasının en kritik metodu birleştirir (Merge). Birleştirir metodu daha küçük boyutlu sıralı iki dizi elemanlarının,

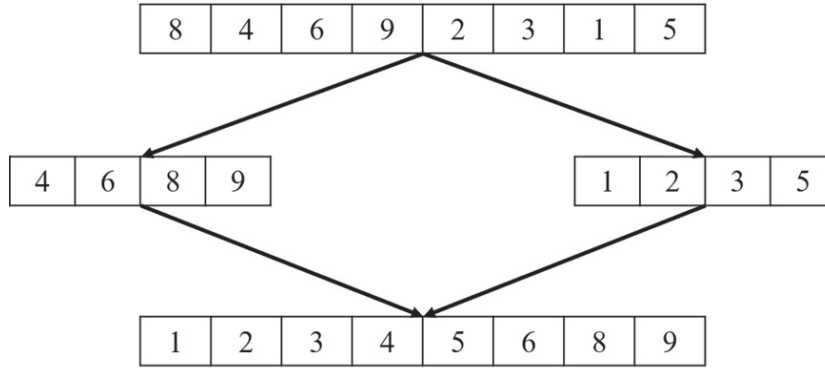
sıralı bir şekilde daha büyük bir diziye yerleştirilmesini gerçekleştirir. Aşağıda Birleştirir metodu örnek bir dizi üzerinde anlatılacaktır.



Şekil 9.5 Birleştirir Metodunun Başlangıç Durumu

Birleştirir metodu a ve b değişkenlerinin değerlerini belirledikten sonra L ve R olmak üzere iki adet yeni dizi oluşturur. X dizisindeki p ve q değişkenlerin arasında kalan değerleri L dizisine ve q+1 ve r aralığındaki değerleri R dizisine kopyalar. Her iki dizinin sonuna oldukça büyük bir değer yerleştirilir. Döngü değişkenleri olan i, j ve k'ye 1 değeri atanır. Böylece i değişkeni L dizisi-

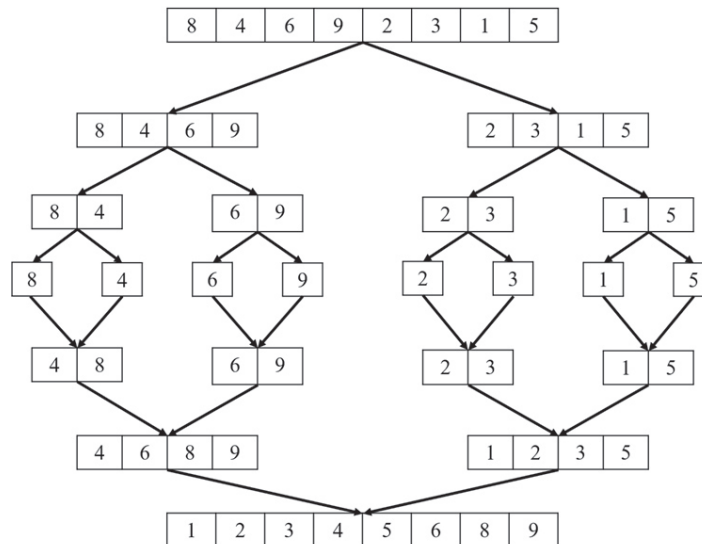
nin, j değişkeni R dizisinin ve k değişkeni X dizisinin başlangıç konumlarını göstermesi sağlanır. Sonraki adımlarda i ve j değişkenlerin gösterdiği değerler karşılaştırılır hangisi küçük ise o değer X dizisini kopyalanır. Her bir kopyalama işleminden sonra k değişkenin değeri bir artırılır. Hangi dizinin elemanı küçük ise onun değişkeni 1 artırılır diğer dizinin değişkeni değiştirilmez.



Şekil 9.6 Birleştirir Metodunun Bir Diziye Uygulanması

Şekil 9.7'de Birleşmeli sıralama algoritmasının [8, 4, 6, 9, 2, 3, 1, 5] örneklemini üzerinde nasıl çalıştığı adım adım gösterilmiştir. İlk olarak dizi elemanları bir boyutlu oluncaya kadar parçalara bölünmüş ve bu bir boyutlu dizi parçaları Birleştirir metodu yardımıyla iki boyutlu sıralı dizi par-

çaları hale getirilmiştir. Daha sonra iki boyutlu parçalar birleştirilmiş ve dört boyutlu sıralı diziler elde edilmiştir. Son olarak ta, bu dört boyutlu parçalar birleştirilmiş ve nihayetinde sıralı bir dizi elde edilmiştir.



Şekil 9.7 Birleşmeli Sıralama Algoritmasının Bir Diziye Uygulanması

2.2. Hızlı Sıralama

Hızlı sıralama algoritmasının çalışma mantığı şöyledir. Dizideki elemanlardan biri pivot olarak belirlenir. Dizinin sol tarafındaki elemanlar belirlenen pivottan küçük ve sağ tarafındaki elemanlar pivottan büyük olacak şekilde dizi iki parçaya ayrılır. Böylece pivot olarak seçilen eleman

olması gereken konuma yerleşirken pivottan küçük değerler sırasız bir şekilde pivottan önce ve büyük değerler pivottan sonraya yerleştirilir. Hem pivottan önceki hem de pivottan sonraki dizi parçaları özyinelemeli olarak sıralanır.

5	4	6	9	2	3	1	8	Sıralı olmayan tam sayı dizisi
3	2	1	4					Pivottan küçükler sol parçaya yerleştirilir
					9	6	8	Pivottan büyükler sağ parçaya yerleştirilir
				5				Pivot olması gereken konuma yerleştirilir
1	2	3	4	5	6	8	9	Sağ ve sol parçalar özyinelemeli olarak sıralanır

Şekil 9.8 Hızlı Sıralama Algoritması

Hızlı sıralama algoritmasının en kritik metodu bölümlenmedir (Partition). Şekil 9.9'da da görüleceği üzere; bu metot, X dizisi, p ve r olmak üzere 3 adet parametre alır. Girdi parametreleri p ve r dizinin başlangıç ve son indekslerini belirlemektedir. Bölümlenme metodunun, çalışmasını tamamlamasıyla, pivot olarak seçilen eleman

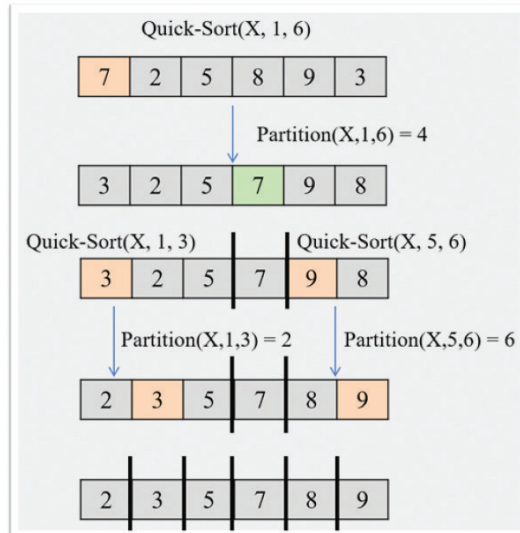
dizi içerisindeki sıralamaya göre, olması gereken yere yerleştirilir. Ayrıca pivottan küçük elemanlar, pivottan önce ve pivottan büyük elemanlar, pivottan sonra olacak şekilde dizide yerleştirilir. Bunlara ek olarak pivot elemanın dizideki konumu metot tarafından döndürülür.

```

1. Partition(X,p,r)
2. k ← ap ; s ← p
3. for i = p + 1 to r
4.   if ai < k
5.     s ← s + 1 ; swap(as, ai)
6. swap(ap, as)
7. return s

```

Şekil 9.9 Bölümlenme Metodunun Sözde Kodu



Şekil 9.10 Hızlı Sıralama Algoritmasının Örnek bir dizi üzerinde çalışma adımları

Şekil 9.10'da Hızla sıralama algoritmasının verilen bir girdi üzerinde nasıl çalıştığı adım adım sıralı diziyi nasıl oluşturduğu gösterilmiştir. Şekilde de görüleceği üzere, ilk bölümleme metodu uygulandıktan sonra pivot dizide dördüncü konuma taşınmış ve kendisinden küçüklerin sırasız bir şekilde bölümlemenin sol tarafında ve büyüklerin ise sağ tarafında olması sağlanmıştır. Hızlı sıralama metodu bir sonraki adımda özyinemeli olarak (1...3) ve (5...6) dizi parçaları için çağrılmıştır. Sol parça için pivot değeri olarak 3 ve sağ parça için pivot değeri olarak 9 alınmıştır. En son aşamada her bir parça bir elemanlı dizi parçalarından meydana geldiği için Hızlı sıralama metodu sonlandırılmıştır.

Algoritmanın çalışma süresi, bölümlemenin dengeli veya dengesiz olmasına bağlıdır. Bölüm-

leme dengeli ise, algoritma birleştirme sıralaması kadar hızlı çalışır ama dengesiz ise daha yavaş çalışabilir. En kötü durum dizinin en başta sıralı (küçükten büyüğe veya büyükten küçüğe olması fark etmez) bir şekilde olması durumunda ortaya çıkar. Bölümleme metodu her adımda sıfır elemanlı bir alt problem ve bir eleman daha az elemanlı başka bir alt problem oluşturur. Bu durumda algoritmanın çalışma süresi $O(n^2)$ olarak ortaya çıkar. Bu süre kabarcık ve seçmeli sıralama algoritmalarının çalışma sürelerine eşittir. En iyi durum yöntemin her seferinde $n / 2$ elemanlı alt problemler oluşturmasında gözlemlenir. Bu durumda hızlı sıralama algoritması, birleşmeli sıralama ile aynı zamanda sıralama işlemini gerçekleştirir.

3. SIRALAMA ALGORİTMALARININ KARŞILAŞTIRILMASI

Hızlı ve Kabarcık sıralama algoritmaları gibi bazı sıralama algoritmaları girdi değerlerine bağlı olarak farklı performans gösterebilmektedirler. Birleşmeli sıralama algoritmaları gibi bazıları ise girdi değerlerinden bağımsız olarak her durumda aynı performansı sergileyebilmektedirler. Sıralama algoritmalarının performanslarını

belirlerken gerçekleştirdikleri hem toplam karşılaştırma hem de yer değiştirme sayıları dikkate alınır. Yer değiştirme maliyeti çoğu durumda karşılaştırma maliyetinin yanında ihmal edilebilir seviyede olduğu için Tablo 9.3'de sadece sıralama algoritmalarının karşılaştırma maliyetleri açısından değerlendirmeleri gerçekleştirilmiştir.

Tablo 9.3 Sıralama Algoritmalarının Performans Karşılaştırılması

Sıralama Algoritması	Ortalama durum (Average)	En iyi durum (Best)	En kötü durum (Worst)
Seçmeli sıralama	$O(n^2)$	$O(n^2)$	$O(n^2)$
Kabarcık sıralama	$O(n^2)$	$O(n^2)$	$O(n^2)$
Birleşmeli sıralama	$O(n \cdot \log n)$	$O(n \cdot \log n)$	$O(n \cdot \log n)$
Hızlı sıralama	$O(n \cdot \log n)$	$O(n \cdot \log n)$	$O(n^2)$

Seçmeli ve Kabarcık sıralama algoritmaları girdi elemanlarının durumlarına bakmaksızın her durumda $O(n^2)$ kadar karşılaştırma yapmaları gerekmektedir ama yer değiştirme açısından durum farklıdır. Eğer dizi büyükten küçüğe sıralı ise bu durumda Kabarcık sıralaması en fazla yer değiştirme işlemi gerçekleştirmek zorundadır. Seçmeli sıralama benzer durumda daha az yer

değiştirme yapması yeterli olacaktır. Hızlı sıralama yukarıda da bahsedildiği üzere pivot elemanın seçimine bağlı olarak dengeli veya dengesiz parçalar oluşturabilmektedir. Bu durum aynı boyutlu diziler için farklı çalışma sürelerine sebep olabilmektedir. Fakat Birleşmeli sıralama için böyle bir durum söz konusu değildir.

BÖLÜM ÖZETİ

Sıralama veri işlemede ve modern bilimsel hesaplamada önemli bir rol oynar. Aynı zamanda sonraki bölümlerde anlatılacak olan Kruskal algoritması gibi birçok problemin çözümünde ara işlem olarak uygulanır. Bu bölümde Seçmeli, Kabarcık, Birleşmeli ve Hızlı olmak üzere dört adet karşılaştırma temelli sıralama algoritması açıklanmıştır. Seçmeli ve Kabarcık sıralama algoritmaları kaba kuvvet tekniğini kullanarak sıralama işlemini gerçekleştirmektedir. Kaba kuvvet yöntemleri bir problemi çözmek için problemin çözüm kümesinin bütün olası durumlarını dikkate aldığı için çoğu durumda diğer tekniklere göre yavaş çalışma sürelerine sahiptir. Böl ve fethet yöntemleri bir problemi genel olarak üç adımda çözerler. Birinci adımda problem yaklaşık eşit büyüklükte küçük parçalara ayrılır. Her bir küçük parça uygun bir şekilde çözülür ve son adımda çözülmüş olan parçalar birleştirilir. Böylece büyük parçanın çözümü elde edilmiş olur. Birleşmeli ve Hızlı sıralama algoritmaları böl-ve-fethet tekniği ile çalışmaktadırlar. Sıralama algoritmaları yaptıkları karşılaştırma ve yer değiştirme sayıları, kullandıkları ekstra bellek alanı miktarı ve kararlı olup olmadıkları açısından karşılaştırılırlar. Kabarcık ve Seçmeli sıralama en basit sıralama algoritmaları arasındadır ama en yüksek zaman karmaşıklığı değerlerine sahiptirler. Ancak, her iki algoritmanın da bellek kullanımları düşüktür. Birleşmeli ve Hızlı sıralama algoritmaları daha az zaman karmaşıklığına sahiptirler. Hızlı sıralama, diğer Birleşmeli sıralama algoritmasına göre daha az miktarda bellek kullandığı için daha verimli bir sıralama algoritmasıdır.



ARAŞTIRALIM

Bu bölümde 4 adet temel sıralama algoritması açıklanmıştır ama daha birçok sıralama algoritması mevcuttur. Burada açıklanmayan algoritmaları farklı kaynaklardan araştırılabilir.

GÖZDEN GEÇİRELİM

1. Hızlı sıralama algoritması kullanarak sekiz tam sayıdan oluşan bir diziyi sıraladığımızı varsayalım. İlk bölümlene metodunun gerçekleştirilmesinden sonra dizinin görünümü aşağıda gibi olduğuna göre hangi değer pivot olarak kullanılmıştır?

5	8	4	7	15	13	12	9
---	---	---	---	----	----	----	---

- a) 7
b) 8
c) 12
d) 13
e) 15

2. Aşağıda verilen sıralama algoritmalarından hangilerinin ortalama durum (average-case) zaman karmaşıklığı $O(n^2)$ 'dir?

- I. Seçmeli (Selection) sıralama
II. Hızlı (Quick) sıralama
III. Birleşmeli (Merge) sıralama

- a) Yalnız I
b) Yalnız II
c) I ve II
d) I ve III
e) I, II ve III

3. Aşağıda verilen sıralama algoritmalarından hangilerinin en iyi durum (best-case) zaman karmaşıklığı $O(n^2)$ 'dir?

- I. Seçmeli (Selection) sıralama
II. Hızlı (Quick) sıralama
III. Birleşmeli (Merge) sıralama

- a) Yalnız I
b) Yalnız II
c) I ve II
d) II ve III
e) I, II ve III

4. Aşağıda verilen sıralama algoritmalarından hangilerinin en kötü durum (worst-case) zaman karmaşıklığına $O(n \cdot \log n)$ 'dir?

- I. Seçmeli (Selection) sıralama
II. Birleşmeli (Merge) sıralama
III. Hızlı (Quick) sıralama

- a) Yalnız I
b) Yalnız II
c) I ve II
d) I ve III
e) I, II ve III

5. Aşağıda verilen sıralama algoritmalarından hangileri böl-ve-fethet tasarım tekniği kullanılarak geliştirilmiştir?

- I. Seçmeli (Selection) sıralama
II. Kabarcık (Bubble) sıralama
III. Birleşmeli (Merge) sıralama

- a) Yalnız I
b) Yalnız II
c) Yalnız III
d) I ve III
e) II ve III

6. Aşağıdaki dizi durumlarından hangisi küçükten büyüğe artan sıralama probleminde Kabarcık sıralama algoritması kullanıldığında, ilk döngü dizide elemanların tamamı üzerinde işlemini tamamladığında elde edilebilir?

- a) [7, 9, 3, 5, 1]
b) [5, 7, 9, 3, 1]
c) [3, 5, 9, 1, 7]
d) [3, 7, 1, 5, 9]
e) [9, 5, 7, 1, 3]

7. Aşağıda verilen tam sayı dizileri Kabarcık sıralama algoritması kullanılarak küçükten büyüğe doğru sıralanacaktır. Bu durumda en fazla yer değiştirme işlemi hangi seçenekte verilen dizi için gerçekleştirilir?

- a) [9, 7, 3, 1, 5]
- b) [5, 7, 9, 3, 1]
- c) [3, 5, 9, 1, 7]
- d) [3, 7, 1, 5, 9]
- e) [9, 7, 5, 3, 1]

9. Aşağıda verilen sıralama algoritmalarından hangileri kaba kuvvet tasarım tekniği kullanarak geliştirilmiştir?

- I. Seçmeli (Selection) sıralama
- II. Kabarcık (Bubble) sıralama
- III. Birleşmeli (Merge) sıralama

- a) Yalnız I
- b) Yalnız II
- c) I ve II
- d) II ve III
- e) I, II ve III

8. Aşağıdakilerden hangisi küçükten büyüğe artan sıralama probleminde Seçmeli sıralama algoritması kullanıldığında, ilk döngü dizide elemanların tamamı üzerinde işlemini tamamladığında elde edilebilir?

- a) [3, 5, 1, 9, 7]
- b) [3, 7, 1, 9, 5]
- c) [5, 3, 7, 9, 1]
- d) [1, 5, 9, 3, 7]
- e) [7, 3, 1, 9, 5]

10. Aşağıdakilerden hangisi bir karşılaştırma temelli sıralama algoritması değildir?

- a) Seçmeli (Selection) sıralama
- b) Kabarcık (Bubble) sıralama
- c) Birleşmeli (Merge) sıralama
- d) Hızlı (Quick) sıralama
- e) Kök (Radix) sıralama

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

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.
- ÇÖLKESİN T.R. (2019) Veri Yapıları ve Algoritmalar, Papatya Yayıncılık
- Sedgewick R., Wayne K. (2014) Algorithms, 4th Edition, Addison-Wesley
- Çamoğlu K. (2021) Algoritma, Kodlab