BÖLÜM 8

ARAMA ALGORİTMALARI

ANAHTAR KAVRAMLAR
Doğrusal Arama, İkili Arama,
Enterpolasyon Arama, Atlamalı Arama

ÖĞRENME HEDEFLERİ

1. Arama problemini tanımlar, başarılı ve başarısız arama kavramlarını bilir
2. Doğrusal arama algoritmasını uygulayabilir
3. İkili arama yöntemini kullanabilir
4. Enterpolas-yon arama algoritmasını kullanabilir
5. Atlama arama yöntemini uygulayabilir

SEKİZİNCİ BÖLÜM

GİRİŞ

Bir değerin belirli bir liste içerisinde yer alıp almadığını belirlemek için gerçekleştirilen işleme arama işlemi denir. Arama işleminin sonunda, verilen değer liste içinde bulunmuş ise arama işlemi başarılı olarak kabul edilir. Arama işleminin kullanıldığı problem veya uygulamaya bağlı olarak başarılı bir aramada, aranan değerin liste içindeki adresi çevrilir. Bu değer listeyi oluşturmak için dizi kullanılmış ise, dizinin indisi, bağlı liste kullanılmış ise aranan elemanın yer aldığı düğümün adresidir. Arama işleminde istenen değer liste içinde yer almıyor ise arama işlemi başarısız olarak kabul edilir. Bu durumu göstermek için dizi kullanılıyorsa -1, bağlı liste kullanılıyorsa NULL değeri çevrilir. Arama işlemi için farklı yöntemler kullanılabilir. Doğrusal arama ve ikili arama yöntemleri en bilinen yöntemlerdir. İkili arama yönteminde listedeki elemanların sıralı olması gerekir. Listedeki elemanların sıralı olması halinde kullanılabilecek diğer yaklaşımlar enterpolasyon arama ve atlamalı arama algoritmalarıdır.

1. ARAMA ALGORİTMALARI

Programlama problemlerinin çözümünde farklı veri yapıları kullanılabilir. Kullanılan veri yapıları ile uygulamalarda yer alan bilgilerin verimli bir şekilde işlenmesi hedeflenir. Problemin veya uygulamanın yapısına göre farklı veri tiplerinden tanımlanmış elemanlar ilgili veri yapılarına yerleştirilirler. Bazı problemlerde verilen bir elemanın bir veri yapısı içinde yer alıp almadığının belirlenmesi gerekir. Bu problem arama problemi olarak tanımlanır. Arama problemi genel bir yaklaşımla verilen bir liste içerisinde aranan bir değerin var olup olmadığının araştırılması olarak da ifade edilebilir. Arama işleminde, aranan değer liste içinde bulunursa, değerin liste içerisindeki adresi çevrilir. Bu adres kullanılan veri tipine göre farklılık gösterir. Eğer bir dizi kullanılıyor ise bir indis değeri, bağlı liste kullanılmış ise adres değeri döndürülür. Eğer aranan değer liste içinde bulunamamışsa arama işleminin başarısız olduğunu gösteren -1 veya benzeri bir indis değeri veya NULL adres değeri çevrilir. Listede yer alan elemanların sıralı olup olmamasına bağlı olarak farklı arama algoritmaları kullanılabilir.

Bu bölümde arama algoritmalarını anlatmak için listedeki elemanların bir dizi içinde yer aldığı düşünülmüştür. Tüm arama fonksiyonları bir diziyi ve aranacak değeri parametre olarak alır. Fonksiyonlar, bu parametrelere ek olarak farklı değerler de kullanabilirler. Örneğin ikili arama fonksiyonu çağrılırken dizinin ilk ve son elemanının indis değerleri, diğer fonksiyonlar çağrılırken ek olarak dizinin boyutu kullanılmıştır.

DİKKAT EDELİM

Listedeki elemanların sıralı olup olmamasına göre farklı arama algoritmaları uygulanabilir

ARAŞTIRALIM

Arama algoritmalarını, listedeki elemanların bağlı listede yer aldığı problemler üzerinde uygulanmasını araştıralım.

2. DOĞRUSAL ARAMA

Arama algoritmaları arasında en basit olan yaklaşımdır. Bu yöntemi uygulamak için listedeki elemanların sıralı düzende olması gerekli değildir. Listede yer alan ilk elemandan başlayarak aranan değer sırasıyla her eleman ile karşılaştırılır. Aranan değer liste içinde bulunursa bu değer indisi çevrilerek arama işlemi başarılı bir şekilde sona erer. Tüm liste elemanları içerisinde aranan değerin bulunamaması durumunda, bu durumu ifade etmek amacı ile -1 değeri çevrilir. Şekil 8.1'de verilen linearSearch fonksiyonu doğrusal arama yöntemini gerçekleştirmektedir. Bu fonksiyon, listenin yer aldığı diziyi, dizideki eleman sayısını ve aranan değeri parametre olarak almaktadır. Aranan değer, listede yer alan elemanlar ile ilk elemandan başlayarak karşılaştırılır. Eğer değer bulunursa, elemanın yer aldığı indis değeri çevrilir. Tüm elemanlar ile karşılaştırma sonrasında değer bulunamazsa fonksiyon tanımında görüldü gibi -1 değeri çevrilmektedir. Şekil 8.2'de verilen örnekte 48 değeri liste içinde aranıyorsa, arama işleminin sonucu olarak 7 indisi çevrilir. Eğer aynı örnek üzerinde 47 değeri aranan değer ise, bu değer listede olmadığı için tem elemanlar ile karşılaştırma işlemi sonucunda -1 değeri cevap olarak döndürülür.

```
1    int linearSearch(int ar[], int n, int value)
2    {
3        int i;
4        for (i = 0; i < n; i++) {
5            if (ar[i] == value)
6                return i;
7        }
8        return -1;
9    }
```

Şekil 8.1. Doğrusal arama fonksiyonu

0	34
1	45
2	27
3	11
4	93
5	46
6	25
7	48
8	13
9	21

Şekil 8.2. Doğrusal arama yönteminin örnek bir listede uygulanması

3. İKİLİ ARAMA

İkili arama yöntemini uygulayabilmek için listede yer alan elemanların sıralı bir düzene sahip olması gerekir. Listede yer alan elemanların artan veya azalan sırada olması durumunda ikili arama algoritması kullanılabilir. Bu bölümde verilecek ikili arama fonksiyonu listenin artan sırada olduğunu varsaymaktadır. Azalan sıralı listelerde, verilen bu fonksiyon, karşılaştırma işlemi basit bir şekilde değiştirilerek uygulanabilir. İkili arama fonksiyonu Şekil 8.3'te verilmektedir. Bu yöntemde listede yer alan en düşük indis ile en yüksek indis değerlerinin ortalaması bulunur. Bu ortalama indis değerinde yer alan eleman ile aranan değer karşılaştırılır. Eğer bir eşitlik söz konusu ise aranan değer bulunmuş olur ve ilgili indis değeri çevrilir. Eğer aranan değer bu elemandan daha büyükse, listenin ilk yarısı elenir. En küçük indis ortalama indis değerinin bir fazlası olarak güncellenir. Eğer aranan değer ortalama indiste yer alan elemandan daha küçükse listenin ikinci yarısı elenir. En büyük indisi gösteren değer ortalama indis değerinin bir eksiği olarak güncellenir. Bu güncellemelerden sonra eğer küçük indis değeri büyük indis değerinden küçük veya eşit olma durumunu sürdürüyorsa ortalama indis değeri tekrar hesaplanır ve ilgili karşılaştırmalar tekrarlanır. Eğer küçük indisi gösteren indis değerinin büyük indisi gösteren değerden büyük olma durumu söz konusu olursa aranan değerin listede olmadığı anlaşılır ve -1 indis değeri çevrilir. Şekil 8.4'te verilen örnekte 15 elemanlı sıralı bir liste yer almaktadır. 18 değerinin arandığını varsayalım. İlk hesaplanan indis değeri 7'dir. Aranan 18 değeri, hesaplanan indiste yer alan 52 değerinden küçüktür. Bu nedenle listenin ikinci yarısı elenir. En büyük indis değeri ilk hesaplanan indis değerinin 1 eksiği olan 6 olarak belirlenir. Arama işlemi listenin ilk yarısında devam

eder. Yeni hesaplanan indis 3 tür ve indiste yer alan değer 22'dir. Aranan 18 değeri 22 den küçüktür bu nedenle tekrar listenin 2. yarısı elenir. En büyük indis değeri 3 ün bir eksiği alan 2 olarak değiştirilir. Arama işleminin devam edeceği alt listede en küçük indis değeri 0 ve en büyük indis değeri 2'dir. Buradan hesaplanan yeni ortalama indis 1 olur. Bu indiste yer alan değer aranan 18 değerine eşittir ve 1 indis değeri cevap olarak çevrilir.

```
1     int binarySearch(int ar[], int low, int high, int value)
2     {
3         while (low <= high) {
4             int m = (low + high) / 2;
5             if (ar[m] == value)
6                 return m;
7             if (ar[m] < value)
8                 low = m + 1;
9             else
10                high = m - 1;
11        }
12        return -1;
13    }
```

Şekil 8.3. İkili arama fonksiyonu

0	15
1	18
2	21
3	22
4	37
5	40
6	48
7	52
8	55
9	67
10	75
11	77
12	79
13	83
14	86

Şekil 8.4. İkili arama yönteminin örnek bir listede uygulanması

ARAŞTIRALIM

İkili arama algoritmasının, listenin boyutu artarken doğrusal arama algoritmasına karşı üstünlüğünü araştıralım.

4. ENTERPOLASYON ARAMA

Enterpolasyon arama kavramı, bir telefon rehberindeki isimleri arama yöntemine benzer. Örneğin, bir telefon rehberinde Burak ismini ararken, onun rehberin ilk kısımlarında yer alacağı tahmin edilebilir. Bu nedenle listeyi her seferinde ikiye bölerek ikili arama yaklaşımın kullanmak, bu problem için doğru bir yöntem değildir. Bu ve benzeri problemlerde aranan elemanın değeri arama işlemine avantaj oluşturmak için kullanılabilir. Enterpolasyon aramada, ikili arama yönteminde olduğu gibi listenin sıralı olması gerekir. Listenin ilk elemanı ve son elemanı kullanılarak, aranan değerinin listedeki yeri konusunda yaklaşık bir indis değeri üretilir. Bu değerin hesaplanması Şekil 8.5'te yer alan interpolationSearch fonksiyonunda 11. satırda verilmiştir. Eğer üretilen bu indis değerinde yer alan değer aranan değere eşitse arama işlemi başarılı olarak tamamlanmış olur ve indis değeri çevrilir. Eğer hesaplanan indis değerinde yer alan değer aranan değerden küçükse listenin ilk elemanını gösteren indis değeri hesaplanan indis değerinin bir fazlası olarak belirlenir. Hesaplanan indis değerinin gösterdiği değer aranan değerden büyükse listenin son elemanını gösteren indis değeri hesaplanan indis değerinin bir eksiği olarak atanır. Bu şekilde arama işleminin gerçekleştiği aralık daraltılır. Enterpolasyon arama yöntemini uygulayan, interpolationSearch fonksiyonu listenin yer aldığı diziyi, listenin eleman sayısını ve aranan değeri parametre olarak almaktadır.

```
1   int interpolationSearch(int ar[], int n, int value)
2   {
3       int low = 0, high = (n - 1);
4       while (low <= high && value >= ar[low] && value <= ar[high]) {
5           if (low == high)
6           {
7               if (ar[low] == value) return low;
8                   return -1
9           }
10          int pos;
11          pos = low + (((double)(high - low) / (ar[high] - ar[low])) * (value - ar[low]));
12
13          if (ar[pos] == value)
14              return pos;
15
16          if (ar[pos] < value)
17              low = pos + 1;
18          else
19              high = pos - 1;
20      }
21      return -1;
22  }
```

Şekil 8.5. Enterpolasyon arama fonksiyonu

ARAŞTIRALIM

Sıralı listelerde kullanılabilecek farklı arama algoritmalarını araştıralım.

5. ATLAMALI ARAMA

Listemizin sıralı olması durumunda kullanabileceğimiz bir diğer arama yöntemi atlamalı arama yöntemidir. Bu yöntemde listede yer alan tüm elemanlar taranmak zorunda değildir. Listede yer alan bir eleman aranan değer ile karşılaştırılır. Karşılaştırılan bu eleman aranan değerden küçükse belirli sayıda eleman atlanarak farklı bir eleman ile karşılaştırma gerçekleştirilir. Karşılaştırılan yeni eleman aranan değerden büyükse, aranan değerin yer alabileceği bir sınır elde edilmiş olur. Aranan değerin, bir önce karşılaştırılan değer ile son karşılaştırılan değer arasında olduğu anlaşılmış olur. Karşılaştırılan değerin daha küçük olması durumunda yeniden belirli sayıda eleman atlanarak işlem tekrar edilir. Atlama miktarı listedeki eleman sayısına bağlı olarak ve genellikle eleman sayısının karekökü olarak hesaplanır. Aranan değerin yer aldığı aralık bulunduğunda bu aralık içerisinde doğrusal arama gerçekleştirerek arama işlemi tamamlanır. Şekil 8.6'da atlamalı arama yöntemini uygulayan jumpSearch fonksiyonu verilmektedir. Şekil 8.7'de yer alan sıralı listede 49 değerini arayalım. Sekil 8.6'da verilen jumpSearch fonksiyonunda görüldüğü gibi atlama miktarı listedeki eleman sayısının karekökü olarak belirlenir. Bu örnekte eleman sayısı 10 olarak verilmiştir ve atlama miktarı 3 olarak hesaplanır. Aranan 49 değeri öncelikli olarak 3. eleman 29 ile karşılaştırılır. 29 aranan değerden küçük olduğu için, bir sonraki karşılaştırma işlemi atlama miktarı kadar eleman sonrasında 6. eleman olan 41 ile gerçekleştirilir. 41'de aranan değer 49'dan küçüktür ve karşılaştırma sırası 9. eleman olan 55 değerine gelir. 55 aranan değer 49'dan büyüktür ve aranan değerin yer aldığı aralık belirlenmiş olur. Bu aralık içerisinde doğrusal arama işlemi ile 49 değerinin yer aldığı indis değeri bulunur ve 7 cevap olarak döndürülür.

```
1   int jumpSearch(int ar[], int n, int value)
2   {
3       int step = sqrt(n);
4       int prev = 0;
5       while (ar[min(step, n)-1] < value)
6       {
7           prev = step;
8           step += sqrt(n);
9           if (prev >= n)
10              return -1;
11      }
12      while (ar[prev] < value)
13      {
14          prev++;
15          if (prev == min(step, n))
16              return -1;
17      }
18      if (ar[prev] == value)
19          return prev;
20
21      return -1;
22  }
```

Şekil 8.6. Atlama arama fonksiyonu

0	12
1	25
2	29
3	33
4	37
5	41
6	45
7	49
8	55
9	64

Şekil 8.7. Atlamalı arama yönteminin örnek bir listede uygulanması

ARAŞTIRALIM

Atlamalı arama algoritmasında kullanılan atlama miktarını farklı şekilde hesaplayan yaklaşımları araştıralım.

DİKKAT EDELİM

İkili arama, enterpolasyan arama ve atlamalı arama algoritmaları sadece sıralı listelerde kullanılabilir.

BÖLÜM ÖZETİ

Bu bölümde temel arama algoritmaları anlatılmıştır. Verilen liste içinde bir değerin yer alıp almadığının araştırılması işlemi olarak tanımlanan arama problemi, listenin sıralı olup olmamasına bağlı olarak farklı yöntemler ile çözülebilir. Liste sıralı değil ise temel yaklaşım ilk elemandan başlayarak tüm elemanların aranan değere eşit olup olmadığını kontrol etmek olacaktır. Doğrusal arama olarak isimlendirilen bu çözüm bir anlamda kaba kuvvet algoritmasıdır. Eğer verilen liste sıralı ise ikili arama, enterpolasyon arama ve atlamalı arama gibi farklı çözümler kullanılabilir. İkili arama O(logn) çalışma zamanında arama işlemini tamamlamayı hedefleyen bir algoritmadır. Her karşılaştırma sonucunda listenin yarısı elenir. Enterpolasyan arama yönteminde, sıralı listenin ilk ve son elemanlarının değerleri kullanılarak aranan değerin liste içindeki yeri tahmin edilmeye çalışılır. Bu arama yönteminde aranan elemanın sayısal değerinin, elemanın listenin hangi bölgesinde yer alacağına yönelik bir avantaj üretmesi beklenir. Tabii ki bu tahminde listenin ilk ve son elemanlarının değerleri de büyük önem taşımaktadır. Aranan değer bulunamazsa değerin hesaplanan indiste yer alan değer den büyük veya küçük olmasına bağlı olarak oluşturulan alt listede aynı yöntemle arama işlemi devam eder. Sıralı listelerde kullanılan diğer bir yöntem atlamalı arama algoritmasıdır. Bu yöntemde aranan değer ile karşılaştırma işlemi gerçekleştirilecek elemanlar bir atlama değeri kullanılarak belirlenir. Atlama miktarı eleman sayısının karekökü hesaplanarak elde edilir. Bu karşılaştırmalar sonucunda aranan değerin yer aldığı aralık belirlenir. Bu aralık içinde doğrusal arama ile aranan değerin yer aldığı indis belirlenir.

GÖZDEN GEÇİRELİM

1. **Aşağıdaki arama algoritmalarından hangisinde listedeki elemanların sıralı olması gerekmez?**
   a) Doğrusal arama
   b) İkili arama
   c) Enterpolasyon arama
   d) Atlamalı arama
   e) Hiçbiri

2. **Aşağıdaki arama algoritmalarından hangisinde karşılaştırma işleminde, eğer aranan değer bulunamazsa listedeki elemanların yarısı elenir?**
   a) Enterpolasyon arama
   b) Hepsi
   c) İkili arama
   d) Doğrusal arama
   e) Atlamalı arama

3. **Atlamalı arama algoritmasında atlama miktarı aşağıdaki yöntemlerden hangisi ile hesaplanabilir?**
   a) Listedeki eleman sayısının yarısı
   b) Listedeki eleman sayısının karekökü
   c) Atlama miktarı her zaman 1 olarak kullanılır
   d) Atlama miktarı her zaman 3 olarak kullanılır
   e) Atlama miktarı her zaman 5 olarak kullanılır

4. **Doğrusal arama algoritmasının en kötü çalışma zamanı aşağıdakilerden hangisidir?**
   a) O(logn)
   b) $O(n^2)$
   c) O(n)
   d) O(1)
   e) $O(n^3)$

5. **İkili arama algoritmasında 1024 elemana sahip bir dizide en kötü çalışma zamanında yaklaşık kaç karşılaştırma gerekir?**
   a) 1
   b) 10
   c) 100
   d) 500
   e) 1024

6. **Arama işleminde, tahmin için dizinin ilk ve son elemanlarının değerini kullanan arama algoritması aşağıdakilerden hangisidir?**
   a) Hiçbiri
   b) Doğrusal arama
   c) Atlamalı arama
   d) Enterpolasyon arama
   e) İkili arama

7.

0	64
1	55
2	37
3	21
4	93
5	36
6	15
7	88
8	23
9	41

Yukarıda verilen listede doğrusal arama gerçekleştirildiği varsayılsın. 21 değerini bulmak için kaç karşılaştırma işlemi gerekir?

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

8.

0	22
1	35
2	39
3	43
4	47
5	51
6	55
7	59
8	65
9	74
10	87

Yukarıda verilen listede ikili arama gerçekleştirildiği varsayılsın. 65 değerini bulmak için kaç karşılaştırma işlemi gerekir?

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

9. **İkili arama algoritmasında hedeflenen çalışma zamanı aşağıdakilerden hangisidir?**

a) O(1)
b) O(logn)
c) O(n)
d) O(nlogn)
e) $O(n^2)$

10. **Doğrusal arama algoritmasında en iyi çalışma zamanı aşağıdakilerden hangisidir?**

a) $O(n^2)$
b) O(nlogn)
c) O(n)
d) O(logn)
e) O(1)

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

Kaynakça

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.