Bölüm 4: Arama Algoritmaları

Bölüm 4: Arama Algoritmaları · Çalışma Soruları

Doğrusal ve bekçili arama | İkili arama | Hash ile arama | Ara değer arama | Atlayarak arama | Üstel arama | Karşılaştırma

Sercan KÜLCÜ · Giresun Üniversitesi · Bilgisayar Mühendisliği

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü:
https://sercankulcu.github.io/files/algorithms/slides/Bolum_04_Arama_Algoritmalari_Calisma_Sorulari.html

ÇIKTI SORUSU · DOĞRUSAL ARAMA

Soru 1: Kaç Karşılaştırma?

`dizi = {33, 21, 12, 7, 8, 1, 3, 44, 56, 18}` (n = 10) için `dogrusalArama(dizi, 10, 44)` ve `dogrusalArama(dizi, 10, 17)` ne döndürür? Her birinde `dizi[i] != aranan` kaç kez değerlendirilir?

```
int dogrusalArama(int[] dizi, int n,
                  int aranan) {
  int i = 0;
  while (i < n && dizi[i] != aranan)
    i++;
  if (i < n)
    return i;
  else
    return -1;
}
```

▶ **Cevabı göster**

Soru 2: Bekçili Arama

`gozcuIleAra(dizi, 5, 7)` çağrısı `dizi = {12, 5, 9, 20, 7}` için çalışıyor. Döngü nerede durur, metot ne döndürür?

```
1  int gozcuIleAra(int[] dizi, int n,
2                  int aranan) {
3    if (n == 0) return -1;
4    int sonEleman = dizi[n - 1];
5    dizi[n - 1] = aranan;     // bekçi
6    int i = 0;
7    while (dizi[i] != aranan)
8      i++;
9    dizi[n - 1] = sonEleman;  // geri koy
10   if (i < n - 1 || sonEleman == aranan)
11     return i;
12   return -1;
13 }
```

Adım 1 / 8

DEĞİŞKENLER

—

ÇIKTI

Soru 2: Bekçili Arama

`gozcuIleAra(dizi, 5, 7)` çağrısı `dizi = {12, 5, 9, 20, 7}` için çalışıyor. Döngü nerede durur, metot ne döndürür?

```
1  int gozcuIleAra(int[] dizi, int n,
2                  int aranan) {
3    if (n == 0) return -1;
▶    int sonEleman = dizi[n - 1];
▶    dizi[n - 1] = aranan;     // bekçi
6    int i = 0;
7    while (dizi[i] != aranan)
8      i++;
9    dizi[n - 1] = sonEleman;  // geri koy
10   if (i < n - 1 || sonEleman == aranan)
11     return i;
12   return -1;
13 }
```

DEĞİŞKENLER

`sonEleman = 7` `dizi = [12, 5, 9, 20, 7]`

ÇIKTI



Aranan 7, son öğe de 7. sonEleman = 7 saklanır, son hücreye bekçi olarak aranan (7) yazılır; dizi görünürde değişmez.

Adım 2 / 8

Soru 2: Bekçili Arama

`gozcuIleAra(dizi, 5, 7)` çağrısı `dizi = {12, 5, 9, 20, 7}` için çalışıyor. Döngü nerede durur, metot ne döndürür?

```
1  int gozcuIleAra(int[] dizi, int n,
2                  int aranan) {
3    if (n == 0) return -1;
4    int sonEleman = dizi[n - 1];
5    dizi[n - 1] = aranan;     // bekçi
▶    int i = 0;
▶    while (dizi[i] != aranan)
▶      i++;
9    dizi[n - 1] = sonEleman;  // geri koy
10   if (i < n - 1 || sonEleman == aranan)
11     return i;
12   return -1;
13 }
```

DEĞİŞKENLER

`i = 0` `dizi[i] = 12`

ÇIKTI



dizi[0] = 12 ≠ 7 → i = 1. Döngüde yalnızca **bir** karşılaştırma var; `i < n` testi yok.

Adım 3 / 8

Soru 2: Bekçili Arama

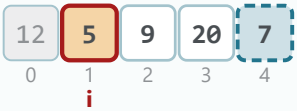
`gozcuIleAra(dizi, 5, 7)` çağrısı `dizi = {12, 5, 9, 20, 7}` için çalışıyor. Döngü nerede durur, metot ne döndürür?

```
1  int gozcuIleAra(int[] dizi, int n,
2                  int aranan) {
3    if (n == 0) return -1;
4    int sonEleman = dizi[n - 1];
5    dizi[n - 1] = aranan;      // bekçi
▶    int i = 0;
▶    while (dizi[i] != aranan)
▶      i++;
9    dizi[n - 1] = sonEleman;  // geri koy
10   if (i < n - 1 || sonEleman == aranan)
11     return i;
12   return -1;
13 }
```

DEĞİŞKENLER

`i = 1` `dizi[i] = 5`

ÇIKTI



dizi[1] = 5 ≠ 7 → i = 2.

Adım 4 / 8

Soru 2: Bekçili Arama

`gozcuIleAra(dizi, 5, 7)` çağrısı `dizi = {12, 5, 9, 20, 7}` için çalışıyor. Döngü nerede durur, metot ne döndürür?

```
1  int gozcuIleAra(int[] dizi, int n,
2                  int aranan) {
3    if (n == 0) return -1;
4    int sonEleman = dizi[n - 1];
5    dizi[n - 1] = aranan;      // bekçi
▶    int i = 0;
▶    while (dizi[i] != aranan)
▶      i++;
9    dizi[n - 1] = sonEleman;  // geri koy
10   if (i < n - 1 || sonEleman == aranan)
11     return i;
12   return -1;
13 }
```



dizi[2] = 9 ≠ 7 → i = 3.

Adım 5 / 8

DEĞİŞKENLER

`i = 2` `dizi[i] = 9`

ÇIKTI

Soru 2: Bekçili Arama

`gozcuIleAra(dizi, 5, 7)` çağrısı `dizi = {12, 5, 9, 20, 7}` için çalışıyor. Döngü nerede durur, metot ne döndürür?

```
1  int gozcuIleAra(int[] dizi, int n,
2                  int aranan) {
3    if (n == 0) return -1;
4    int sonEleman = dizi[n - 1];
5    dizi[n - 1] = aranan;      // bekçi
▶    int i = 0;
▶    while (dizi[i] != aranan)
▶      i++;
9    dizi[n - 1] = sonEleman;  // geri koy
10   if (i < n - 1 || sonEleman == aranan)
11     return i;
12   return -1;
13 }
```

DEĞİŞKENLER

`i = 3` `dizi[i] = 20`

ÇIKTI



dizi[3] = 20 ≠ 7 → i = 4.

Adım 6 / 8

Soru 2: Bekçili Arama

`gozcuIleAra(dizi, 5, 7)` çağrısı `dizi = {12, 5, 9, 20, 7}` için çalışıyor. Döngü nerede durur, metot ne döndürür?

```
1  int gozcuIleAra(int[] dizi, int n,
2                  int aranan) {
3    if (n == 0) return -1;
4    int sonEleman = dizi[n - 1];
5    dizi[n - 1] = aranan;     // bekçi
6    int i = 0;
▶    while (dizi[i] != aranan)
8      i++;
▶    dizi[n - 1] = sonEleman;  // geri koy
10   if (i < n - 1 || sonEleman == aranan)
11     return i;
12   return -1;
13 }
```



dizi[4] = 7 (bekçi) → döngü durur. Son hücreye sonEleman geri yazılır.

Adım 7 / 8

DEĞİŞKENLER

`i = 4` `dizi[i] = 7`

ÇIKTI

Soru 2: Bekçili Arama

`gozcuIleAra(dizi, 5, 7)` çağrısı `dizi = {12, 5, 9, 20, 7}` için çalışıyor. Döngü nerede durur, metot ne döndürür?

```
1  int gozcuIleAra(int[] dizi, int n,
2                  int aranan) {
3    if (n == 0) return -1;
4    int sonEleman = dizi[n - 1];
5    dizi[n - 1] = aranan;     // bekçi
6    int i = 0;
7    while (dizi[i] != aranan)
8      i++;
9    dizi[n - 1] = sonEleman;  // geri koy
▶    if (i < n - 1 || sonEleman == aranan)
▶      return i;
12   return -1;
13 }
```

DEĞİŞKENLER

`i = 4` `sonEleman = 7`

ÇIKTI

```
4
```



i < n − 1 yanlış (4 < 4), ama `sonEleman == aranan` doğru → **4** döner. Bu ikinci koşul olmasaydı gerçek son öğe "bulunamadı" sayılırdı. 4 aransaydı aynı döngü i = 4'te durur ve −1 dönerdi.

Adım 8 / 8

ÇIKTI SORUSU · İKİLİ ARAMA

Soru 3: Yazdırılan Orta Öğeler

Dizi [2, 3, 4, 7, 8, 9, 10, 21, 22, 31] iken aranan 40 olursa aşağıdaki kod ekrana ne yazdırır ve ne döndürür?

```
int baslangic = 0;
int bitis = dizi.length - 1;
while (baslangic <= bitis) {
  int orta = baslangic + (bitis - baslangic) / 2;
  System.out.println(dizi[orta]);
  if (dizi[orta] == aranan) {
    return orta;
  } else if (dizi[orta] > aranan) {
    bitis = orta - 1;
  } else {
    baslangic = orta + 1;
  }
}
return -1;
```

▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · İKİLİ ARAMA

Soru 4: İki Milyon Öğe

Sıralı 2.000.000 öğeli bir dizide ikili arama en kötü durumda kaç tur (orta karşılaştırması) yapar?

A) 20

B) 22

C) 21

D) 1.000.000

ÇOKTAN SEÇMELİ · İKİLİ ARAMA

Soru 5: ±1 Unutulursa

İkili aramada `son = orta - 1;` yerine `son = orta;` yazılırsa (döngü koşulu `bas <= son`) ne olabilir?

A) Hiçbir şey değişmez; yalnızca bazen bir tur fazla yapılır.

B) Dizi sıralı değilse yanlış sonuç döner, sıralıysa doğru sonuç döner.

C) bas == son iken öğe yoksa aralık küçülmez: sonsuz döngü.

D) Derleyici sonsuz döngüyü fark edip hata verir.

KLASİK SORU · İKİLİ ARAMA

Soru 6: Örtük Arama Ağacı

[4, 9, 13, 20, 26, 31, 40] dizisinde ikili aramanın karşılaştırma ağacını çizin. Hangi öğeler 1, 2 ve 3 karşılaştırmada bulunur? Başarılı aramalarda ortalama kaç karşılaştırma yapılır?

▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · İKİLİ ARAMA

Soru 7: İlk Konumu Bulmak

Tekrarlı öğeler içeren sıralı dizide aranan değerin **ilk** konumunu bulmak için eşitlik durumunda (`dizi[orta] == aranan`) ne yapılmalıdır?

A) `return orta;` : ilk eşleşme zaten ilk konumdur.

B) `bas = orta + 1;` ile sağa devam edilir.

C) Ortadan sola doğrusal tarama yapılıp ilk konum döndürülür.

D) `sonuc = orta; son = orta - 1;` ile sola devam edilir.

ÇIKTI SORUSU · HASH İLE ARAMA

Soru 8: Karakter Toplamı Hash'i

Sunumdaki `hashFonksiyonu` m = 10 ile "kar", "ark" ve "kara" anahtarları için hangi kovaları verir? (Unicode: a = 97, k = 107, r = 114)

```
int hashFonksiyonu(String anahtar, int m) {
  int toplam = 0;
  for (char k : anahtar.toCharArray())
    toplam += k;
  return toplam % m;
}
```

▶ **Cevabı göster**

Soru 9: Doğrusal Yoklama ile Ekleme

Boş tabloya (m = 10) 23, 43, 13, 27, 37 sırayla `hashEkle` ile ekleniyor. Her anahtar hangi kovaya yerleşir? Ardından 33 aranırsa kaç kova yoklanır?

```
1 void hashEkle(Integer[] tablo,
2               int anahtar) {
3   int m = tablo.length;     // m > 0
4   int indis = Math.floorMod(anahtar, m);
5   while (tablo[indis] != null)
6     indis = (indis + 1) % m; // sonraki
7   tablo[indis] = anahtar;
8 }
```

Adım 1 / 8

DEĞİŞKENLER

—

ÇIKTI

Soru 9: Doğrusal Yoklama ile Ekleme

Boş tabloya (m = 10) 23, 43, 13, 27, 37 sırayla `hashEkle` ile ekleniyor. Her anahtar hangi kovaya yerleşir? Ardından 33 aranırsa kaç kova yoklanır?

```
▶ void hashEkle(Integer[] tablo,
▶                int anahtar) {
▶   int m = tablo.length;     // m > 0
▶   int indis = Math.floorMod(anahtar, m);
5   while (tablo[indis] != null)
6     indis = (indis + 1) % m; // sonraki
7   tablo[indis] = anahtar;
8 }
```

DEĞİŞKENLER

`m = 10` `tablo = boş`

ÇIKTI



Boş tablo, m = 10, h(x) = x % 10. Sırayla 23, 43, 13, 27, 37 eklenecek; dolu kovada bir sonrakine geçilir (doğrusal yoklama).

Adım 2 / 8

Soru 9: Doğrusal Yoklama ile Ekleme

Boş tabloya (m = 10) 23, 43, 13, 27, 37 sırayla `hashEkle` ile ekleniyor. Her anahtar hangi kovaya yerleşir? Ardından 33 aranırsa kaç kova yoklanır?

```
1 void hashEkle(Integer[] tablo,
2                int anahtar) {
3   int m = tablo.length;     // m > 0
4   int indis = Math.floorMod(anahtar, m);
▶   while (tablo[indis] != null)
6     indis = (indis + 1) % m; // sonraki
▶   tablo[indis] = anahtar;
8 }
```



h(23) = 3: kova boş → 23 buraya yerleşir.

Adım 3 / 8

DEĞİŞKENLER

`anahtar = 23` `indis = 3`

ÇIKTI

Soru 9: Doğrusal Yoklama ile Ekleme

Boş tabloya (m = 10) 23, 43, 13, 27, 37 sırayla `hashEkle` ile ekleniyor. Her anahtar hangi kovaya yerleşir? Ardından 33 aranırsa kaç kova yoklanır?

```
1 void hashEkle(Integer[] tablo,
2                int anahtar) {
3   int m = tablo.length;     // m > 0
4   int indis = Math.floorMod(anahtar, m);
▶   while (tablo[indis] != null)
▶     indis = (indis + 1) % m; // sonraki
▶   tablo[indis] = anahtar;
8 }
```



h(43) = 3 dolu → 4 yoklanır; ilk boş kova **4**. (2 yoklama)

Adım 4 / 8

DEĞİŞKENLER

`anahtar = 43` `indis = 4`

ÇIKTI

Soru 9: Doğrusal Yoklama ile Ekleme

Boş tabloya (m = 10) 23, 43, 13, 27, 37 sırayla `hashEkle` ile ekleniyor. Her anahtar hangi kovaya yerleşir? Ardından 33 aranırsa kaç kova yoklanır?

```
1 void hashEkle(Integer[] tablo,
2                int anahtar) {
3   int m = tablo.length;     // m > 0
4   int indis = Math.floorMod(anahtar, m);
▶   while (tablo[indis] != null)
▶     indis = (indis + 1) % m; // sonraki
▶   tablo[indis] = anahtar;
8 }
```



h(13) = 3 dolu → 4, 5 yoklanır; ilk boş kova **5**. (3 yoklama)

Adım 5 / 8

DEĞİŞKENLER

`anahtar = 13` `indis = 5`

ÇIKTI

Soru 9: Doğrusal Yoklama ile Ekleme

Boş tabloya (m = 10) 23, 43, 13, 27, 37 sırayla `hashEkle` ile ekleniyor. Her anahtar hangi kovaya yerleşir? Ardından 33 aranırsa kaç kova yoklanır?

```
1 void hashEkle(Integer[] tablo,
2                int anahtar) {
3   int m = tablo.length;     // m > 0
4   int indis = Math.floorMod(anahtar, m);
▶   while (tablo[indis] != null)
6     indis = (indis + 1) % m; // sonraki
▶   tablo[indis] = anahtar;
8 }
```



h(27) = 7: kova boş → 27 buraya yerleşir.

Adım 6 / 8

DEĞİŞKENLER

`anahtar = 27` `indis = 7`

ÇIKTI

Soru 9: Doğrusal Yoklama ile Ekleme

Boş tabloya (m = 10) 23, 43, 13, 27, 37 sırayla `hashEkle` ile ekleniyor. Her anahtar hangi kovaya yerleşir? Ardından 33 aranırsa kaç kova yoklanır?

```
1 void hashEkle(Integer[] tablo,
2                int anahtar) {
3   int m = tablo.length;     // m > 0
4   int indis = Math.floorMod(anahtar, m);
▶   while (tablo[indis] != null)
▶     indis = (indis + 1) % m; // sonraki
▶   tablo[indis] = anahtar;
8 }
```

DEĞİŞKENLER

`anahtar = 37` `indis = 8`

ÇIKTI



h(37) = 7 dolu → 8 yoklanır; ilk boş kova **8**. (2 yoklama)

Adım 7 / 8

Soru 9: Doğrusal Yoklama ile Ekleme

Boş tabloya (m = 10) 23, 43, 13, 27, 37 sırayla `hashEkle` ile ekleniyor. Her anahtar hangi kovaya yerleşir? Ardından 33 aranırsa kaç kova yoklanır?

```
1 void hashEkle(Integer[] tablo,
2                int anahtar) {
3   int m = tablo.length;     // m > 0
4   int indis = Math.floorMod(anahtar, m);
▶   while (tablo[indis] != null)
▶     indis = (indis + 1) % m; // sonraki
7   tablo[indis] = anahtar;
8 }
```

DEĞİŞKENLER

`aranan = 33`

ÇIKTI

```
-1
```



hashArama(33): h = 3 → 23, 43, 13 ile karşılaştırılır, kova 6 boş → **−1**. 4 kova yoklandı: 3, 4, 5 kovaları bitişik bir küme (birincil kümelenme) oluşturdu.

Adım 8 / 8

ÇOKTAN SEÇMELİ · HASH İLE ARAMA

Soru 10: Dolu Tablo

Doğrusal yoklama kullanan `hashArama` metoduyla, **tamamen dolu** bir tabloda, tabloda bulunmayan bir anahtar aranırsa ne olur?

A) −1 döner; tüm kovalar bir kez yoklanır.

B) Döngü hiç bitmez: boş kova bulunamaz.

C) ArrayIndexOutOfBoundsException fırlatılır.

D) İlk kovadaki anahtarın indisi döner.

Soru 11: Çarpık Dağılımda Ara Değer

`araDegerArama(dizi, 8, 32)` [2, 4, 8, 16, 32, 64, 128, 256] dizisinde çalışıyor. Her turda hangi indis tahmin edilir, kaç yoklama yapılır?

```
1  int araDegerArama(int[] dizi, int n, int aranan) {
2    int bas = 0, son = n - 1;
3    while (bas <= son && aranan >= dizi[bas]
4                      && aranan <= dizi[son]) {
5      if (dizi[son] == dizi[bas]) return bas;
6      long pay = (long) (aranan - dizi[bas]) * (son -
   bas);
7      int indis = bas + (int) (pay / (dizi[son] -
   dizi[bas]));
8      if (dizi[indis] == aranan) return indis;
9      if (dizi[indis] < aranan) bas = indis + 1;
10     else son = indis - 1;
11   }
12   return -1;
13 }
```

Adım 1 / 7

DEĞİŞKENLER

—

ÇIKTI

Soru 11: Çarpık Dağılımda Ara Değer

`araDegerArama(dizi, 8, 32)` [2, 4, 8, 16, 32, 64, 128, 256] dizisinde çalışıyor. Her turda hangi indis tahmin edilir, kaç yoklama yapılır?

```
1  int araDegerArama(int[] dizi, int n, int aranan) {
▶    int bas = 0, son = n - 1;
3    while (bas <= son && aranan >= dizi[bas]
4                      && aranan <= dizi[son]) {
5      if (dizi[son] == dizi[bas]) return bas;
6      long pay = (long) (aranan - dizi[bas]) * (son -
   bas);
7      int indis = bas + (int) (pay / (dizi[son] -
   dizi[bas]));
8      if (dizi[indis] == aranan) return indis;
9      if (dizi[indis] < aranan) bas = indis + 1;
10     else son = indis - 1;
11   }
12   return -1;
13 }
```

DEĞİŞKENLER

`bas = 0` `son = 7`

ÇIKTI

2	4	8	16	32	64	128	256
0	1	2	3	4	5	6	7

Dizi sıralı ama üstel artıyor (2, 4, 8, ..., 256): düzgün dağılım varsayımı bozuk. Aranan 32.

Adım 2 / 7

Soru 11: Çarpık Dağılımda Ara Değer

`araDegerArama(dizi, 8, 32)` [2, 4, 8, 16, 32, 64, 128, 256] dizisinde çalışıyor. Her turda hangi indis tahmin edilir, kaç yoklama yapılır?

```
1  int araDegerArama(int[] dizi, int n, int aranan) {
2    int bas = 0, son = n - 1;
3    while (bas <= son && aranan >= dizi[bas]
4                      && aranan <= dizi[son]) {
▶      if (dizi[son] == dizi[bas]) return bas;
▶      long pay = (long) (aranan - dizi[bas]) * (son -
   bas);
▶      int indis = bas + (int) (pay / (dizi[son] -
   dizi[bas]));
8      if (dizi[indis] == aranan) return indis;
▶      if (dizi[indis] < aranan) bas = indis + 1;
10     else son = indis - 1;
11   }
12   return -1;
13 }
```

DEĞİŞKENLER

`bas = 0` `son = 7` `pay = 210` `indis = 0`

ÇIKTI



indis = 0 + 210 / 254 = 0 (aşağı yuvarlanır); dizi[0] = 2 < 32 → bas = 1. 256 tahmini sola çeker.

Adım 3 / 7

Soru 11: Çarpık Dağılımda Ara Değer

`araDegerArama(dizi, 8, 32)` [2, 4, 8, 16, 32, 64, 128, 256] dizisinde çalışıyor. Her turda hangi indis tahmin edilir, kaç yoklama yapılır?

```
1  int araDegerArama(int[] dizi, int n, int aranan) {
2    int bas = 0, son = n - 1;
3    while (bas <= son && aranan >= dizi[bas]
4                      && aranan <= dizi[son]) {
▶      if (dizi[son] == dizi[bas]) return bas;
▶      long pay = (long) (aranan - dizi[bas]) * (son -
   bas);
▶      int indis = bas + (int) (pay / (dizi[son] -
   dizi[bas]));
8      if (dizi[indis] == aranan) return indis;
▶      if (dizi[indis] < aranan) bas = indis + 1;
10     else son = indis - 1;
11   }
12   return -1;
13 }
```

DEĞİŞKENLER

`bas = 1` `son = 7` `pay = 168` `indis = 1`

ÇIKTI



indis = 1 + 168 / 252 = 1 (aşağı yuvarlanır); dizi[1] = 4 < 32 → bas = 2.

Adım 4 / 7

Soru 11: Çarpık Dağılımda Ara Değer

`araDegerArama(dizi, 8, 32)` [2, 4, 8, 16, 32, 64, 128, 256] dizisinde çalışıyor. Her turda hangi indis tahmin edilir, kaç yoklama yapılır?

```
1  int araDegerArama(int[] dizi, int n, int aranan) {
2    int bas = 0, son = n - 1;
3    while (bas <= son && aranan >= dizi[bas]
4                      && aranan <= dizi[son]) {
▶      if (dizi[son] == dizi[bas]) return bas;
▶      long pay = (long) (aranan - dizi[bas]) * (son -
       bas);
▶      int indis = bas + (int) (pay / (dizi[son] -
       dizi[bas]));
8      if (dizi[indis] == aranan) return indis;
▶      if (dizi[indis] < aranan) bas = indis + 1;
10     else son = indis - 1;
11   }
12   return -1;
13 }
```

DEĞİŞKENLER

`bas = 2` `son = 7` `pay = 120` `indis = 2`

ÇIKTI



indis = 2 + 120 / 248 = 2 (aşağı yuvarlanır); dizi[2] = 8 < 32 → bas = 3.

Adım 5 / 7

Soru 11: Çarpık Dağılımda Ara Değer

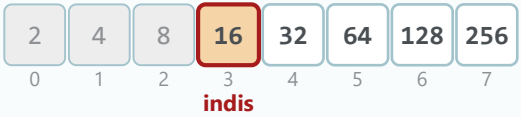
`araDegerArama(dizi, 8, 32)` [2, 4, 8, 16, 32, 64, 128, 256] dizisinde çalışıyor. Her turda hangi indis tahmin edilir, kaç yoklama yapılır?

```
1  int araDegerArama(int[] dizi, int n, int aranan) {
2    int bas = 0, son = n - 1;
3    while (bas <= son && aranan >= dizi[bas]
4                      && aranan <= dizi[son]) {
▶      if (dizi[son] == dizi[bas]) return bas;
▶      long pay = (long) (aranan - dizi[bas]) * (son -
   bas);
▶      int indis = bas + (int) (pay / (dizi[son] -
   dizi[bas]));
8      if (dizi[indis] == aranan) return indis;
▶      if (dizi[indis] < aranan) bas = indis + 1;
10     else son = indis - 1;
11   }
12   return -1;
13 }
```

DEĞİŞKENLER

`bas = 3` `son = 7` `pay = 64` `indis = 3`

ÇIKTI



indis = 3 + 64 / 240 = 3 (aşağı yuvarlanır); dizi[3] = 16 < 32 → bas = 4.

Adım 6 / 7

Soru 11: Çarpık Dağılımda Ara Değer

`araDegerArama(dizi, 8, 32)` [2, 4, 8, 16, 32, 64, 128, 256] dizisinde çalışıyor. Her turda hangi indis tahmin edilir, kaç yoklama yapılır?

```
1  int araDegerArama(int[] dizi, int n, int aranan) {
2    int bas = 0, son = n - 1;
3    while (bas <= son && aranan >= dizi[bas]
4                      && aranan <= dizi[son]) {
▶      if (dizi[son] == dizi[bas]) return bas;
▶      long pay = (long) (aranan - dizi[bas]) * (son -
   bas);
▶      int indis = bas + (int) (pay / (dizi[son] -
   dizi[bas]));
▶      if (dizi[indis] == aranan) return indis;
9      if (dizi[indis] < aranan) bas = indis + 1;
10     else son = indis - 1;
11   }
12   return -1;
13 }
```

DEĞİŞKENLER

`bas = 4` `son = 7` `pay = 0` `indis = 4`

ÇIKTI

```
4
```



indis = 4 + 0 / 224 = 4; dizi[4] = 32 → **return 4**. 5 yoklama: ikili arama 3 yoklamada (3, 5, 4) bulurdu.

Adım 7 / 7

ÇOKTAN SEÇMELİ · ARA DEĞER ARAMA

Soru 12: Ara Çarpımda Taşma

Ara değer aramada `(aranan - dizi[bas]) * (son - bas)` çarpımı için hangisi doğrudur?

A) Sonuç her zaman son − bas'tan küçük olduğu için taşma olmaz.

B) Taşma yalnızca dizi sırasızsa olur; sıralı dizide imkânsızdır.

C) Bölme önce yapılırsa (pay / payda) sonuç her zaman aynı kalır.

D) Büyük değerlerde int'i taşırabilir; long kullanılmalıdır.

Soru 13: Atlayarak Arama: Başarısız

`atlayarakAraBul(dizi, 30, 16)`, D = [3, 8, 15, 21, 27, 34, 40, 46, 52, 59, 65, 71, 78, 84, 90, 97] için çalışıyor. Hangi indisler yoklanır, ne döner?

```
1  int atlayarakAraBul(int[] dizi, int eleman,
2      int n) { if (n == 0) return -1;
3    int blok = (int) Math.sqrt(n);
4    int atlama = blok, konum = 0;
5    while (dizi[Math.min(atlama, n) - 1] < eleman) {
6      konum = atlama; atlama += blok;
7      if (konum >= n) return -1;
8    }
9    while (dizi[konum] < eleman) {
10     konum++;
11     if (konum == Math.min(atlama, n)) return -1;
12   }
13   if (dizi[konum] == eleman) return konum;
14   return -1;
15 }
```

Adım 1 / 7

DEĞİŞKENLER

—

ÇIKTI

Soru 13: Atlayarak Arama: Başarısız

`atlayarakAraBul(dizi, 30, 16)`, D = [3, 8, 15, 21, 27, 34, 40, 46, 52, 59, 65, 71, 78, 84, 90, 97] için çalışıyor. Hangi indisler yoklanır, ne döner?

```
1  int atlayarakAraBul(int[] dizi, int eleman,
2      int n) { if (n == 0) return -1;
▶    int blok = (int) Math.sqrt(n);
▶    int atlama = blok, konum = 0;
5    while (dizi[Math.min(atlama, n) - 1] < eleman) {
6      konum = atlama; atlama += blok;
7      if (konum >= n) return -1;
8    }
9    while (dizi[konum] < eleman) {
10     konum++;
11     if (konum == Math.min(atlama, n)) return -1;
12   }
13   if (dizi[konum] == eleman) return konum;
14   return -1;
15 }
```

DEĞİŞKENLER

`blok = 4` `atlama = 4` `konum = 0`

ÇIKTI

3	8	15	21	27	34	40	46	52	59	65	71	78	84	90	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

n = 16 → blok = √16 = 4. Aranan 30; her bloğun son öğesine bakılacak.

Adım 2 / 7

Soru 13: Atlayarak Arama: Başarısız

`atlayarakAraBul(dizi, 30, 16)`, D = [3, 8, 15, 21, 27, 34, 40, 46, 52, 59, 65, 71, 78, 84, 90, 97] için çalışıyor. Hangi indisler yoklanır, ne döner?

```
1  int atlayarakAraBul(int[] dizi, int eleman,
2      int n) { if (n == 0) return -1;
3    int blok = (int) Math.sqrt(n);
4    int atlama = blok, konum = 0;
▶    while (dizi[Math.min(atlama, n) - 1] < eleman) {
6      konum = atlama; atlama += blok;
7      if (konum >= n) return -1;
8    }
9    while (dizi[konum] < eleman) {
10     konum++;
11     if (konum == Math.min(atlama, n)) return -1;
12   }
13   if (dizi[konum] == eleman) return konum;
14   return -1;
15 }
```



dizi[min(4, 16) − 1] = dizi[3] = 21 < 30 → bloğu atla.

Adım 3 / 7

DEĞİŞKENLER

`blok = 4` `atlama = 4` `konum = 0`

ÇIKTI

Soru 13: Atlayarak Arama: Başarısız

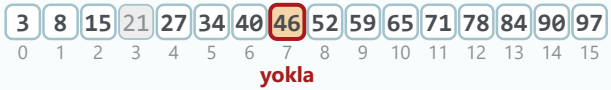
`atlayarakAraBul(dizi, 30, 16)`, D = [3, 8, 15, 21, 27, 34, 40, 46, 52, 59, 65, 71, 78, 84, 90, 97] için çalışıyor. Hangi indisler yoklanır, ne döner?

```
 1 int atlayarakAraBul(int[] dizi, int eleman,
 2     int n) { if (n == 0) return -1;
 3   int blok = (int) Math.sqrt(n);
 4   int atlama = blok, konum = 0;
▶  while (dizi[Math.min(atlama, n) - 1] < eleman) {
▶    konum = atlama; atlama += blok;
▶    if (konum >= n) return -1;
 8   }
 9   while (dizi[konum] < eleman) {
10     konum++;
11     if (konum == Math.min(atlama, n)) return -1;
12   }
13   if (dizi[konum] == eleman) return konum;
14   return -1;
15 }
```

DEĞİŞKENLER

`blok = 4` `atlama = 8` `konum = 4`

ÇIKTI



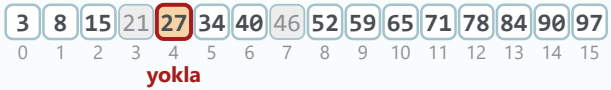
konum = 4, atlama = 8. dizi[7] = 46 ≥ 30 → atlama biter; aranan [4, 8) bloğunda olabilir.

Adım 4 / 7

Soru 13: Atlayarak Arama: Başarısız

`atlayarakAraBul(dizi, 30, 16)`, D = [3, 8, 15, 21, 27, 34, 40, 46, 52, 59, 65, 71, 78, 84, 90, 97] için çalışıyor. Hangi indisler yoklanır, ne döner?

```
1  int atlayarakAraBul(int[] dizi, int eleman,
2      int n) { if (n == 0) return -1;
3    int blok = (int) Math.sqrt(n);
4    int atlama = blok, konum = 0;
5    while (dizi[Math.min(atlama, n) - 1] < eleman) {
6      konum = atlama; atlama += blok;
7      if (konum >= n) return -1;
8    }
▶    while (dizi[konum] < eleman) {
10     konum++;
11     if (konum == Math.min(atlama, n)) return -1;
12   }
13   if (dizi[konum] == eleman) return konum;
14   return -1;
15 }
```



Doğrusal tarama: dizi[4] = 27 < 30 → konum = 5.

Adım 5 / 7

DEĞİŞKENLER

`blok = 4` `atlama = 8` `konum = 4`

ÇIKTI

Soru 13: Atlayarak Arama: Başarısız

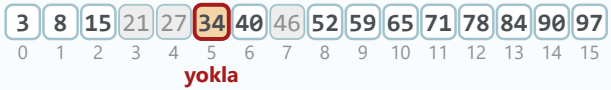
`atlayarakAraBul(dizi, 30, 16)`, D = [3, 8, 15, 21, 27, 34, 40, 46, 52, 59, 65, 71, 78, 84, 90, 97] için çalışıyor. Hangi indisler yoklanır, ne döner?

```
1  int atlayarakAraBul(int[] dizi, int eleman,
2      int n) { if (n == 0) return -1;
3    int blok = (int) Math.sqrt(n);
4    int atlama = blok, konum = 0;
5    while (dizi[Math.min(atlama, n) - 1] < eleman) {
6      konum = atlama; atlama += blok;
7      if (konum >= n) return -1;
8    }
▶    while (dizi[konum] < eleman) {
▶      konum++;
▶      if (konum == Math.min(atlama, n)) return -1;
12   }
13   if (dizi[konum] == eleman) return konum;
14   return -1;
15 }
```

DEĞİŞKENLER

`blok = 4` `atlama = 8` `konum = 5`

ÇIKTI



konum = 5 ≠ min(8, 16); dizi[5] = 34 < 30 yanlış → döngüden çıkılır.

Adım 6 / 7

Soru 13: Atlayarak Arama: Başarısız

`atlayarakAraBul(dizi, 30, 16)`, D = [3, 8, 15, 21, 27, 34, 40, 46, 52, 59, 65, 71, 78, 84, 90, 97] için çalışıyor. Hangi indisler yoklanır, ne döner?

```
 1 int atlayarakAraBul(int[] dizi, int eleman,
 2     int n) { if (n == 0) return -1;
 3   int blok = (int) Math.sqrt(n);
 4   int atlama = blok, konum = 0;
 5   while (dizi[Math.min(atlama, n) - 1] < eleman) {
 6     konum = atlama; atlama += blok;
 7     if (konum >= n) return -1;
 8   }
 9   while (dizi[konum] < eleman) {
10     konum++;
11     if (konum == Math.min(atlama, n)) return -1;
12   }
▶    if (dizi[konum] == eleman) return konum;
▶    return -1;
15 }
```

DEĞİŞKENLER

`blok = 4` `atlama = 8` `konum = 5`

ÇIKTI

```
-1
```

3	8	15	21	27	34	40	46	52	59	65	71	78	84	90	97
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

dizi[5] = 34 ≠ 30 → **−1**. Yoklanan indisler 3, 7, 4, 5: 30, 27 ile 34 arasında olurdu. En kötü durum ≈ n/m + m − 1 = 7 yoklama.

Adım 7 / 7

ÇOKTAN SEÇMELİ · ATLAYARAK ARAMA

Soru 14: En İyi Blok Boyu

Atlayarak aramada blok boyu m iken en kötü yoklama ≈ n/m + (m − 1)'dir. n = 10.000 için en iyi m ve en kötü yoklama sayısı nedir?

A) m = 100, 199 yoklama

B) m = 100, 100 yoklama

C) m = 50, 249 yoklama

D) m = 14, 14 yoklama

ÇIKTI SORUSU · ÜSTEL ARAMA

Soru 15: Üstel Aramada Yoklamalar

`ustelArama(D, 27, 16)` çağrısında (D = [3, 8, 15, 21, 27, 34, 40, 46, 52, 59, 65, 71, 78, 84, 90, 97]) hangi indisler yoklanır ve ne döner? İkili arama hangi aralıkta yapılır?

```
int ustelArama(int[] dizi, int aranan, int n) {
  if (n == 0) return -1;
  if (dizi[0] == aranan) return 0;
  int konum = 1;
  while (konum < n && dizi[konum] <= aranan)
    konum *= 2;
  int bas = konum / 2;
  int son = Math.min(konum, n - 1);
  while (bas <= son) {
    int orta = (bas + son) / 2;
    if (dizi[orta] == aranan) return orta;
    if (dizi[orta] < aranan) bas = orta + 1;
    else son = orta - 1;
  }
  return -1;
}
```

▶ **Cevabı göster**

ÇOKTAN SEÇMELİ · ÜSTEL ARAMA

Soru 16: Üstel Arama Ne Zaman?

Üstel arama hangi durumda ikili aramadan daha uygundur?

A) Dizi sırasızsa ve tek bir sorgu yapılacaksa

B) Değerler düzgün dağılmış sayısal anahtarlarsa

C) Liste boyu bilinmiyorsa ya da aranan başa yakınsa

D) Geriye doğru okumak pahalı olan bir manyetik banttaysa

ÇOKTAN SEÇMELİ · KARŞILAŞTIRMA

Soru 17: Hangi Arama?

Sıralı kayıtlar bir manyetik bantta tutuluyor; ileri okumak ucuz, geri sarmak çok pahalı. Hangi arama yöntemi en uygundur?

A) İkili arama

B) Atlayarak arama

C) Ara değer arama

D) Hash ile arama

KLASİK SORU · KARŞILAŞTIRMA

Soru 18: Doğrusal mı, İkili mi?

Doğrusal ve ikili aramayı avantaj ve dezavantajlarıyla karşılaştırın. Sırasız n öğeli bir dizide k sorgu yapılacaksa hangisini, hangi koşulda seçersiniz?

▶ **Cevabı göster**

Son

Bölüm 4: Arama Algoritmaları: Çalışma Soruları

Konu anlatımı: Bölüm 4: Arama Algoritmaları · Terim sözlüğü