Bölüm 1.3: Algoritmalar

Java ile Nesne Yönelimli Programlama

Problem çözme adımları | Sözde kod | Koda dönüştürme | Örnek algoritmalar | Elle izleme | Arama algoritmaları | Sıralama algoritmaları

Sercan KÜLCÜ · Giresun Üniversitesi · Elektrik-Elektronik Mühendisliği

GİRİŞ

Öğrenme Çıktıları

Problem çözme

Bir problemi girdi, çıktı ve test senaryolarıyla tanımlamak; çözümü önce algoritma, sonra kod olarak yazmak.

Sözde kod ve izleme

Algoritmayı sözde kodla (pseudocode) ifade etmek ve değişken tablosuyla adım adım elle izlemek.

Java'ya dönüştürme

Okuma, yazma, karar ve döngü adımlarını Java karşılıklarına çevirmek.

Temel algoritmalar

Sıralı yazdırma, faktöriyel, Fibonacci, palindrom, asal sayı ve üs alma algoritmalarını yazmak ve test etmek.

Arama

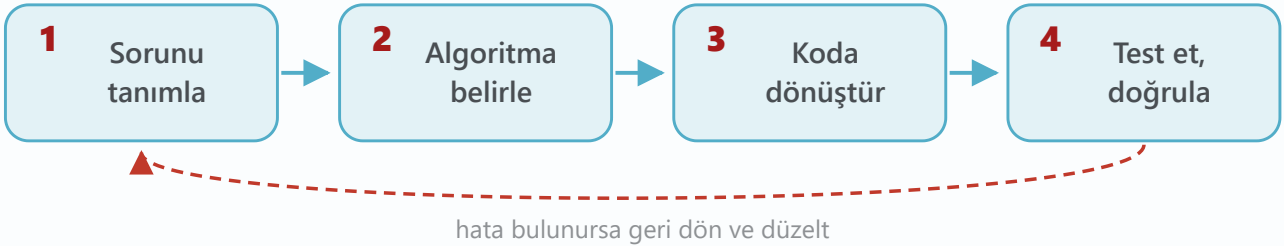
Rastgele, doğrusal ve ikili aramayı karşılaştırmak; ikili aramanın neden sıralı veri gerektirdiğini açıklamak.

Sıralama

Rastgele (karıştırma), seçimli sıralamayı ve kabarcık sıralamasını bir dizi üzerinde adım adım uygulamak.

PROBLEM ÇÖZME

Önce Algoritma, Sonra Uygulama



- **Adım 1:** Sorunu net bir şekilde tanımlamak başarının anahtarıdır: ne verilecek, ne istenecek?
- **Adım 2:** Algoritma, sorunu çözmek için adım adım yönergeler sunar. Doğru algoritmayı seçmek programın verimli çalışmasını sağlar.
- **Adım 3:** Algoritma belirlendikten sonra seçilen programlama dilinde (bizim için Java) kodlama yapılır.
- **Adım 4:** Test senaryoları ile program doğrulanır, hatalar bulunur ve gerekirse önceki adımlara dönülür.

PROBLEM ÇÖZME

Algoritma (Algorithm) Nedir?

Algoritma: Bir problemi çözmek için izlenen, sırası belli, açık ve sonlu sayıda adımdan oluşan yönergeler dizisidir.

- Sözcük, 9. yüzyılda yaşamış matematikçi **el-Hârizmî**'nin adının Latince yazılışından gelir.
- Bilgisayar algoritmayı **anlamaz**; algoritmayı bir programlama diline çevirdiğimizde çalıştırır.
- Aynı problemi çözen birden çok algoritma olabilir; bunlar hız ve bellek bakımından karşılaştırılır.

Benzetme: Yemek tarifi bir algoritmadır. Malzemeler *girdi*, yapılış adımları *algoritma*, ortaya çıkan yemek *çıktı*dır. "Biraz tuz ekle" gibi belirsiz bir adım bilgisayar için yeterli değildir.

Girdi (input)
Sıfır veya daha fazla başlangıç verisi.

Çıktı (output)
En az bir sonuç üretir.

Kesinlik
Her adım açık ve tek anlamlıdır.

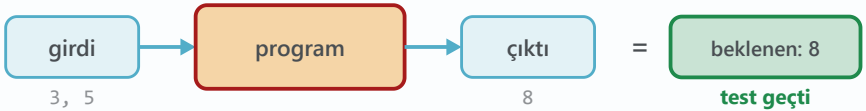
Sonluluk
Sonlu sayıda adımda biter.

Etkinlik
Her adım gerçekten yapılabilir.

PROBLEM ÇÖZME

Sorunu Tanımlama

- Programın **girdi** ve **çıktıları** doğal bir dilde (Türkçe) açıklanır.
- Programın doğru çalıştığını doğrulamak için **test senaryoları** (test cases) oluşturulur: verilen girdi için *beklenen çıktı* önceden yazılır.



Sorun: İki sayının toplamını hesaplamak

Test	Girdi	Beklenen çıktı
1	3, 5	8
2	−2, 7	5
3 (sınır)	0, 0	0

Sorun: Bir metindeki kelime sayısını bulmak

Test	Girdi	Beklenen çıktı
1	"Merhaba, dünya!"	2
2	"Bu bir denemedir."	3
3 (sınır)	"" (boş metin)	0

Üçüncü satırlar sınır durumlardır (edge case): sıfır, boş metin, negatif sayı gibi uç girdiler hataların en sık saklandığı yerlerdir.

PROBLEM ÇÖZME

Algoritma Belirleme ve Test Etme

Sorunu mantıklı bir şekilde çözmek için gerekli adımlar sırayla belirlenir.

Sorun: İki sayının toplamını hesaplama.

Algoritma:

1. İlk sayıyı al.
2. İkinci sayıyı al.
3. İki sayıyı topla.
4. Toplamı ekrana yazdır.

Algoritma oluşturulduktan sonraki adım **test etmektir**:

- Küçük ama anlamlı test senaryoları seçilir.
- Algoritma kâğıt üzerinde adım adım çalıştırılır ve her adımın sonucu bir **değişken tablosuna** yazılır (elle izleme, *dry run*).

Test 1 (girdi 3, 5) için elle izleme: ekrana 8 yazıldı, beklenen de 8

Adım	sayı1	sayı2	toplam	Ekran
1. İlk sayıyı al	3			
2. İkinci sayıyı al	3	5		
3. İki sayıyı topla	3	5	8	
4. Toplamı yazdır	3	5	8	8

PROBLEM ÇÖZME

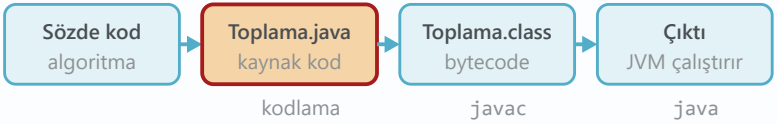
Algoritmayı İfade Etmek: Sözde Kod (Pseudocode)

Algoritma doğal dille, **sözde kodla**, akış diyagramıyla veya doğrudan bir programlama diliyle yazılabilir. Sözde kod, programlama diline benzeyen ama derleyici kuralları olmayan, insan için yazılmış bir gösterimdir; mantık hatalarını kodlamadan önce bulmayı ve programcılar arası işbirliğini kolaylaştırır.

Sözde kod	Anlamı	Java karşılığı
`Başla` … `Bitir`	Algoritmanın başı ve sonu	`main` metodunun `{ … }` gövdesi
`Oku x`	Kullanıcıdan değer al	`x = tarayici.nextInt();`
`Yaz "…", x`	Ekrana yazdır	`System.out.println("…" + x);`
`x = 5`	Değişkene değer ata	`x = 5;`
`Eğer x = y ise`	Eşitliği karşılaştır	`if (x == y)` (iki eşittir!)
`Değilse`	Koşul yanlışsa	`else`
`Döngü i = 1'den n'e kadar`	Tekrarla	`for (int i = 1; i <= n; i++)`
`Döngüyü Kır`	Döngüden hemen çık	`break;`
`%`, `ve`, `veya`	Kalan, mantıksal ve/veya	`%`, `&&`, `||`

Koda Dönüştürme (Implementation)

- Algoritmayı oluşturduktan sonraki adım **kodlamadır**.
- Programlama dili, algoritmayı bilgisayarın anlayacağı (derlenip çalıştırılabilecek) şekilde ifade eder.
- Bir programı yönetilebilir ve anlaşılır hale getirmek için kod **mantıksal parçalara** ayrılır: girdi alma, hesaplama, çıktı verme; daha büyük programlarda **fonksiyonlar**.
- Sözde koddaki her satır genellikle bir veya birkaç Java satırına karşılık gelir.



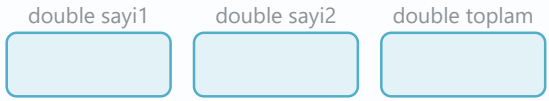
Algoritma adımı	Java
İlk sayıyı al	`sayi1 = tarayici.nextDouble();`
İkinci sayıyı al	`sayi2 = tarayici.nextDouble();`
İki sayıyı topla	`toplam = sayi1 + sayi2;`
Toplamı yazdır	`System.out.println(toplam);`

Amaç, algoritma adımı ile kod satırı arasındaki eşleşmeyi görmektir. Kütüphaneler çoğunlukla İngilizce adlar kullanır; bu derste kendi adlarımız Türkçe ama Türkçe karakter kullanılmadan seçilir.

KODA DÖNÜŞTÜRME · ADIM ADIM KOD İZLEME

Toplama Programının Adım Adım İzlenmesi

```
1 Scanner tarayici = new Scanner(System.in);
2 System.out.print("Birinci sayıyı giriniz: ");
3 double sayi1 = tarayici.nextDouble();
4 System.out.print("İkinci sayıyı giriniz: ");
5 double sayi2 = tarayici.nextDouble();
6 double toplam = sayi1 + sayi2;
7 System.out.println(sayi1 + " ile " + sayi2
8         + " toplamı: " + toplam);
9 tarayici.close();
```



Klavyeden okumak için bir `Scanner` nesnesi oluşturulur; henüz hiçbir sayı yok.

Adım 1 / 8

DEĞİŞKENLER

`tarayici = Scanner`

ÇIKTI

KODA DÖNÜŞTÜRME · ADIM ADIM KOD İZLEME

Toplama Programının Adım Adım İzlenmesi

```
1 Scanner tarayici = new Scanner(System.in);
▶ System.out.print("Birinci sayıyı giriniz: ");
3 double sayi1 = tarayici.nextDouble();
4 System.out.print("İkinci sayıyı giriniz: ");
5 double sayi2 = tarayici.nextDouble();
6 double toplam = sayi1 + sayi2;
7 System.out.println(sayi1 + " ile " + sayi2
8         + " toplamı: " + toplam);
9 tarayici.close();
```

double sayi1	double sayi2	double toplam

`print` mesajı yazar ama alt satıra geçmez; imleç mesajın hemen yanında bekler.

Adım 2 / 8

DEĞİŞKENLER

`tarayici = Scanner`

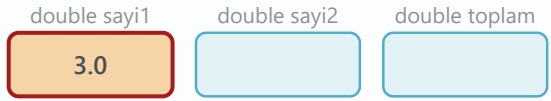
ÇIKTI

```
Birinci sayıyı giriniz:
```

KODA DÖNÜŞTÜRME · ADIM ADIM KOD İZLEME

Toplama Programının Adım Adım İzlenmesi

```
1 Scanner tarayici = new Scanner(System.in);
2 System.out.print("Birinci sayıyı giriniz: ");
▶ double sayi1 = tarayici.nextDouble();
4 System.out.print("İkinci sayıyı giriniz: ");
5 double sayi2 = tarayici.nextDouble();
6 double toplam = sayi1 + sayi2;
7 System.out.println(sayi1 + " ile " + sayi2
8         + " toplamı: " + toplam);
9 tarayici.close();
```



Kullanıcı **3** yazıp Enter'a basar; `nextDouble()` bunu 3.0 olarak okur ve `sayi1`'e koyar.

Adım 3 / 8

DEĞİŞKENLER

`tarayici = Scanner`

`sayi1 = 3.0`

ÇIKTI

```
Birinci sayıyı giriniz: 3
```

KODA DÖNÜŞTÜRME · ADIM ADIM KOD İZLEME

Toplama Programının Adım Adım İzlenmesi

```
1 Scanner tarayici = new Scanner(System.in);
2 System.out.print("Birinci sayıyı giriniz: ");
3 double sayi1 = tarayici.nextDouble();
▶ System.out.print("İkinci sayıyı giriniz: ");
5 double sayi2 = tarayici.nextDouble();
6 double toplam = sayi1 + sayi2;
7 System.out.println(sayi1 + " ile " + sayi2
8         + " toplamı: " + toplam);
9 tarayici.close();
```

double sayi1	double sayi2	double toplam
3.0		

İkinci mesaj yazılır.

Adım 4 / 8

DEĞİŞKENLER

`tarayici = Scanner`

`sayi1 = 3.0`

ÇIKTI

```
Birinci sayıyı giriniz: 3
İkinci sayıyı giriniz:
```

KODA DÖNÜŞTÜRME · ADIM ADIM KOD İZLEME

Toplama Programının Adım Adım İzlenmesi

```
1 Scanner tarayici = new Scanner(System.in);
2 System.out.print("Birinci sayıyı giriniz: ");
3 double sayi1 = tarayici.nextDouble();
4 System.out.print("İkinci sayıyı giriniz: ");
▶ double sayi2 = tarayici.nextDouble();
6 double toplam = sayi1 + sayi2;
7 System.out.println(sayi1 + " ile " + sayi2
8         + " toplamı: " + toplam);
9 tarayici.close();
```

double sayi1	double sayi2	double toplam
3.0	5.0	

Kullanıcı **5** girer; `sayi2` = 5.0.

Adım 5 / 8

DEĞİŞKENLER

`tarayici` = **Scanner**

`sayi1` = **3.0** `sayi2` = **5.0**

ÇIKTI

```
Birinci sayıyı giriniz: 3
İkinci sayıyı giriniz: 5
```

KODA DÖNÜŞTÜRME · ADIM ADIM KOD İZLEME

Toplama Programının Adım Adım İzlenmesi

```
1 Scanner tarayici = new Scanner(System.in);
2 System.out.print("Birinci sayıyı giriniz: ");
3 double sayi1 = tarayici.nextDouble();
4 System.out.print("İkinci sayıyı giriniz: ");
5 double sayi2 = tarayici.nextDouble();
▶ double toplam = sayi1 + sayi2;
7 System.out.println(sayi1 + " ile " + sayi2
8         + " toplamı: " + toplam);
9 tarayici.close();
```

double sayi1	double sayi2	double toplam
3.0	5.0	8.0

Sağ taraf önce hesaplanır: 3.0 + 5.0 = 8.0; sonuç `toplam` 'a atanır.

Adım 6 / 8

DEĞİŞKENLER

`tarayici` = **Scanner**

`sayi1` = **3.0** `sayi2` = **5.0**

`toplam` = **8.0**

ÇIKTI

```
Birinci sayıyı giriniz: 3
İkinci sayıyı giriniz: 5
```

KODA DÖNÜŞTÜRME · ADIM ADIM KOD İZLEME

Toplama Programının Adım Adım İzlenmesi

```
1 Scanner tarayici = new Scanner(System.in);
2 System.out.print("Birinci sayıyı giriniz: ");
3 double sayi1 = tarayici.nextDouble();
4 System.out.print("İkinci sayıyı giriniz: ");
5 double sayi2 = tarayici.nextDouble();
6 double toplam = sayi1 + sayi2;
▶ System.out.println(sayi1 + " ile " + sayi2
▶         + " toplamı: " + toplam);
9 tarayici.close();
```

double sayi1	double sayi2	double toplam
3.0	5.0	8.0

`+` metinleri birleştirir; sayılar metne çevrilerek tek satır oluşur.

Adım 7 / 8

DEĞİŞKENLER

`tarayici` = **Scanner**

`sayi1` = **3.0** `sayi2` = **5.0**

`toplam` = **8.0**

ÇIKTI

```
Birinci sayıyı giriniz: 3
İkinci sayıyı giriniz: 5
3.0 ile 5.0 toplamı: 8.0
```

KODA DÖNÜŞTÜRME · ADIM ADIM KOD İZLEME

Toplama Programının Adım Adım İzlenmesi

```
1 Scanner tarayici = new Scanner(System.in);
2 System.out.print("Birinci sayıyı giriniz: ");
3 double sayi1 = tarayici.nextDouble();
4 System.out.print("İkinci sayıyı giriniz: ");
5 double sayi2 = tarayici.nextDouble();
6 double toplam = sayi1 + sayi2;
7 System.out.println(sayi1 + " ile " + sayi2
8         + " toplamı: " + toplam);
▶ tarayici.close();
```

double sayi1	double sayi2	double toplam
3.0	5.0	8.0

Tarayıcı kapatılır, program biter. Çıktı test senaryosundaki beklenen 8 ile uyuşuyor.

Adım 8 / 8

DEĞİŞKENLER

`tarayici = kapalı`

`sayi1 = 3.0` `sayi2 = 5.0`

`toplam = 8.0`

ÇIKTI

```
Birinci sayıyı giriniz: 3
İkinci sayıyı giriniz: 5
3.0 ile 5.0 toplamı: 8.0
```

Örnek Algoritmalar

Her algoritma için: sorun tanımı → sözde kod → değişken tablosuyla elle izleme → Java kodu ve test.

İki tam sayıyı sıralı yazdırma
Karar yapısı: `Eğer … Değilse`

Üç tam sayıyı sıralı yazdırma
İç içe kararlar, `ve` bağlacı

Faktöriyel bulma
Döngü ve biriktirici değişken

Fibonacci serisi
Döngüde değişkenleri kaydırma

Palindrom kontrolü
Metni temizleme, ters çevirme, karşılaştırma

Asal sayı kontrolü
Bayrak (flag) ve döngüyü kırma

Üs alma
Tekrarlı çarpma

Arama ve sıralama
Bölümün ikinci yarısında: rastgele, doğrusal, ikili arama; karıştırma, seçimli sıralama, kabarcık sıralaması.

İki Tam Sayıyı Küçükten Büyüğe Yazdırma

```
Başla
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı
    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı
    Eğer birinciTamsayı < ikinciTamsayı ise
        Yaz birinciTamsayı, " ", ikinciTamsayı
    Değilse
        Yaz ikinciTamsayı, " ", birinciTamsayı
    Eğer Sonu
Bitir
```

Sorun tanımı

Girdi: iki tam sayı. Çıktı: aynı sayılar, küçük olan önce.

Girdi	Beklenen
4, 11	4 11
13, 11	11 13
7, 7	7 7

Sayılar eşitse koşul yanlış olur ve `Değilse` kolu çalışır; iki sayı aynı olduğu için çıktı yine doğrudur.

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
4      Yaz "İkinci tamsayıyı girin: "
5      Oku ikinciTamsayı
6      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
11 Bitir
```

birinciTamsayı

ikinciTamsayı

Kullanıcıdan birinci sayı istenir.

Adım 1 / 13

DEĞİŞKENLER

`birinciTamsayı = –`

`ikinciTamsayı = –`

ÇIKTI

```
Birinci tamsayıyı girin:
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
4      Yaz "İkinci tamsayıyı girin: "
5      Oku ikinciTamsayı
6      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
11 Bitir
```



Kullanıcı **4** girer; değer `birinciTamsayı` kutusuna yazılır.

Adım 2 / 13

DEĞİŞKENLER

`birinciTamsayı = 4`

`ikinciTamsayı = –`

ÇIKTI

```
Birinci tamsayıyı girin: 4
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
▶      Yaz "İkinci tamsayıyı girin: "
▶      Oku ikinciTamsayı
6      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
11 Bitir
```

birinciTamsayı	ikinciTamsayı
4	11

İkinci sayı istenir; kullanıcı **11** girer.

Adım 3 / 13

DEĞİŞKENLER

`birinciTamsayı = 4`

`ikinciTamsayı = 11`

ÇIKTI

```
Birinci tamsayıyı girin: 4
İkinci tamsayıyı girin: 11
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
4      Yaz "İkinci tamsayıyı girin: "
5      Oku ikinciTamsayı
▶      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
11 Bitir
```

birinciTamsayı		ikinciTamsayı
4	<	11

doğru

Koşul: 4 < 11 → **doğru**.

Adım 4 / 13

DEĞİŞKENLER

- `birinciTamsayı = 4`
- `ikinciTamsayı = 11`

ÇIKTI

```
Birinci tamsayıyı girin: 4
İkinci tamsayıyı girin: 11
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
4      Yaz "İkinci tamsayıyı girin: "
5      Oku ikinciTamsayı
6      Eğer birinciTamsayı < ikinciTamsayı ise
▶          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
11 Bitir
```

birinciTamsayı		ikinciTamsayı
4	<	11

doğru

Koşul doğru olduğu için `ise` kolu çalışır: önce birinci, sonra ikinci yazılır.

Adım 5 / 13

DEĞİŞKENLER

- `birinciTamsayı = 4`
- `ikinciTamsayı = 11`

ÇIKTI

```
Birinci tamsayıyı girin: 4
İkinci tamsayıyı girin: 11
4 11
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
4      Yaz "İkinci tamsayıyı girin: "
5      Oku ikinciTamsayı
6      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
▶  Bitir
```

birinciTamsayı: 4

ikinciTamsayı: 11

Algoritma biter. Çıktı **4 11**, beklenenle aynı.

Adım 6 / 13

DEĞİŞKENLER

- `birinciTamsayı = 4`
- `ikinciTamsayı = 11`

ÇIKTI

```
Birinci tamsayıyı girin: 4
İkinci tamsayıyı girin: 11
4 11
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
▶      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
4      Yaz "İkinci tamsayıyı girin: "
5      Oku ikinciTamsayı
6      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
11 Bitir
```

birinciTamsayı

ikinciTamsayı

İkinci test: algoritma 13 ve 11 girdileriyle yeniden çalıştırılıyor.

Adım 7 / 13

DEĞİŞKENLER

`birinciTamsayı = –`

`ikinciTamsayı = –`

ÇIKTI

```
Birinci tamsayıyı girin:
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
4      Yaz "İkinci tamsayıyı girin: "
5      Oku ikinciTamsayı
6      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
11 Bitir
```

birinciTamsayı: 13

ikinciTamsayı:

Kullanıcı **13** girer; değer `birinciTamsayı` kutusuna yazılır.

Adım 8 / 13

DEĞİŞKENLER

`birinciTamsayı = 13`

`ikinciTamsayı = –`

ÇIKTI

```
Birinci tamsayıyı girin: 13
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
▶      Yaz "İkinci tamsayıyı girin: "
▶      Oku ikinciTamsayı
6      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
11 Bitir
```

birinciTamsayı	ikinciTamsayı
13	11

İkinci sayı istenir; kullanıcı **11** girer.

Adım 9 / 13

DEĞİŞKENLER

`birinciTamsayı = 13`

`ikinciTamsayı = 11`

ÇIKTI

```
Birinci tamsayıyı girin: 13
İkinci tamsayıyı girin: 11
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
4      Yaz "İkinci tamsayıyı girin: "
5      Oku ikinciTamsayı
▶      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
11 Bitir
```

birinciTamsayı		ikinciTamsayı
13	< /	11

yanlış

Koşul: 13 < 11 → **yanlış**.

Adım 10 / 13

DEĞİŞKENLER

- `birinciTamsayı = 13`
- `ikinciTamsayı = 11`

ÇIKTI

```
Birinci tamsayıyı girin: 13
İkinci tamsayıyı girin: 11
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
4      Yaz "İkinci tamsayıyı girin: "
5      Oku ikinciTamsayı
6      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
▶      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
11 Bitir
```

birinciTamsayı		ikinciTamsayı
13	<	11

yanlış

Koşul yanlış: `Değilse` koluna geçilir.

Adım 11 / 13

DEĞİŞKENLER

- `birinciTamsayı = 13`
- `ikinciTamsayı = 11`

ÇIKTI

```
Birinci tamsayıyı girin: 13
İkinci tamsayıyı girin: 11
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
4      Yaz "İkinci tamsayıyı girin: "
5      Oku ikinciTamsayı
6      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
▶          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
11 Bitir
```

birinciTamsayı		ikinciTamsayı
13	<	11

yanlış

Önce ikinci (küçük olan), sonra birinci yazılır.

Adım 12 / 13

DEĞİŞKENLER

- `birinciTamsayı = 13`
- `ikinciTamsayı = 11`

ÇIKTI

```
Birinci tamsayıyı girin: 13
İkinci tamsayıyı girin: 11
11 13
```

İki Tam Sayı: Elle İzleme (4, 11) ve (13, 11)

```
1  Başla
2      Yaz "Birinci tamsayıyı girin: "
3      Oku birinciTamsayı
4      Yaz "İkinci tamsayıyı girin: "
5      Oku ikinciTamsayı
6      Eğer birinciTamsayı < ikinciTamsayı ise
7          Yaz birinciTamsayı, " ", ikinciTamsayı
8      Değilse
9          Yaz ikinciTamsayı, " ", birinciTamsayı
10     Eğer Sonu
▶   Bitir
```

birinciTamsayı	ikinciTamsayı
13	11

Algoritma biter. Çıktı **11 13**, beklenenle aynı.

Adım 13 / 13

DEĞİŞKENLER

`birinciTamsayı = 13`

`ikinciTamsayı = 11`

ÇIKTI

```
Birinci tamsayıyı girin: 13
İkinci tamsayıyı girin: 11
11 13
```

ÖRNEK ALGORİTMALAR

Üç Tam Sayıyı Küçükten Büyüğe Yazdırma

```
Başla
    // İlk tamsayıyı kullanıcıdan al
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı
    // İkinci tamsayıyı kullanıcıdan al
    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı
    // Üçüncü tamsayıyı kullanıcıdan al
    Yaz "Üçüncü tamsayıyı girin: "
    Oku üçüncüTamsayı
    // Küçükten büyüğe sırala ve yazdır
    Eğer birinciTamsayı ≤ ikinciTamsayı ve
         birinciTamsayı ≤ üçüncüTamsayı ise
        Yaz birinciTamsayı
        Eğer ikinciTamsayı ≤ üçüncüTamsayı ise
            Yaz ikinciTamsayı, " ", üçüncüTamsayı
        Değilse
            Yaz üçüncüTamsayı, " ", ikinciTamsayı
        Eğer Sonu
    Değilse Eğer ikinciTamsayı ≤ birinciTamsayı ve
         ikinciTamsayı ≤ üçüncüTamsayı ise
        Yaz ikinciTamsayı
        Eğer birinciTamsayı ≤ üçüncüTamsayı ise
            Yaz birinciTamsayı, " ", üçüncüTamsayı
        Değilse
            Yaz üçüncüTamsayı, " ", birinciTamsayı
        Eğer Sonu
    Değilse
        Yaz üçüncüTamsayı
        Eğer birinciTamsayı ≤ ikinciTamsayı ise
            Yaz birinciTamsayı, " ", ikinciTamsayı
        Değilse
            Yaz ikinciTamsayı, " ", birinciTamsayı
        Eğer Sonu
    Eğer Sonu
Bitir
```

Fikir: önce **en küçüğü** bul ve yaz; kalan iki sayıyı iki tam sayı algoritmasıyla sırala. Aynı küçük algoritma üç kez kullanılıyor: bir fonksiyon olmaya çok uygun.

Üç Tam Sayı: Java Kodu ve Testler

```java
import java.util.Scanner;

public class UcSayiSirala {
    public static void main(String[] args) {
        Scanner tarayici = new Scanner(System.in);
        System.out.print("Üç tamsayı girin: ");
        int birinci = tarayici.nextInt();
        int ikinci = tarayici.nextInt();
        int ucuncu = tarayici.nextInt();
        if (birinci <= ikinci && birinci <= ucuncu) {
            System.out.print(birinci + " ");
            if (ikinci <= ucuncu) System.out.println(ikinci + " " + ucuncu);
            else                  System.out.println(ucuncu + " " + ikinci);
        } else if (ikinci <= birinci && ikinci <= ucuncu) {
            System.out.print(ikinci + " ");
            if (birinci <= ucuncu) System.out.println(birinci + " " + ucuncu);
            else                   System.out.println(ucuncu + " " + birinci);
        } else {
            System.out.print(ucuncu + " ");
            if (birinci <= ikinci) System.out.println(birinci + " " + ikinci);
            else                   System.out.println(ikinci + " " + birinci);
        }
        tarayici.close();
    }
}
```

Testler: 3! = 6 farklı sıranın hepsi (1 2 3, 1 3 2, 2 1 3, 2 3 1, 3 1 2, 3 2 1) → `1 2 3`; eşit sayılar 5 5 2 → `2 5 5`.

Üç Tam Sayı: Java Kodunun İzlenmesi (7, 3, 5)

```
▶ if (birinci <= ikinci && birinci <= ucuncu) {
2     System.out.print(birinci + " ");
3     if (ikinci <= ucuncu)
4         System.out.println(ikinci + " " + ucuncu);
5     else
6         System.out.println(ucuncu + " " + ikinci);
7 } else if (ikinci <= birinci && ikinci <= ucuncu) {
8     System.out.print(ikinci + " ");
9     if (birinci <= ucuncu)
10        System.out.println(birinci + " " + ucuncu);
11    else
12        System.out.println(ucuncu + " " + birinci);
13 } else {
14    System.out.print(ucuncu + " ");
15    if (birinci <= ikinci)
16        System.out.println(birinci + " " + ikinci);
17    else
18        System.out.println(ikinci + " " + birinci);
19 }
```

birinci	ikinci	ucuncu
7	3	5

7 ≤ 3 ? yanlış

Girdi 7, 3, 5. İlk koşul: 7 ≤ 3 **yanlış**; `&&` 'nin sağ tarafına bakılmaz bile.

Adım 1 / 5

DEĞİŞKENLER

`birinci = 7` `ikinci = 3` `ucuncu = 5`

ÇIKTI

Üç Tam Sayı: Java Kodunun İzlenmesi (7, 3, 5)

```
1  if (birinci <= ikinci && birinci <= ucuncu) {
2      System.out.print(birinci + " ");
3      if (ikinci <= ucuncu)
4          System.out.println(ikinci + " " + ucuncu);
5      else
6          System.out.println(ucuncu + " " + ikinci);
▶ } else if (ikinci <= birinci && ikinci <= ucuncu) {
8      System.out.print(ikinci + " ");
9      if (birinci <= ucuncu)
10         System.out.println(birinci + " " + ucuncu);
11     else
12         System.out.println(ucuncu + " " + birinci);
13 } else {
14     System.out.print(ucuncu + " ");
15     if (birinci <= ikinci)
16         System.out.println(birinci + " " + ikinci);
17     else
18         System.out.println(ikinci + " " + birinci);
19 }
```

birinci	ikinci	ucuncu
7	3	5

3 ≤ 7 ve 3 ≤ 5 : doğru

İkinci koşul: 3 ≤ 7 **doğru** ve 3 ≤ 5 **doğru** → en küçük `ikinci`.

Adım 2 / 5

DEĞİŞKENLER

`birinci = 7` `ikinci = 3` `ucuncu = 5`

ÇIKTI

Üç Tam Sayı: Java Kodunun İzlenmesi (7, 3, 5)

```
1  if (birinci <= ikinci && birinci <= ucuncu) {
2      System.out.print(birinci + " ");
3      if (ikinci <= ucuncu)
4          System.out.println(ikinci + " " + ucuncu);
5      else
6          System.out.println(ucuncu + " " + ikinci);
7  } else if (ikinci <= birinci && ikinci <= ucuncu) {
▶      System.out.print(ikinci + " ");
9      if (birinci <= ucuncu)
10         System.out.println(birinci + " " + ucuncu);
11     else
12         System.out.println(ucuncu + " " + birinci);
13 } else {
14     System.out.print(ucuncu + " ");
15     if (birinci <= ikinci)
16         System.out.println(birinci + " " + ikinci);
17     else
18         System.out.println(ikinci + " " + birinci);
19 }
```

birinci	ikinci	ucuncu
7	3	5

en küçük: 3

En küçük sayı yazılır; `print` satır sonu eklemez.

Adım 3 / 5

DEĞİŞKENLER

`birinci = 7` `ikinci = 3` `ucuncu = 5`

ÇIKTI

```
3
```

Üç Tam Sayı: Java Kodunun İzlenmesi (7, 3, 5)

```
1  if (birinci <= ikinci && birinci <= ucuncu) {
2      System.out.print(birinci + " ");
3      if (ikinci <= ucuncu)
4          System.out.println(ikinci + " " + ucuncu);
5      else
6          System.out.println(ucuncu + " " + ikinci);
7  } else if (ikinci <= birinci && ikinci <= ucuncu) {
8      System.out.print(ikinci + " ");
▶      if (birinci <= ucuncu)
10         System.out.println(birinci + " " + ucuncu);
11     else
12         System.out.println(ucuncu + " " + birinci);
13 } else {
14     System.out.print(ucuncu + " ");
15     if (birinci <= ikinci)
16         System.out.println(birinci + " " + ikinci);
17     else
18         System.out.println(ikinci + " " + birinci);
19 }
```

birinci	ikinci	ucuncu
7	3	5

7 ≤ 5 ? yanlış

Kalan iki sayı karşılaştırılır: 7 ≤ 5 **yanlış**.

Adım 4 / 5

DEĞİŞKENLER

`birinci = 7` `ikinci = 3` `ucuncu = 5`

ÇIKTI

```
3
```

Üç Tam Sayı: Java Kodunun İzlenmesi (7, 3, 5)

```
1  if (birinci <= ikinci && birinci <= ucuncu) {
2      System.out.print(birinci + " ");
3      if (ikinci <= ucuncu)
4          System.out.println(ikinci + " " + ucuncu);
5      else
6          System.out.println(ucuncu + " " + ikinci);
7  } else if (ikinci <= birinci && ikinci <= ucuncu) {
8      System.out.print(ikinci + " ");
9      if (birinci <= ucuncu)
10         System.out.println(birinci + " " + ucuncu);
▶      else
▶          System.out.println(ucuncu + " " + birinci);
13 } else {
14     System.out.print(ucuncu + " ");
15     if (birinci <= ikinci)
16         System.out.println(birinci + " " + ikinci);
17     else
18         System.out.println(ikinci + " " + birinci);
19 }
```

birinci	ikinci	ucuncu
7	3	5

sıra: 3 5 7

`else` kolu: önce 5, sonra 7 yazılır. Çıktı **3 5 7**.

Adım 5 / 5

DEĞİŞKENLER

`birinci = 7` `ikinci = 3` `ucuncu = 5`

ÇIKTI

```
3 5 7
```

Faktöriyel Bulma

```
Başla
    // Bir sayıyı gir
    Oku sayı
    // Sonucu saklamak için bir değişkeni başlat
    faktoriyel = 1
    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngü Sonu
    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

Tanım: n! = 1 × 2 × 3 × … × n, ayrıca 0! = 1.

n	n!
0	1
1	1
3	6
5	120
10	3 628 800

`faktoriyel` bir **biriktiricidir** (accumulator): çarpmanın etkisiz elemanı olan **1** ile başlatılır. 0 ile başlatılsaydı sonuç hep 0 olurdu.

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
▶     Oku sayı
3     faktoriyel = 1
4     Döngü i = 1'den başlayarak sayı'ya kadar
5         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1

Kullanıcı **5** girer.

Adım 1 / 14

DEĞİŞKENLER

`sayı = 5` `faktoriyel = –` `i = –`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
4     Döngü i = 1'den başlayarak sayı'ya kadar
5         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1

Biriktirici 1 ile başlatılır.

Adım 2 / 14

DEĞİŞKENLER

`sayı = 5` `faktoriyel = 1` `i = –`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
▶     Döngü i = 1'den başlayarak sayı'ya kadar
5         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1

Döngü: i = 1 (i ≤ 5 olduğu için gövde çalışır).

Adım 3 / 14

DEĞİŞKENLER

`sayı = 5` `faktoriyel = 1` `i = 1`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
4     Döngü i = 1'den başlayarak sayı'ya kadar
▶         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1

faktoriyel = 1 × 1 = **1**.

Adım 4 / 14

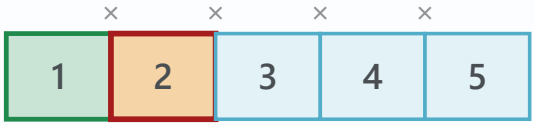
DEĞİŞKENLER

`sayı = 5` `faktoriyel = 1` `i = 1`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
▶     Döngü i = 1'den başlayarak sayı'ya kadar
5         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1

Döngü: i = 2 (i ≤ 5 olduğu için gövde çalışır).

Adım 5 / 14

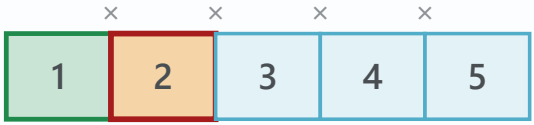
DEĞİŞKENLER

`sayı = 5` `faktoriyel = 1` `i = 2`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
4     Döngü i = 1'den başlayarak sayı'ya kadar
▶         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1 × 2 = 2

faktoriyel = 1 × 2 = **2**.

Adım 6 / 14

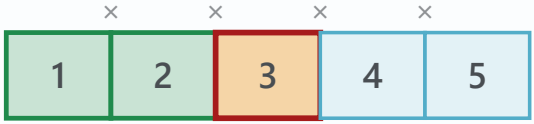
DEĞİŞKENLER

`sayı = 5` `faktoriyel = 2` `i = 2`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
▶     Döngü i = 1'den başlayarak sayı'ya kadar
5         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1 × 2 = 2

Döngü: i = 3 (i ≤ 5 olduğu için gövde çalışır).

Adım 7 / 14

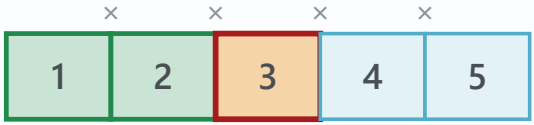
DEĞİŞKENLER

`sayı = 5` `faktoriyel = 2` `i = 3`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
4     Döngü i = 1'den başlayarak sayı'ya kadar
▶         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1 × 2 × 3 = 6

faktoriyel = 2 × 3 = **6**.

Adım 8 / 14

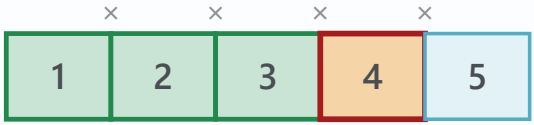
DEĞİŞKENLER

`sayı = 5` `faktoriyel = 6` `i = 3`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
▶     Döngü i = 1'den başlayarak sayı'ya kadar
5         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1 × 2 × 3 = 6

Döngü: i = 4 (i ≤ 5 olduğu için gövde çalışır).

Adım 9 / 14

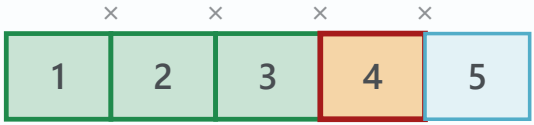
DEĞİŞKENLER

`sayı = 5` `faktoriyel = 6` `i = 4`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
4     Döngü i = 1'den başlayarak sayı'ya kadar
▶         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1 × 2 × 3 × 4 = 24

faktoriyel = 6 × 4 = **24**.

Adım 10 / 14

DEĞİŞKENLER

`sayı = 5` `faktoriyel = 24` `i = 4`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
▶     Döngü i = 1'den başlayarak sayı'ya kadar
5         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1 × 2 × 3 × 4 = 24

Döngü: i = 5 (i ≤ 5 olduğu için gövde çalışır).

Adım 11 / 14

DEĞİŞKENLER

`sayı = 5` `faktoriyel = 24` `i = 5`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
4     Döngü i = 1'den başlayarak sayı'ya kadar
▶         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1 × 2 × 3 × 4 × 5 = 120

faktoriyel = 24 × 5 = **120**.

Adım 12 / 14

DEĞİŞKENLER

`sayı = 5` `faktoriyel = 120` `i = 5`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
4     Döngü i = 1'den başlayarak sayı'ya kadar
5         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1 × 2 × 3 × 4 × 5 = 120

i = 6 olur; 6 ≤ 5 yanlış olduğu için döngü biter.

Adım 13 / 14

DEĞİŞKENLER

`sayı = 5` `faktoriyel = 120` `i = 6`

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2     Oku sayı
3     faktoriyel = 1
4     Döngü i = 1'den başlayarak sayı'ya kadar
5         faktoriyel = faktoriyel * i
6     Döngü Sonu
7     Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



faktoriyel = 1 × 2 × 3 × 4 × 5 = 120

Sonuç yazdırılır: 5! = 120.

Adım 14 / 14

DEĞİŞKENLER

`sayı = 5` `faktoriyel = 120` `i = 6`

ÇIKTI

```
Girilen sayının faktöriyeli: 120
```

ÖRNEK ALGORİTMALAR

Fibonacci Serisi Bulma

```
Başla
    // Kaç terim istendiğini gir
    Oku n
    // İlk iki Fibonacci sayısını başlat
    a = 0
    b = 1
    Eğer n ≥ 1 ise Yaz a
    Eğer n ≥ 2 ise Yaz b
    // Kalan terimleri hesapla
    Döngü i = 2'den başlayarak n – 1'e kadar
        sonuc = a + b
        a = b
        b = sonuc
        Yaz sonuc
    Döngü Sonu
Bitir
```

Tanım: İlk iki terim 0 ve 1; sonraki her terim önceki ikisinin toplamıdır.

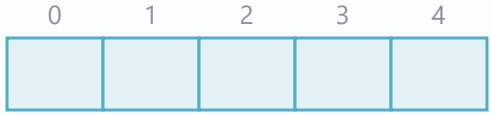
`0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …`

n (terim sayısı)	Beklenen çıktı
1	0
2	0 1
5	0 1 1 2 3
7	0 1 1 2 3 5 8

Döngü i = 2 … n − 1 arasında **n − 2** kez döner; ilk iki terim döngüden önce yazılır. `a = b` ve `b = sonuc` satırları değerleri bir adım "kaydırır".

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
▶      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
9          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



Kullanıcı **n = 5** girer: ilk 5 terim istenir.

Adım 1 / 20

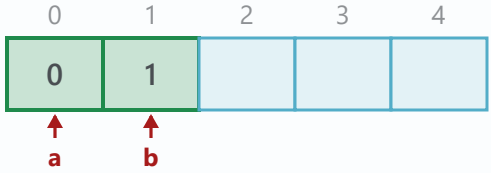
DEĞİŞKENLER

`n = 5` `a = –` `b = –` `i = –` `sonuc = –`

ÇIKTI

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
▶      a = 0
▶      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
9          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



İlk iki terim: a = 0, b = 1.

Adım 2 / 20

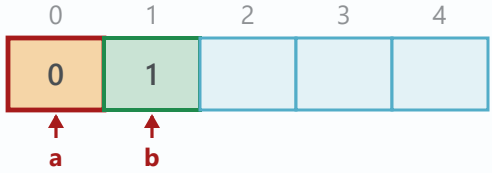
DEĞİŞKENLER

n = 5 | a = 0 | b = 1 | i = – | sonuc = –

ÇIKTI

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
▶      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
9          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



5 ≥ 1: `a` yazılır.

Adım 3 / 20

DEĞİŞKENLER

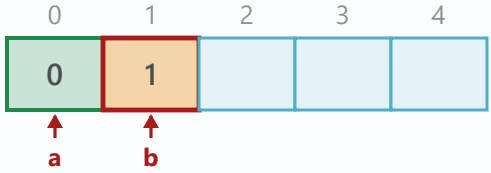
`n = 5` `a = 0` `b = 1` `i = –` `sonuc = –`

ÇIKTI

```
0
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
9          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



5 ≥ 2: `b` yazılır.

Adım 4 / 20

DEĞİŞKENLER

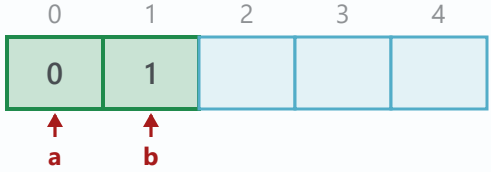
`n = 5` `a = 0` `b = 1` `i = –` `sonuc = –`

ÇIKTI

```
0 1
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
▶      Döngü i = 2'den başlayarak n - 1'e kadar
8          sonuc = a + b
9          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



Döngü: i = 2 (i ≤ n − 1 = 4).

Adım 5 / 20

DEĞİŞKENLER

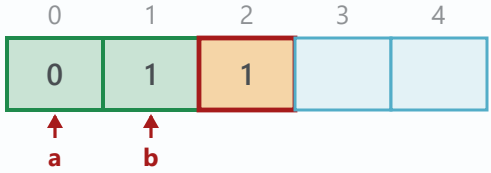
n = **5** | a = **0** | b = **1** | i = **2** | sonuc = **–**

ÇIKTI

```
0 1
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
▶          sonuc = a + b
9          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



sonuc = a + b = 0 + 1 = **1**.

Adım 6 / 20

DEĞİŞKENLER

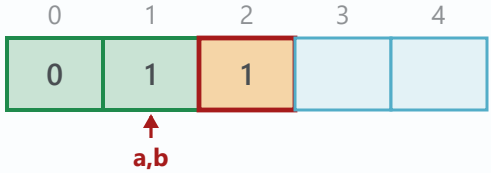
n = **5** | a = **0** | b = **1** | i = **2** | sonuc = **1**

ÇIKTI

```
0 1
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
▶          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



a = b → a = 1 (bir adım ileri kaydı).

Adım 7 / 20

DEĞİŞKENLER

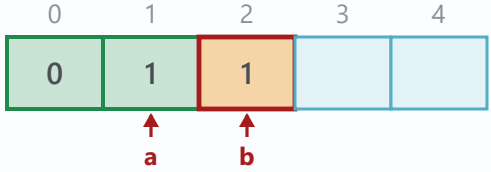
n = 5 | a = 1 | b = 1 | i = 2 | sonuc = 1

ÇIKTI

```
0 1
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
9          a = b
▶          b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



b = sonuc → b = 1.

Adım 8 / 20

DEĞİŞKENLER

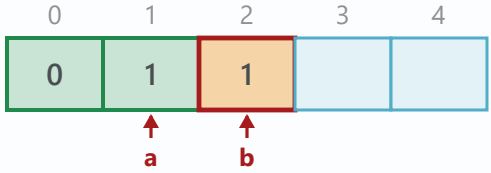
n = **5** | a = **1** | b = **1** | i = **2** | sonuc = **1**

ÇIKTI

```
0 1
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
9          a = b
10         b = sonuc
▶          Yaz sonuc
12     Döngü Sonu
13 Bitir
```



Yeni terim yazılır: 1.

Adım 9 / 20

DEĞİŞKENLER

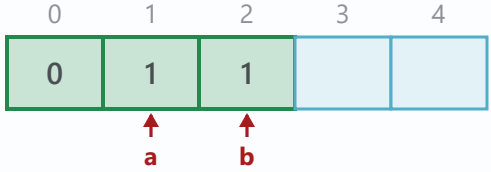
`n = 5` `a = 1` `b = 1` `i = 2` `sonuc = 1`

ÇIKTI

```
0 1 1
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
▶      Döngü i = 2'den başlayarak n - 1'e kadar
8          sonuc = a + b
9          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



Döngü: i = 3 (i ≤ n − 1 = 4).

Adım 10 / 20

DEĞİŞKENLER

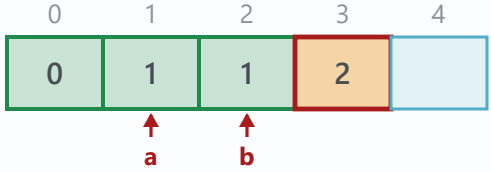
n = **5** | a = **1** | b = **1** | i = **3** | sonuc = **1**

ÇIKTI

```
0 1 1
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
▶          sonuc = a + b
9          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



sonuc = a + b = 1 + 1 = **2**.

Adım 11 / 20

DEĞİŞKENLER

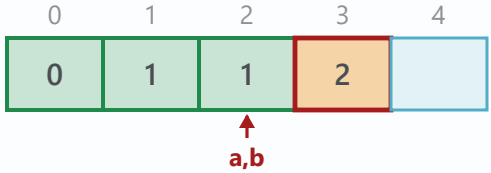
`n = 5` `a = 1` `b = 1` `i = 3` `sonuc = 2`

ÇIKTI

```
0 1 1
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
▶          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



a = b → a = 1 (bir adım ileri kaydı).

Adım 12 / 20

DEĞİŞKENLER

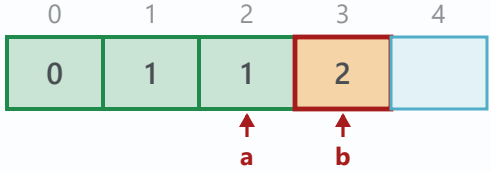
n = **5** | a = **1** | b = **1** | i = **3** | sonuc = **2**

ÇIKTI

```
0 1 1
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
9          a = b
▶          b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



b = sonuc → b = 2.

Adım 13 / 20

DEĞİŞKENLER

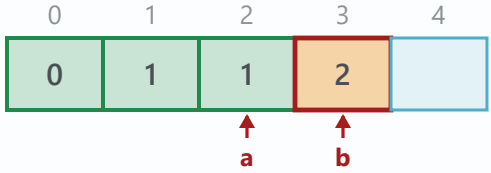
n = 5 | a = 1 | b = 2 | i = 3 | sonuc = 2

ÇIKTI

```
0 1 1
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
9          a = b
10         b = sonuc
▶          Yaz sonuc
12     Döngü Sonu
13 Bitir
```



Yeni terim yazılır: 2.

Adım 14 / 20

DEĞİŞKENLER

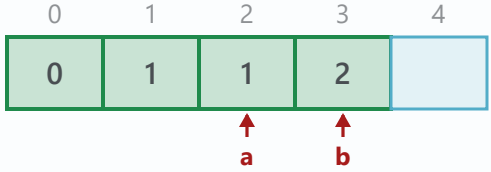
`n = 5` `a = 1` `b = 2` `i = 3` `sonuc = 2`

ÇIKTI

```
0 1 1 2
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
▶      Döngü i = 2'den başlayarak n - 1'e kadar
8          sonuc = a + b
9          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



Döngü: i = 4 (i ≤ n − 1 = 4).

Adım 15 / 20

DEĞİŞKENLER

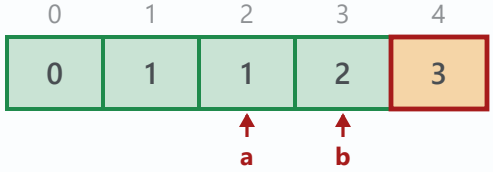
n = 5 | a = 1 | b = 2 | i = 4 | sonuc = 2

ÇIKTI

```
0 1 1 2
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
▶          sonuc = a + b
9          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



sonuc = a + b = 1 + 2 = **3**.

Adım 16 / 20

DEĞİŞKENLER

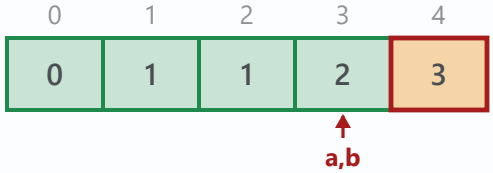
`n = 5` `a = 1` `b = 2` `i = 4` `sonuc = 3`

ÇIKTI

```
0 1 1 2
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
▶          a = b
10         b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



a = b → a = 2 (bir adım ileri kaydı).

Adım 17 / 20

DEĞİŞKENLER

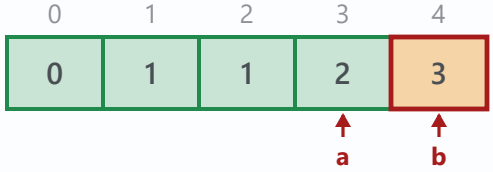
n = 5 | a = 2 | b = 2 | i = 4 | sonuc = 3

ÇIKTI

```
0 1 1 2
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
9          a = b
▶          b = sonuc
11         Yaz sonuc
12     Döngü Sonu
13 Bitir
```



b = sonuc → b = 3.

Adım 18 / 20

DEĞİŞKENLER

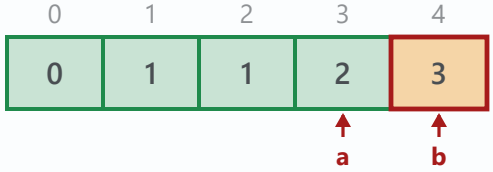
n = 5 | a = 2 | b = 3 | i = 4 | sonuc = 3

ÇIKTI

```
0 1 1 2
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
9          a = b
10         b = sonuc
▶          Yaz sonuc
12     Döngü Sonu
13 Bitir
```



Yeni terim yazılır: 3.

Adım 19 / 20

DEĞİŞKENLER

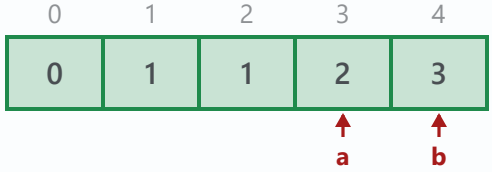
`n = 5` `a = 2` `b = 3` `i = 4` `sonuc = 3`

ÇIKTI

```
0 1 1 2 3
```

Fibonacci: Elle İzleme (n = 5)

```
1  Başla
2      Oku n
3      a = 0
4      b = 1
5      Eğer n ≥ 1 ise Yaz a
6      Eğer n ≥ 2 ise Yaz b
7      Döngü i = 2'den başlayarak n – 1'e kadar
8          sonuc = a + b
9          a = b
10         b = sonuc
11         Yaz sonuc
▶      Döngü Sonu
13 Bitir
```



i = 5 olur; 5 ≤ 4 yanlış, döngü biter. Tam 5 terim yazıldı: **0 1 1 2 3**.

Adım 20 / 20

DEĞİŞKENLER

`n = 5` `a = 2` `b = 3` `i = 5` `sonuc = 3`

ÇIKTI

```
0 1 1 2 3
```

ÖRNEK ALGORİTMALAR

Palindrom Kontrolü

```
Başla
    // Bir kelime veya cümlenin girilmesini iste
    Oku girdiMetin
    // Girdiyi temizle: küçük harf, boşluk ve noktalama yok
    temizMetin = Temizle(girdiMetin)
    // Temizlenmiş girdiyi ters çevir
    tersGirdi = TersÇevir(temizMetin)
    // Temizlenmiş girdi ile ters girdiyi karşılaştır
    Eğer temizMetin = tersGirdi ise
        Yaz "Girilen metin bir palindromdur."
    Değilse
        Yaz "Girilen metin bir palindrom değildir."
    Eğer Sonu
Bitir
```

Palindrom: Tersten okunuşu düz okunuşuyla aynı olan kelime veya cümle: *kayak, abcba, "Ey Edip Adana'da pide ye"*. Cümlelerde büyük/küçük harf, boşluk ve noktalama dikkate alınmaz; bu yüzden karşılaştırmadan önce metin **temizlenir**.

Girdi	Beklenen
abcba	palindrom
Kayak	palindrom
Kitap	değil

Palindrom: Elle İzleme (abcba ve Kitap)

```
1  Başla
2      Oku girdiMetin
3      temizMetin = Temizle(girdiMetin)
4      tersGirdi = TersÇevir(temizMetin)
5      Eğer temizMetin = tersGirdi ise
6          Yaz "Girilen metin bir palindromdur."
7      Değilse
8          Yaz "Girilen metin bir palindrom değildir."
9      Eğer Sonu
10 Bitir
```

girdiMetin

a	b	c	b	a

Kullanıcı **abcba** girer.

Adım 1 / 10

DEĞİŞKENLER

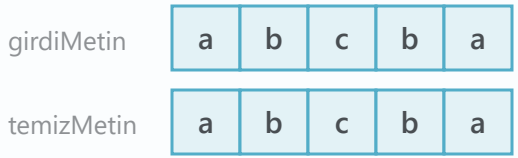
`girdiMetin = "abcba"`

`temizMetin = –` `tersGirdi = –`

ÇIKTI

Palindrom: Elle İzleme (abcba ve Kitap)

```
1  Başla
2      Oku girdiMetin
▶      temizMetin = Temizle(girdiMetin)
4      tersGirdi = TersÇevir(temizMetin)
5      Eğer temizMetin = tersGirdi ise
6          Yaz "Girilen metin bir palindromdur."
7      Değilse
8          Yaz "Girilen metin bir palindrom değildir."
9      Eğer Sonu
10 Bitir
```



Temizleme: küçük harfe çevrilir (değişiklik yok).

Adım 2 / 10

DEĞİŞKENLER

`girdiMetin = "abcba"`

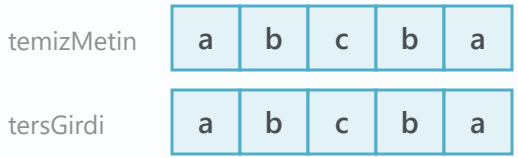
`temizMetin = "abcba"`

`tersGirdi = –`

ÇIKTI

Palindrom: Elle İzleme (abcba ve Kitap)

```
1  Başla
2      Oku girdiMetin
3      temizMetin = Temizle(girdiMetin)
▶      tersGirdi = TersÇevir(temizMetin)
5      Eğer temizMetin = tersGirdi ise
6          Yaz "Girilen metin bir palindromdur."
7      Değilse
8          Yaz "Girilen metin bir palindrom değildir."
9      Eğer Sonu
10 Bitir
```



Temiz metin sondan başa okunur: **abcba**.

Adım 3 / 10

DEĞİŞKENLER

`girdiMetin = "abcba"`

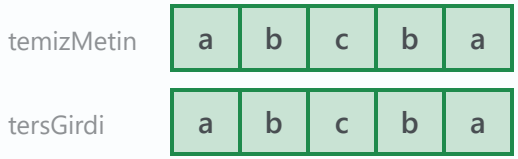
`temizMetin = "abcba"`

`tersGirdi = "abcba"`

ÇIKTI

Palindrom: Elle İzleme (abcba ve Kitap)

```
1  Başla
2      Oku girdiMetin
3      temizMetin = Temizle(girdiMetin)
4      tersGirdi = TersÇevir(temizMetin)
▶      Eğer temizMetin = tersGirdi ise
6          Yaz "Girilen metin bir palindromdur."
7      Değilse
8          Yaz "Girilen metin bir palindrom değildir."
9      Eğer Sonu
10 Bitir
```



Karşılaştırma harf harf: bütün harfler eşleşti → **doğru**.

Adım 4 / 10

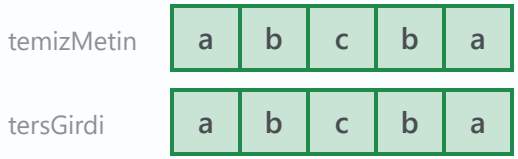
DEĞİŞKENLER

- `girdiMetin = "abcba"`
- `temizMetin = "abcba"`
- `tersGirdi = "abcba"`

ÇIKTI

Palindrom: Elle İzleme (abcba ve Kitap)

```
1  Başla
2      Oku girdiMetin
3      temizMetin = Temizle(girdiMetin)
4      tersGirdi = TersÇevir(temizMetin)
5      Eğer temizMetin = tersGirdi ise
6          Yaz "Girilen metin bir palindromdur."
7      Değilse
8          Yaz "Girilen metin bir palindrom değildir."
9      Eğer Sonu
10 Bitir
```



Koşul doğru: palindrom mesajı yazılır.

Adım 5 / 10

DEĞİŞKENLER

- `girdiMetin = "abcba"`
- `temizMetin = "abcba"`
- `tersGirdi = "abcba"`

ÇIKTI

```
Girilen metin bir palindromdur.
```

Palindrom: Elle İzleme (abcba ve Kitap)

```
1  Başla
▶      Oku girdiMetin
3      temizMetin = Temizle(girdiMetin)
4      tersGirdi = TersÇevir(temizMetin)
5      Eğer temizMetin = tersGirdi ise
6          Yaz "Girilen metin bir palindromdur."
7      Değilse
8          Yaz "Girilen metin bir palindrom değildir."
9      Eğer Sonu
10 Bitir
```

girdiMetin

K	i	t	a	p

İkinci test: Kullanıcı **Kitap** girer.

Adım 6 / 10

DEĞİŞKENLER

`girdiMetin = "Kitap"`

`temizMetin = –` `tersGirdi = –`

ÇIKTI

Palindrom: Elle İzleme (abcba ve Kitap)

```
1  Başla
2      Oku girdiMetin
▶      temizMetin = Temizle(girdiMetin)
4      tersGirdi = TersÇevir(temizMetin)
5      Eğer temizMetin = tersGirdi ise
6          Yaz "Girilen metin bir palindromdur."
7      Değilse
8          Yaz "Girilen metin bir palindrom değildir."
9      Eğer Sonu
10 Bitir
```



Temizleme: küçük harfe çevrilir ("Kitap" → "kitap").

Adım 7 / 10

DEĞİŞKENLER

`girdiMetin = "Kitap"`

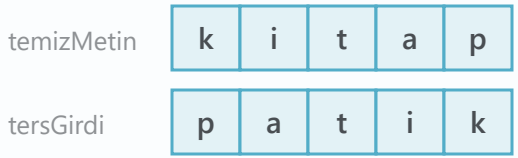
`temizMetin = "kitap"`

`tersGirdi = –`

ÇIKTI

Palindrom: Elle İzleme (abcba ve Kitap)

```
1  Başla
2      Oku girdiMetin
3      temizMetin = Temizle(girdiMetin)
▶      tersGirdi = TersÇevir(temizMetin)
5      Eğer temizMetin = tersGirdi ise
6          Yaz "Girilen metin bir palindromdur."
7      Değilse
8          Yaz "Girilen metin bir palindrom değildir."
9      Eğer Sonu
10 Bitir
```



Temiz metin sondan başa okunur: **patik**.

Adım 8 / 10

DEĞİŞKENLER

`girdiMetin = "Kitap"`

`temizMetin = "kitap"`

`tersGirdi = "patik"`

ÇIKTI

Palindrom: Elle İzleme (abcba ve Kitap)

```
1  Başla
2      Oku girdiMetin
3      temizMetin = Temizle(girdiMetin)
4      tersGirdi = TersÇevir(temizMetin)
▶      Eğer temizMetin = tersGirdi ise
6          Yaz "Girilen metin bir palindromdur."
7      Değilse
8          Yaz "Girilen metin bir palindrom değildir."
9      Eğer Sonu
10 Bitir
```



Karşılaştırma harf harf: kitap ≠ patik → **yanlış**.

Adım 9 / 10

DEĞİŞKENLER

- `girdiMetin = "Kitap"`
- `temizMetin = "kitap"`
- `tersGirdi = "patik"`

ÇIKTI

Palindrom: Elle İzleme (abcba ve Kitap)

```
1  Başla
2      Oku girdiMetin
3      temizMetin = Temizle(girdiMetin)
4      tersGirdi = TersÇevir(temizMetin)
5      Eğer temizMetin = tersGirdi ise
6          Yaz "Girilen metin bir palindromdur."
       Değilse
           Yaz "Girilen metin bir palindrom değildir."
9      Eğer Sonu
10 Bitir
```



Koşul yanlış: `Değilse` kolu çalışır.

Adım 10 / 10

DEĞİŞKENLER

- `girdiMetin = "Kitap"`
- `temizMetin = "kitap"`
- `tersGirdi = "patik"`

ÇIKTI

```
Girilen metin bir palindrom
değildir.
```

ÖRNEK ALGORİTMALAR

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı gir
    Oku sayı
    // Bayrak (flag): 0 ve 1 asal değildir
    Eğer sayı < 2 ise asalMi = False
    Değilse asalMi = True
    // 2'den sayının yarısına kadar bölenleri dene
    Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa sayı asal değildir
            asalMi = False
            Döngüyü Kır
        Eğer Sonu
    Döngü Sonu
    // Bayrağa göre sonucu yazdır
    Eğer asalMi = True ise
        Yaz sayı, " bir asal sayıdır."
    Değilse
        Yaz sayı, " bir asal sayı değildir."
    Eğer Sonu
Bitir
```

Asal sayı: 1'den büyük olup yalnızca 1'e ve kendisine bölünebilen sayı: 2, 3, 5, 7, 11, ...

- **Bayrak** (flag): başta "asal" varsayılır; bir bölen bulunursa bayrak indirilir.
- `%` kalanı verir: 21 % 3 = 0 → tam bölünüyor.
- Bir bölen bulununca devam etmeye gerek yoktur: **Döngüyü Kır**.
- Sayının yarısından büyük bir sayı (kendisi hariç) onu bölemez; `sayı / 2` tam sayı bölmesidir (7 / 2 = 3).
- 1 ve 0 için döngü hiç çalışmaz; bu yüzden ayrıca kontrol edilir.

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
▶      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```

bölen: 2 | 3

henüz bölen denenmedi

Kullanıcı **7** girer.

Adım 1 / 17

DEĞİŞKENLER

`sayı = 7` `asalMi = –` `bölen = –`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
▶      Eğer sayı < 2 ise asalMi = False
▶      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```

bölen: 2 3

henüz bölen denenmedi

7 ≥ 2 olduğu için bayrak **True** ile başlar.

Adım 2 / 17

DEĞİŞKENLER

`sayı = 7` `asalMi = True` `bölen = –`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
▶      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```

bölen: 2 3

7 % 2 = 1

Döngü: bölen = 2 (≤ 7 / 2 = 3).

Adım 3 / 17

DEĞİŞKENLER

`sayı = 7` `asalMi = True` `bölen = 2`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
▶          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```

bölen: 2 | 3

7 % 2 = 1 → bölünmüyor

7 % 2 = 1 ≠ 0 → bölünmüyor, devam.

Adım 4 / 17

DEĞİŞKENLER

`sayı = 7` `asalMi = True` `bölen = 2`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
▶      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```

bölen: 2 3

7 % 3 = 1

Döngü: bölen = 3 (≤ 7 / 2 = 3).

Adım 5 / 17

DEĞİŞKENLER

`sayı = 7` `asalMi = True` `bölen = 3`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
▶          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```

bölen: 2 | 3

7 % 3 = 1 → bölünmüyor

7 % 3 = 1 ≠ 0 → bölünmüyor, devam.

Adım 6 / 17

DEĞİŞKENLER

`sayı = 7` `asalMi = True` `bölen = 3`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
▶      Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```

bölen: 2 | 3

7 % 3 = 1 → bölünmüyor

bölen = 4 olur; 4 ≤ 3 yanlış, döngü biter. Hiç bölen bulunmadı.

Adım 7 / 17

DEĞİŞKENLER

`sayı = 7` `asalMi = True` `bölen = 4`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
▶      Eğer asalMi = True ise
▶          Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```

bölen: 2 | 3

7 % 3 = 1 → bölünmüyor

Bayrak True: **7 asaldır**.

Adım 8 / 17

DEĞİŞKENLER

`sayı = 7` `asalMi = True` `bölen = 4`

ÇIKTI

```
7 bir asal sayıdır.
```

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
▶      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```

bölen:

2	3	4	5	6	7	8	9	10

henüz bölen denenmedi

İkinci test: Kullanıcı **21** girer.

Adım 9 / 17

DEĞİŞKENLER

`sayı = 21` `asalMi = –` `bölen = –`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
▶      Eğer sayı < 2 ise asalMi = False
▶      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```

bölen:

2	3	4	5	6	7	8	9	10

henüz bölen denenmedi

21 ≥ 2 olduğu için bayrak **True** ile başlar.

Adım 10 / 17

DEĞİŞKENLER

`sayı = 21` `asalMi = True` `bölen = –`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
▶      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```



21 % 2 = 1

Döngü: bölen = 2 (≤ 21 / 2 = 10).

Adım 11 / 17

DEĞİŞKENLER

`sayı = 21` `asalMi = True` `bölen = 2`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
▶          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```

bölen:

2	3	4	5	6	7	8	9	10

21 % 2 = 1 → bölünmüyor

21 % 2 = 1 ≠ 0 → bölünmüyor, devam.

Adım 12 / 17

DEĞİŞKENLER

`sayı = 21` `asalMi = True` `bölen = 2`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
▶      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```



21 % 3 = 0

Döngü: bölen = 3 (≤ 21 / 2 = 10).

Adım 13 / 17

DEĞİŞKENLER

`sayı = 21` `asalMi = True` `bölen = 3`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```



21 % 3 = 0 → tam bölünüyor

21 % 3 = 0 → tam bölünüyor!

Adım 14 / 17

DEĞİŞKENLER

`sayı = 21` `asalMi = True` `bölen = 3`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
▶              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```



21 % 3 = 0 → tam bölünüyor

Bayrak indirilir: asalMi = **False**.

Adım 15 / 17

DEĞİŞKENLER

`sayı = 21` `asalMi = False` `bölen = 3`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
▶              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
11     Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
13     Değilse
14         Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```



21 % 3 = 0 → tam bölünüyor

Döngüyü Kır: kalan bölenler denenmez.

Adım 16 / 17

DEĞİŞKENLER

`sayı = 21` `asalMi = False` `bölen = 3`

ÇIKTI

Asal Sayı: Elle İzleme (7 ve 21)

```
1  Başla
2      Oku sayı
3      Eğer sayı < 2 ise asalMi = False
4      Değilse asalMi = True
5      Döngü bölen = 2'den başlayarak sayı / 2'ye kadar
6          Eğer sayı % bölen = 0 ise
7              asalMi = False
8              Döngüyü Kır
9          Eğer Sonu
10     Döngü Sonu
▶      Eğer asalMi = True ise
12         Yaz sayı, " bir asal sayıdır."
▶      Değilse
▶          Yaz sayı, " bir asal sayı değildir."
15     Eğer Sonu
16 Bitir
```



21 % 3 = 0 → tam bölünüyor

Bayrak False: `Değilse` kolu çalışır.

Adım 17 / 17

DEĞİŞKENLER

`sayı = 21` `asalMi = False` `bölen = 3`

ÇIKTI

```
21 bir asal sayı değildir.
```

Üs Alma

```
Başla
    // Bir sayıyı gir
    Oku taban
    // Bir üssü gir
    Oku üs
    // Sonucu saklamak için bir değişkeni başlat
    sonuc = 1
    // Üssü kullanarak sonucu hesapla
    Döngü i = 1'den başlayarak üs'e kadar
        sonuc = sonuc * taban
    Döngü Sonu
    Yaz "Sonuç: ", sonuc
Bitir
```

Tanım: $taban^{üs}$ = taban × taban × … × taban (üs kez); $taban^{0} = 1$.

taban	üs	Beklenen
3	4	81
2	10	1024
5	0	1

Faktöriyelle aynı kalıp: biriktirici 1 ile başlar, döngü her turda bir çarpma yapar. Farkı, hep **aynı** sayıyla (taban) çarpmasıdır. Üs 0 ise döngü hiç çalışmaz ve sonuç 1 kalır.

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
4     sonuc = 1
5     Döngü i = 1'den başlayarak üs'e kadar
6         sonuc = sonuc * taban
7     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 1

Kullanıcı taban = **3** girer.

Adım 1 / 13

DEĞİŞKENLER

`taban = 3` `üs = –` `sonuc = –` `i = –`

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
▶     Oku üs
4     sonuc = 1
5     Döngü i = 1'den başlayarak üs'e kadar
6         sonuc = sonuc * taban
7     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 1

Kullanıcı üs = **4** girer: 3^4 hesaplanacak.

Adım 2 / 13

DEĞİŞKENLER

`taban = 3` `üs = 4` `sonuc = –` `i = –`

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
▶     sonuc = 1
5     Döngü i = 1'den başlayarak üs'e kadar
6         sonuc = sonuc * taban
7     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 1

Biriktirici 1 ile başlar.

Adım 3 / 13

DEĞİŞKENLER

taban = 3 | üs = 4 | sonuc = 1 | i = –

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
4     sonuc = 1
▶     Döngü i = 1'den başlayarak üs'e kadar
6         sonuc = sonuc * taban
7     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 1

Döngü: i = 1.

Adım 4 / 13

DEĞİŞKENLER

taban = 3 | üs = 4 | sonuc = 1 | i = 1

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
4     sonuc = 1
5     Döngü i = 1'den başlayarak üs'e kadar
▶         sonuc = sonuc * taban
7     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 3

sonuc = 1 × 3 = **3**.

Adım 5 / 13

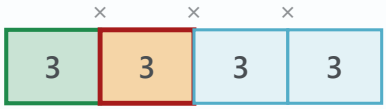
DEĞİŞKENLER

taban = 3 | üs = 4 | sonuc = 3 | i = 1

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
4     sonuc = 1
      Döngü i = 1'den başlayarak üs'e kadar
6         sonuc = sonuc * taban
7     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 3

Döngü: i = 2.

Adım 6 / 13

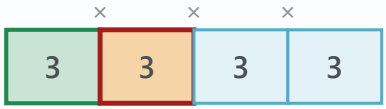
DEĞİŞKENLER

taban = 3 | üs = 4 | sonuc = 3 | i = 2

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
4     sonuc = 1
5     Döngü i = 1'den başlayarak üs'e kadar
6         sonuc = sonuc * taban
7     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 9

sonuc = 3 × 3 = **9**.

Adım 7 / 13

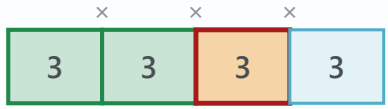
DEĞİŞKENLER

taban = **3** | üs = **4** | sonuc = **9** | i = **2**

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
4     sonuc = 1
▶     Döngü i = 1'den başlayarak üs'e kadar
6         sonuc = sonuc * taban
7     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 9

Döngü: i = 3.

Adım 8 / 13

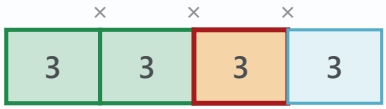
DEĞİŞKENLER

taban = 3 | üs = 4 | sonuc = 9 | i = 3

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
4     sonuc = 1
5     Döngü i = 1'den başlayarak üs'e kadar
6         sonuc = sonuc * taban
7     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 27

sonuc = 9 × 3 = **27**.

Adım 9 / 13

DEĞİŞKENLER

taban = **3** | üs = **4** | sonuc = **27** | i = **3**

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
4     sonuc = 1
      Döngü i = 1'den başlayarak üs'e kadar
6         sonuc = sonuc * taban
7     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 27

Döngü: i = 4.

Adım 10 / 13

DEĞİŞKENLER

taban = 3 | üs = 4 | sonuc = 27 | i = 4

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
4     sonuc = 1
5     Döngü i = 1'den başlayarak üs'e kadar
▶         sonuc = sonuc * taban
7     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 81

sonuc = 27 × 3 = **81**.

Adım 11 / 13

DEĞİŞKENLER

taban = **3** | üs = **4** | sonuc = **81** | i = **4**

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
4     sonuc = 1
5     Döngü i = 1'den başlayarak üs'e kadar
6         sonuc = sonuc * taban
▶     Döngü Sonu
8     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 81

i = 5 > 4: döngü biter.

Adım 12 / 13

DEĞİŞKENLER

taban = 3 | üs = 4 | sonuc = 81 | i = 5

ÇIKTI

Üs Alma: Elle İzleme (3^4)

```
1 Başla
2     Oku taban
3     Oku üs
4     sonuc = 1
5     Döngü i = 1'den başlayarak üs'e kadar
6         sonuc = sonuc * taban
7     Döngü Sonu
▶     Yaz "Sonuç: ", sonuc
9 Bitir
```



sonuc = 81

Sonuç yazdırılır: $3^4 = 81$.

Adım 13 / 13

DEĞİŞKENLER

`taban = 3` `üs = 4` `sonuc = 81` `i = 5`

ÇIKTI

```
Sonuç: 81
```

Arama Algoritmaları

- Bir koleksiyon içerisinde belirli bir öğenin **var olup olmadığını** (ve varsa yerini) sorgular.
- Veri işlemek ve bilgi edinmek için kullanılan temel bir işlemdir.
- Belirli öğelerin hızlı ve doğru bir şekilde bulunmasına yardımcı olur.
- **Kullanım alanları:** kitapların kütüphanedeki yerini bulma, bir web sitesinde belirli bir sayfayı bulma, veri tabanında belirli bir kaydı arama, telefon rehberinde bir isim arama.

Problem tanımı

Girdi:

- Nesnelerin bir koleksiyonu, "Sepet" diyelim.
- Belirli bir nesne, "Elma" diyelim.

Çıktı:

- "Elma" sepetin içindeyse `true`.
- "Elma" sepetin içinde değilse `false`.

Programlarda genellikle bulunan öğenin **indisi** (konumu), bulunamazsa `-1` döndürülür.

ARAMA ALGORİTMALARI

Üç Arama Yöntemi

Rastgele (Random)

Her adımda rastgele bir öğe seçip hedefle karşılaştırır.



Sıra gerekmez · bulma **garantisi yok**

Doğrusal (Linear)

Öğeleri baştan sona sırayla tek tek karşılaştırır.



Sıra gerekmez · en fazla **n** karşılaştırma

İkili (Binary)

Ortadaki öğeye bakar, her adımda arama alanının yarısını eler.



Dizi **sıralı olmalı** · en fazla ≈ $\log_2 n$ karşılaştırma

Rastgele Arama (Random Search)

- Koleksiyon içerisindeki öğeleri **rastgele** kontrol eder: her adımda rastgele bir öğe seçer.
- Hedefi bulana veya deneme hakkı tükenene kadar çalışır.
- Sözde kod:

1. Rastgele bir öğe seç.
2. Seçilen öğeyi hedefle karşılaştır.
3. Hedef bulunduysa işlemi sonlandır.
4. Bulunamadıysa adımları tekrarla.

Avantajlar

+ Basittir.
+ Verinin sırası önemli değildir.

Dezavantajlar

– Her zaman etkili bir çözüm sağlamaz: aynı öğeyi tekrar tekrar seçebilir.
– Bulma olasılığını artırabilir ama kesin bir sonuç garantisi vermez.
– Hedef yoksa sonsuza kadar sürer; deneme sınırı gerekir.
– En kötü durumda bütün koleksiyonu (hatta fazlasını) taramak gerekir.

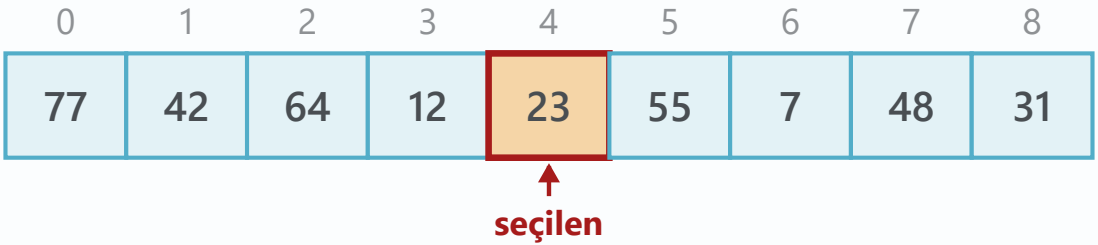
Rastgele Arama: Adım Adım



aranan öğe: 48

Elimizde sırasız bir dizi var; aranan öğe **48**.

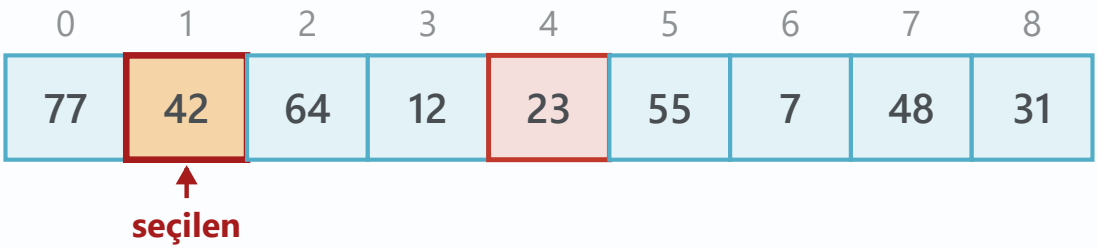
Rastgele Arama: Adım Adım



23 ile 48 karşılaştırılıyor

Adım 1: rastgele seçilen öğe 23; 23 ≠ 48, henüz bulunamadı. Yeni bir rastgele öğe seçilir.

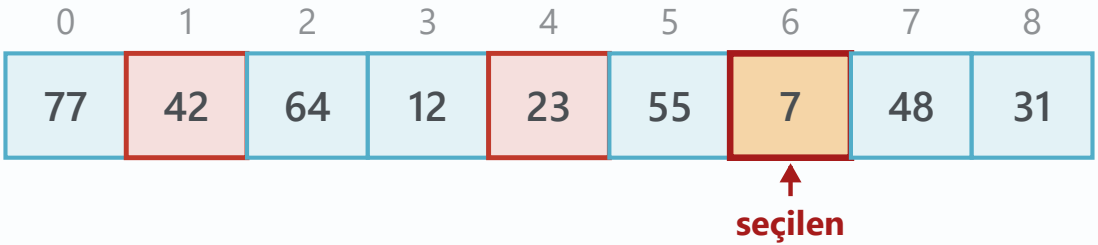
Rastgele Arama: Adım Adım



42 ile 48 karşılaştırılıyor

Adım 2: rastgele seçilen öğe 42; 42 ≠ 48, henüz bulunamadı. Yeni bir rastgele öğe seçilir.

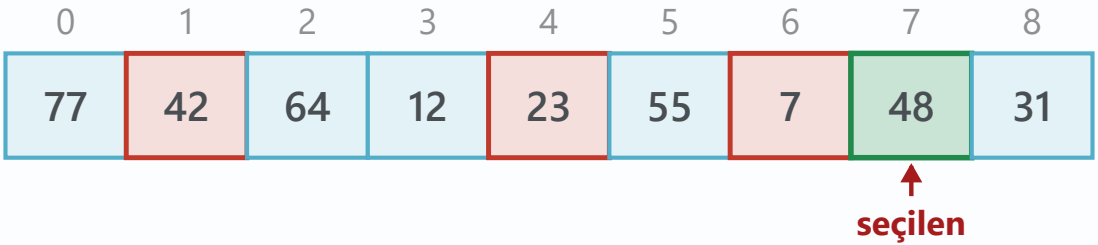
Rastgele Arama: Adım Adım



7 ile 48 karşılaştırılıyor

Adım 3: rastgele seçilen öğe 7; 7 ≠ 48, henüz bulunamadı. Yeni bir rastgele öğe seçilir.

Rastgele Arama: Adım Adım



48 ile 48 karşılaştırılıyor

Adım 4: rastgele seçilen öğe 48 → aranan öğe **bulundu**; arama biter (4 deneme).

Doğrusal Arama (Linear Search)

- Koleksiyon içinde bir öğeyi bulmanın en temel yoludur.
- Öğeleri **sırayla** kontrol ederek aranan öğeyi bulmaya çalışır.
- Öğelerin sıralı olması gerekmez.
- Küçük koleksiyonlarda etkili bir çözümdür.

Adımlar:

1. Öğeleri bir listeye (diziye) koy.
2. İlk öğeden başlayarak sırayla hedef öğe ile karşılaştır.
3. Hedef bulunursa işlemi sonlandır ve konumunu döndür.
4. Bulunmazsa bir sonraki öğeye geç.
5. Listenin sonuna kadar devam et; sona gelindiyse öğe yoktur.

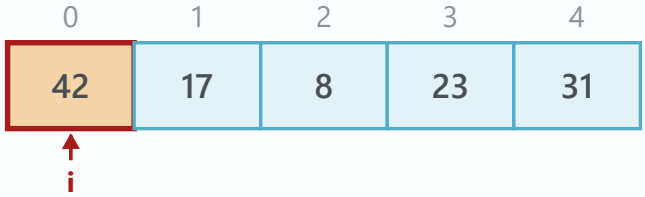
Avantajlar

+ Basit ve anlaşılır.
+ Herhangi bir sırayla (sırasız veride) çalışır.
+ Sonuç kesindir: öğe varsa mutlaka bulunur.

Dezavantajlar

– Büyük koleksiyonlarda yavaştır: her öğeyi kontrol etmesi gerekebilir (en kötü n karşılaştırma).

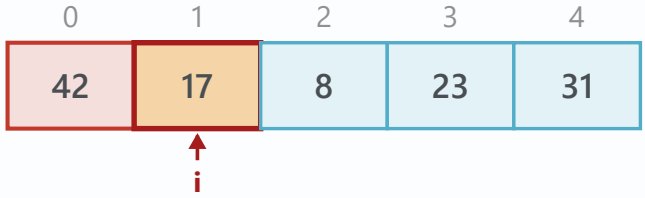
Doğrusal Arama: Adım Adım



dizi[0] = 42 ile 23 karşılaştırılıyor

Adım 1: aranan öğe 23. Dizinin ilk öğesinden (42) başlanır ve aranan öğe ile karşılaştırılır.

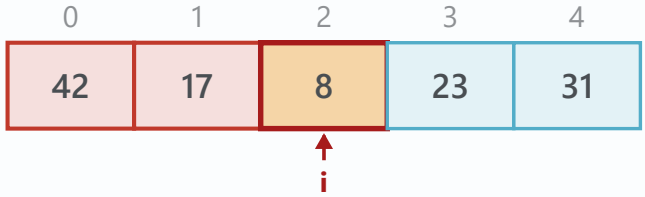
Doğrusal Arama: Adım Adım



dizi[1] = 17 ile 23 karşılaştırılıyor

Adım 2: ilk öğe (42) aranan öğe (23) ile eşleşmez. Bir sonraki öğeye geçilir: 17.

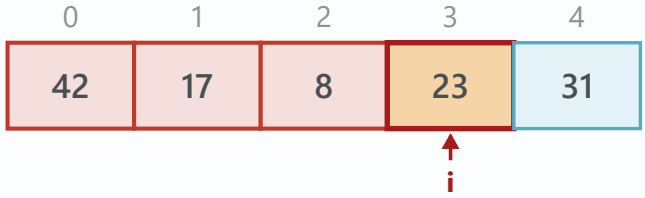
Doğrusal Arama: Adım Adım



dizi[2] = 8 ile 23 karşılaştırılıyor

Adım 3: ikinci öğe (17) de eşleşmez. Bir sonraki öğe: 8.

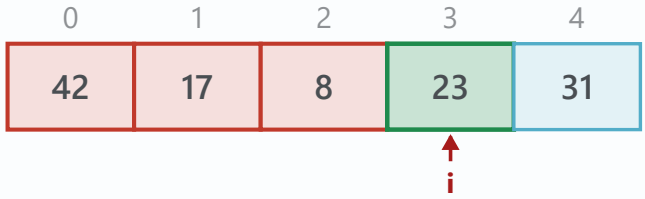
Doğrusal Arama: Adım Adım



dizi[3] = 23 ile 23 karşılaştırılıyor

Adım 4: üçüncü öğe (8) de eşleşmez. Bir sonraki öğe: 23.

Doğrusal Arama: Adım Adım



dizi[3] = 23: bulundu

Adım 5: dördüncü öğe (23) aranan öğe ile **eşleşir**; arama bu adımda biter (4 karşılaştırma).

ARAMA ALGORİTMALARI

İkili Arama (Binary Search)

- Bir koleksiyondaki öğeyi bulmanın hızlı ve etkili bir yoludur.
- **Sıralı** veriler (küçükten büyüğe veya büyükten küçüğe) üzerinde çalışır.
- Her adımda listenin **yarısını** eler.
- Büyük veri koleksiyonlarında arama için idealdir.

Adımlar:

1. Ortadaki öğeyi seç ve hedefle karşılaştır.
2. Hedef bulunursa işlemi sonlandır.
3. Hedef ortadakinden küçükse sol yarıya odaklan.
4. Hedef ortadakinden büyükse sağ yarıya odaklan.
5. Hedef bulunana veya arama alanı tükenene kadar tekrarla.

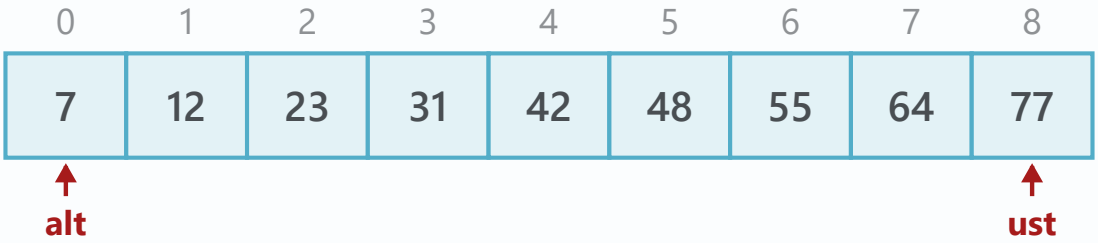
Avantajlar

+ Hızlıdır: 1 000 000 öğede en fazla 20 karşılaştırma.
+ Büyük veri koleksiyonlarında çok etkilidir.

Dezavantajlar

– Veriler sıralı olmalıdır (sıralamanın da bir maliyeti vardır).
– Sırasız dizide yanlış sonuç verebilir (hata vermeden!).

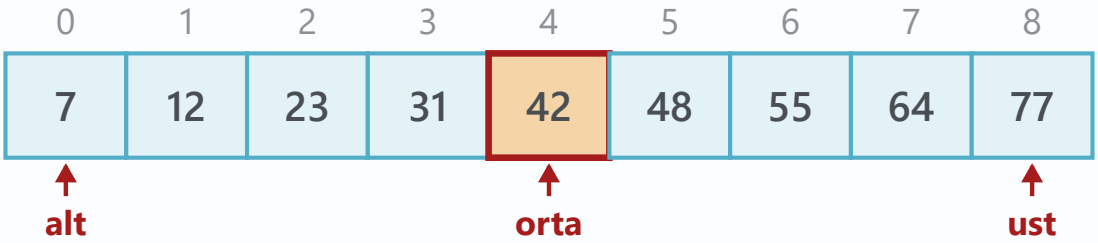
İkili Arama: Adım Adım



aranan öğe: 48

Elimizde sıralı bir dizi var; aranan öğe **48**. Arama alanı başta bütün dizi: alt = 0, üst = 8.

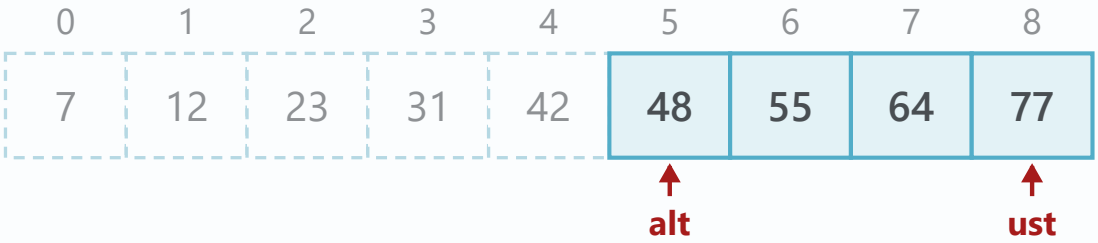
İkili Arama: Adım Adım



orta = (0 + 8) / 2 = 4 → 42

Adım 1: dizinin ortasındaki öğe (42) alınır ve aranan öğe (48) ile karşılaştırılır.

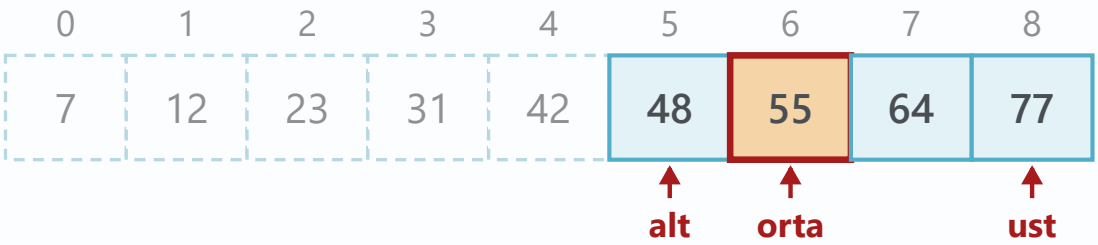
İkili Arama: Adım Adım



48 > 42 → sağ yarı

Adım 2: 48, ortadaki öğeden (42) büyüktür; aranan öğe sağ yarıdadır. Sol yarı artık kullanılmaz: [48, 55, 64, 77].

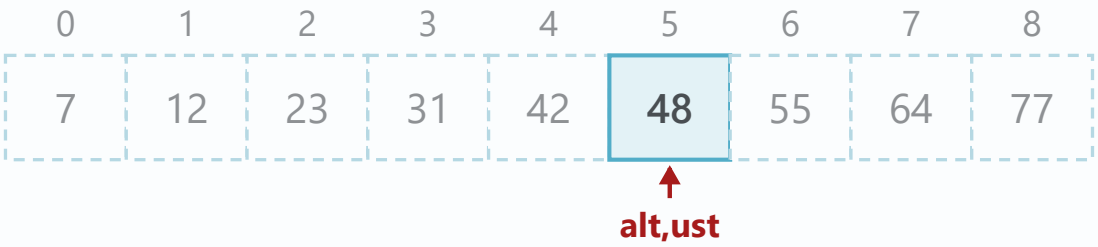
İkili Arama: Adım Adım



orta = (5 + 8) / 2 = 6 → 55

Adım 3: sağ yarının ortası alınır. 4 öğe olduğu için iki orta vardır (55 ve 64); `(alt + üst) / 2` soldakini, 55'i seçer.

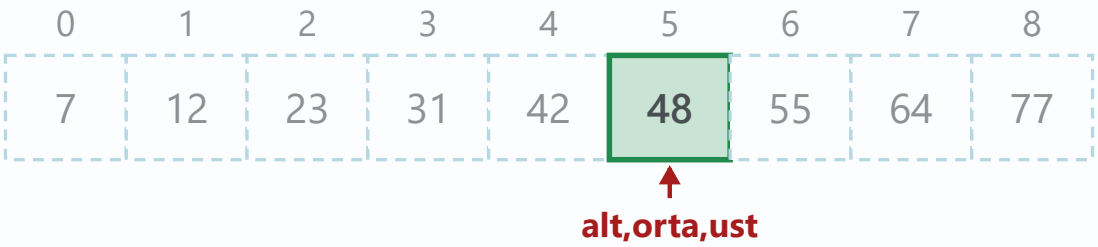
İkili Arama: Adım Adım



48 < 55 → sol yarı

Adım 4: 48, ortadaki öğeden (55) küçüktür; sağ taraf elenir. Geriye yalnızca [48] kalır.

İkili Arama: Adım Adım



orta = 5 → 48 = 48

Adım 5: ortadaki öğe 48, aranan öğeye eşittir: **bulundu**. 9 öğeli dizide yalnızca **3 karşılaştırma** yapıldı.

İkili Arama ve Doğrusal Arama: Karşılaştırma

Doğrusal arama

- Öğeleri tek tek, sırayla kontrol eder.
- Hedefi bulana ya da liste bitene dek sürer.
- Veri sırasız olabilir.
- Büyük koleksiyonlarda yavaş olabilir.

İkili arama

- Sıralı verilerde hedefi hızlıca bulur.
- Veri sıralı olmalı (artan ya da azalan).
- Her adımda kalan listenin yarısını eler.
- Büyük koleksiyonlarda çok hızlıdır.

En kötü durumda karşılaştırma sayısı

Öğe sayısı n	Doğrusal (n)	İkili ($\lfloor \log_2 n \rfloor + 1$)
9	9	4
1 000	1 000	10
1 000 000	1 000 000	20
1 000 000 000	1 000 000 000	30

n iki katına çıkınca doğrusal aramanın işi ikiye katlanır, ikili aramanınki **bir** artar.

ARAMA ALGORİTMALARI

Doğrusal ve İkili Arama: Etkileşimli Deney

Aranan 48 doğrusal ara ikili ara yeni dizi

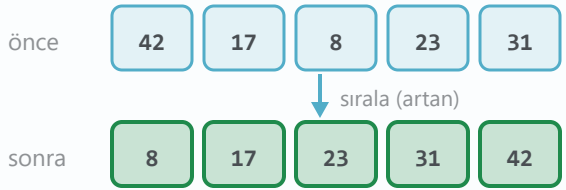
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	7	12	18	23	31	36	42	48	55	61	64	70	77	89

Sıralı bir dizi hazır. Bir sayı yazıp yöntem seçin.

Dizi sıralıdır, bu yüzden iki yöntem de doğru sonuç verir. Turuncu: incelenen öğe, kırmızı: elenen öğeler, yeşil: bulunan öğe. Farklı hedeflerde (ilk, son, olmayan bir sayı) karşılaştırma sayıları değişir.

Sıralama Algoritmaları

- Veri koleksiyonundaki öğeleri belirli bir kurala göre **düzenler**.
- Genellikle öğeleri küçükten büyüğe (artan) veya büyükten küçüğe (azalan) sıralar.
- Veri analizi ve arama işlemleri için temel bir adımdır.
- Sıralı veriler arama işlemlerini hızlandırır: ikili arama yalnızca sıralı dizide çalışır.



Rastgele (Random)

Öğeleri rastgele karıştırır; sıralamayı garanti etmez.

Seçimli (Selection Sort)

En küçüğü seçip başa koyar.

Kabarcık (Bubble Sort)

Komşuları karşılaştırıp büyüğü sona taşır.

Daha hızlı yöntemler de vardır (birleştirmeli, hızlı sıralama); Java'nın hazır `Arrays.sort` metodu bunların geliştirilmiş biçimlerini kullanır.

Rastgele Sıralama (Karıştırma, Shuffle)

- Bir veri koleksiyonunu rastgele bir düzene sokma işlemidir: öğeler rastgele **karıştırılır**.
- Verileri çeşitlendirmek ve analiz etmek için kullanılır: kart destesini karıştırma, sınav sorularının sırasını değiştirme, test verisi üretme.
- Bazı algoritmalar başlangıç verilerini rastgele karıştırarak (ör. hızlı sıralama) kötü durumlardan kaçınır.

Avantajlar

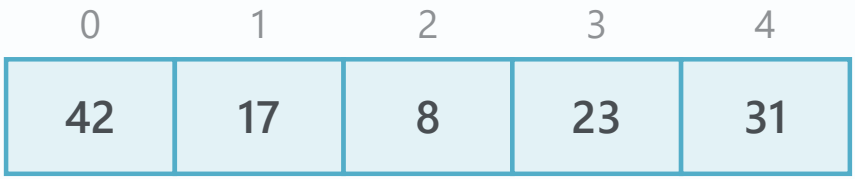
+ Verileri rastgele düzenleyerek her seferinde farklı sonuçlar elde edilir.
+ Java'da hazırdır: `Collections.shuffle(liste)`.

Dezavantajlar

– Verilerin doğru bir şekilde sıralanmasını **sağlamaz**.

– "Sıralanana kadar karıştır" yöntemi (*bogosort*) umutsuz derecede yavaştır: 10 öğenin 3 628 800 farklı dizilişinden yalnızca biri sıralıdır.

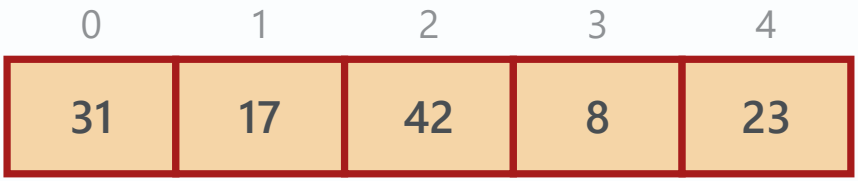
Rastgele Sıralama: Adım Adım



başlangıç dizisi

Elimizde sıralanmamış bir dizi var: [42, 17, 8, 23, 31].

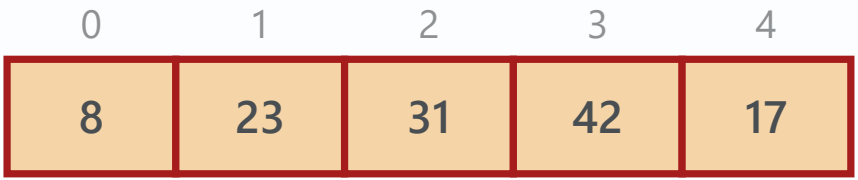
Rastgele Sıralama: Adım Adım



1. karıştırma

Adım 1: dizi öğeleri rastgele bir sırayla düzenlenir: [31, 17, 42, 8, 23]. Her düzenleme farklı bir sonuç üretir.

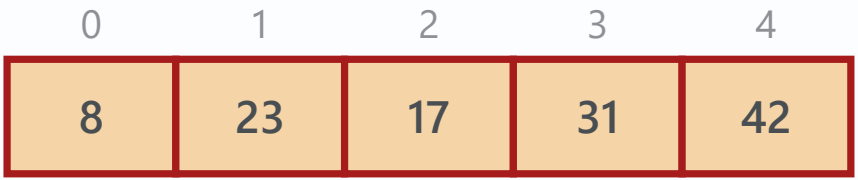
Rastgele Sıralama: Adım Adım



2. karıştırma

Adım 2: dizi öğeleri tekrar rastgele bir sırayla düzenlenir: [8, 23, 31, 42, 17].

Rastgele Sıralama: Adım Adım



3. karıştırma

Adım 3: dizi öğeleri tekrar rastgele bir sırayla düzenlenir: [8, 23, 17, 31, 42]. Sıraya yaklaştı ama hâlâ sıralı değil: karıştırma sıralamayı garanti etmez.

Seçimli Sıralama (Selection Sort)

- Öğeleri karşılaştırarak sıralayan basit bir algoritmadır.
- En küçük (veya en büyük) öğeyi bulup sırayla yer değiştirerek çalışır.

Adımlar:

1. Koleksiyondaki en küçük öğeyi bul.
2. Bu öğeyi koleksiyonun başına taşı (ilk öğeyle yer değiştir).
3. Geri kalan kısımdaki en küçük öğeyi bul.
4. Onu ikinci pozisyona taşı.
5. Bu adımları koleksiyonun sonuna kadar tekrarla.

Avantajlar

+ Basit ve anlaşılır.
+ Küçük veri koleksiyonlarında etkili.
+ En fazla n − 1 yer değiştirme yapar.

Dezavantajlar

– Büyük veri koleksiyonlarında yavaştır: her zaman n(n − 1)/2 karşılaştırma yapar (dizi sıralı olsa bile).

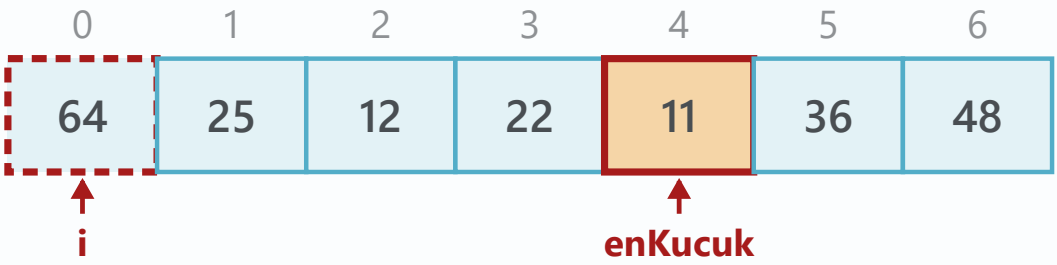
Seçimli Sıralama: Adım Adım



başlangıç

Elimizde sıralanmamış bir dizi var: [64, 25, 12, 22, 11, 36, 48].

Seçimli Sıralama: Adım Adım



sıralanmamış kısmın en küçüğü: 11

Adım 1: sıralanmamış kısmın (indis 0–6) en küçüğü **11**. 1. pozisyondaki 64 ile yer değiştirecek.

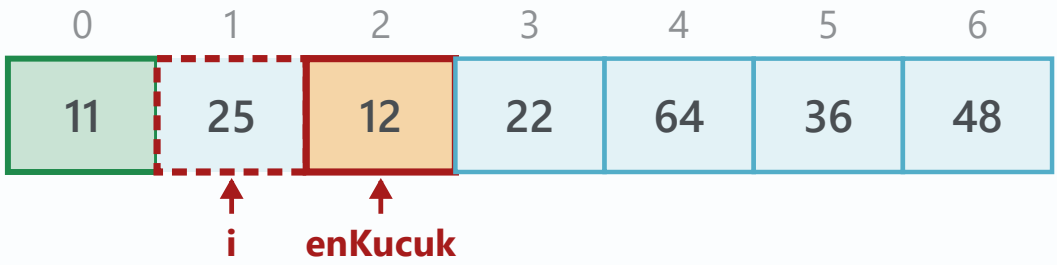
Seçimli Sıralama: Adım Adım



1. adımdan sonra

Dizi 1. adımdan sonra: [11, 25, 12, 22, 64, 36, 48]. Yeşil kısım artık sıralı ve yerinde.

Seçimli Sıralama: Adım Adım



sıralanmamış kısmın en küçüğü: 12

Adım 2: sıralanmamış kısmın (indis 1–6) en küçüğü **12**. 2. pozisyondaki 25 ile yer değiştirecek.

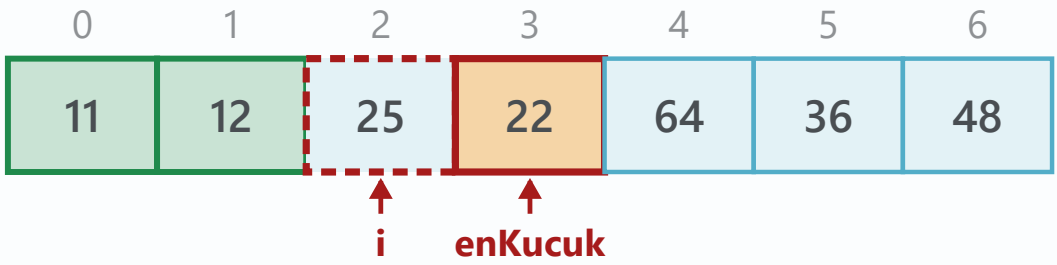
Seçimli Sıralama: Adım Adım



2. adımdan sonra

Dizi 2. adımdan sonra: [11, 12, 25, 22, 64, 36, 48]. Yeşil kısım artık sıralı ve yerinde.

Seçimli Sıralama: Adım Adım



sıralanmamış kısmın en küçüğü: 22

Adım 3: sıralanmamış kısmın (indis 2–6) en küçüğü **22**. 3. pozisyondaki 25 ile yer değiştirecek.

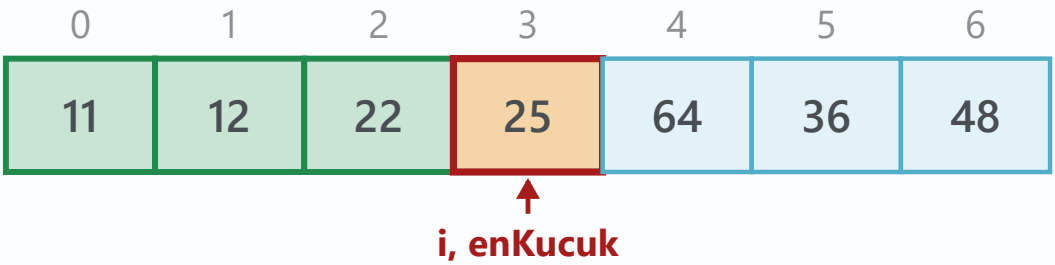
Seçimli Sıralama: Adım Adım



3. adımdan sonra

Dizi 3. adımdan sonra: [11, 12, 22, 25, 64, 36, 48]. Yeşil kısım artık sıralı ve yerinde.

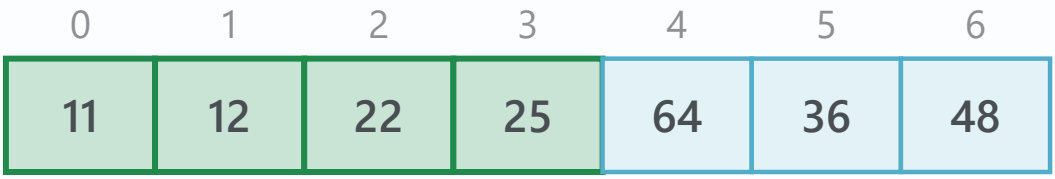
Seçimli Sıralama: Adım Adım



sıralanmamış kısmın en küçüğü: 25

Adım 4: sıralanmamış kısmın (indis 3–6) en küçüğü **25**. Zaten yerinde: değişiklik yok.

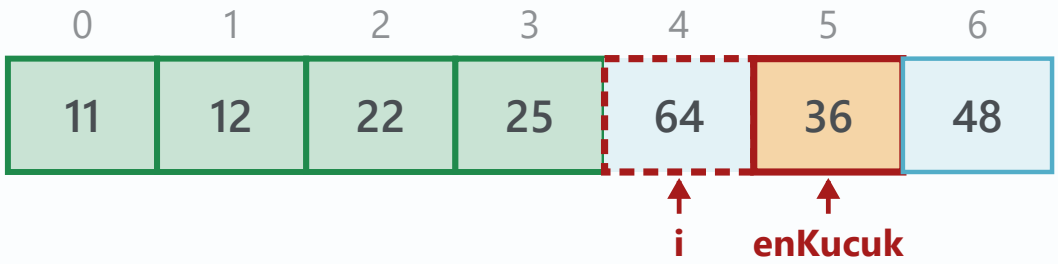
Seçimli Sıralama: Adım Adım



4. adımdan sonra

Dizi 4. adımdan sonra: [11, 12, 22, 25, 64, 36, 48]. Yeşil kısım artık sıralı ve yerinde.

Seçimli Sıralama: Adım Adım



sıralanmamış kısmın en küçüğü: 36

Adım 5: sıralanmamış kısmın (indis 4–6) en küçüğü **36**. 5. pozisyondaki 64 ile yer değiştirecek.

Seçimli Sıralama: Adım Adım



5. adımdan sonra

Dizi 5. adımdan sonra: [11, 12, 22, 25, 36, 64, 48]. Yeşil kısım artık sıralı ve yerinde.

Seçimli Sıralama: Adım Adım



sıralanmamış kısmın en küçüğü: 48

Adım 6: sıralanmamış kısmın (indis 5–6) en küçüğü **48**. 6. pozisyondaki 64 ile yer değiştirecek.

Seçimli Sıralama: Adım Adım



6. adımdan sonra

Dizi 6. adımdan sonra: [11, 12, 22, 25, 36, 48, 64]. Yeşil kısım artık sıralı ve yerinde. Son öğe kendiliğinden yerine oturdu: 7 öğe için **6 adım** yeterli.

Kabarcık Sıralaması (Bubble Sort)

- Öğeleri sırayla karşılaştırarak sıralayan basit bir algoritmadır.
- Yan yana duran öğeleri karşılaştırır; büyük olanı, suyun yüzeyine çıkan bir **kabarcık** gibi sona doğru taşır.

Adımlar:

1. Koleksiyonun başından başla.
2. İki yan yana öğeyi karşılaştır; gerekiyorsa yer değiştirip büyük olanı sağa taşı.
3. Koleksiyonun sonuna kadar bu adımı tekrarla (bir **tur**).
4. En büyük öğe sona geldiği için son öğeyi sıralanmış kabul et.
5. İlk dört adımı kalan kısım için tekrarla; bir turda hiç yer değiştirme olmazsa dur.

Avantajlar

+ Basit ve anlaşılır.
+ Küçük veya neredeyse sıralı koleksiyonlarda etkili (erken durma ile).

Dezavantajlar

– Büyük veri koleksiyonlarında çok yavaştır: en kötü durumda $n(n - 1)/2$ karşılaştırma ve aynı sayıda yer değiştirme.

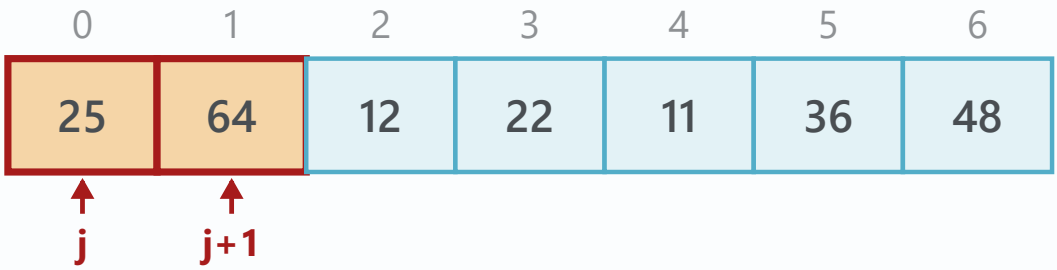
Kabarcık Sıralaması: Adım Adım



başlangıç

Elimizde sıralanmamış bir dizi var: [64, 25, 12, 22, 11, 36, 48].

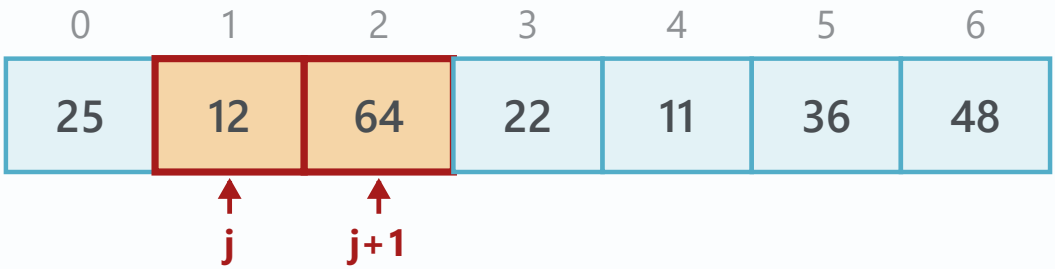
Kabarcık Sıralaması: Adım Adım



1. tur · yer değiştirdi

Adım 1 (1. tur): 64 ve 25 karşılaştırılır; 64 > 25 olduğu için yer değiştirir → [25, 64, 12, 22, 11, 36, 48].

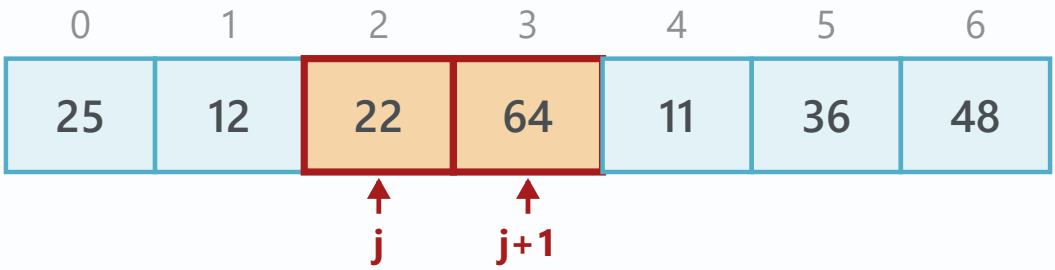
Kabarcık Sıralaması: Adım Adım



1. tur · yer değiştirdi

Adım 2 (1. tur): 64 ve 12 karşılaştırılır; 64 > 12 olduğu için yer değiştirir → [25, 12, 64, 22, 11, 36, 48].

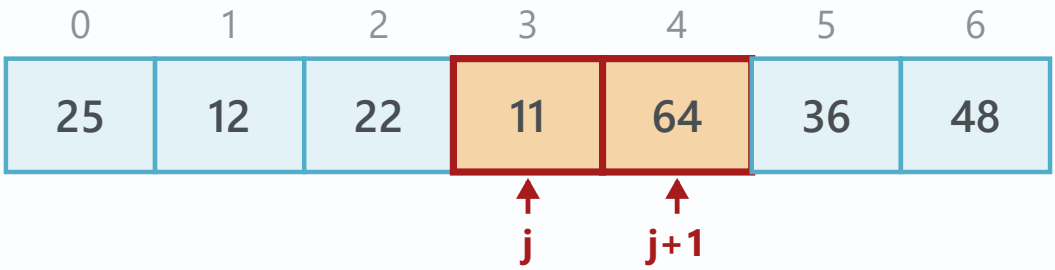
Kabarcık Sıralaması: Adım Adım



1. tur · yer değiştirdi

Adım 3 (1. tur): 64 ve 22 karşılaştırılır; 64 > 22 olduğu için yer değiştirir → [25, 12, 22, 64, 11, 36, 48].

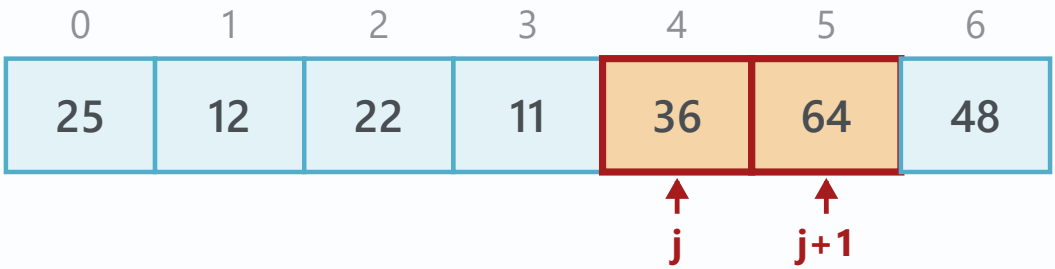
Kabarcık Sıralaması: Adım Adım



1. tur · yer değiştirdi

Adım 4 (1. tur): 64 ve 11 karşılaştırılır; 64 > 11 olduğu için yer değiştirir → [25, 12, 22, 11, 64, 36, 48].

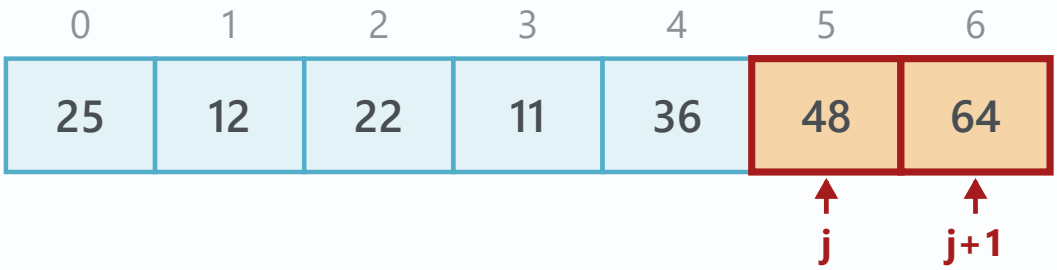
Kabarcık Sıralaması: Adım Adım



1. tur · yer değiştirdi

Adım 5 (1. tur): 64 ve 36 karşılaştırılır; 64 > 36 olduğu için yer değiştirir → [25, 12, 22, 11, 36, 64, 48].

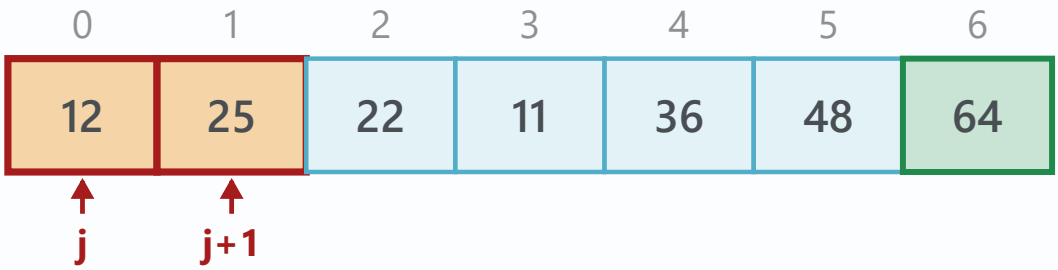
Kabarcık Sıralaması: Adım Adım



1. tur · yer değiştirdi

Adım 6 (1. tur): 64 ve 48 karşılaştırılır; 64 > 48 olduğu için yer değiştirir → [25, 12, 22, 11, 36, 48, 64].

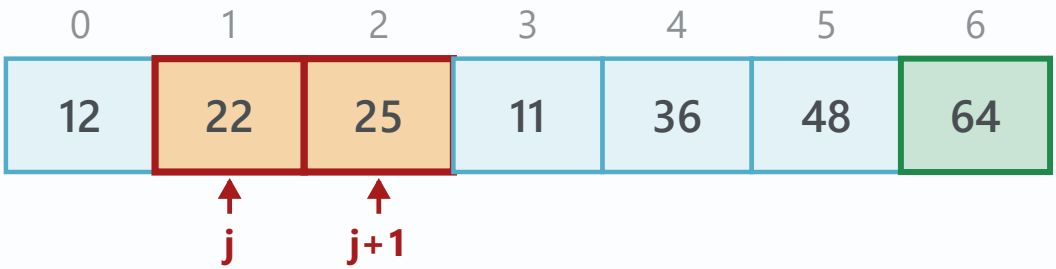
Kabarcık Sıralaması: Adım Adım



2. tur · yer değiştirdi

Adım 7 (2. tur): 25 ve 12 karşılaştırılır; 25 > 12 olduğu için yer değiştirir → [12, 25, 22, 11, 36, 48, 64].

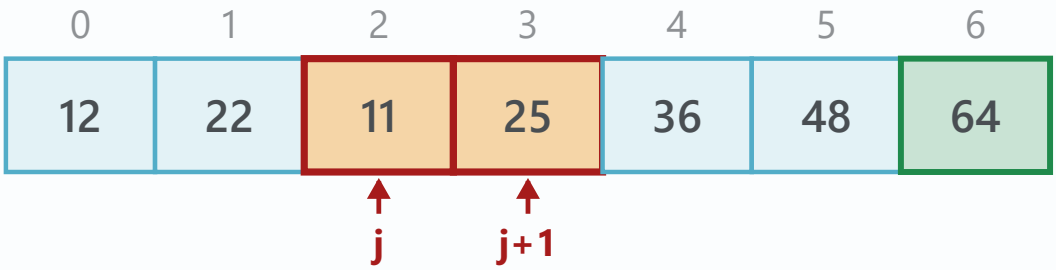
Kabarcık Sıralaması: Adım Adım



2. tur · yer değiştirdi

Adım 8 (2. tur): 25 ve 22 karşılaştırılır; 25 > 22 olduğu için yer değiştirir → [12, 22, 25, 11, 36, 48, 64].

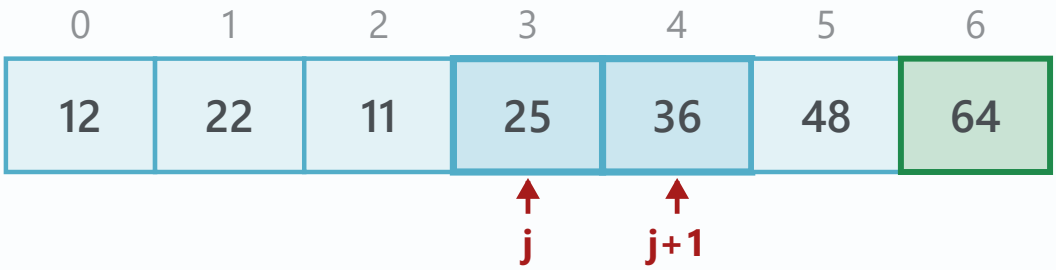
Kabarcık Sıralaması: Adım Adım



2. tur · yer değiştirdi

Adım 9 (2. tur): 25 ve 11 karşılaştırılır; 25 > 11 olduğu için yer değiştirir → [12, 22, 11, 25, 36, 48, 64].

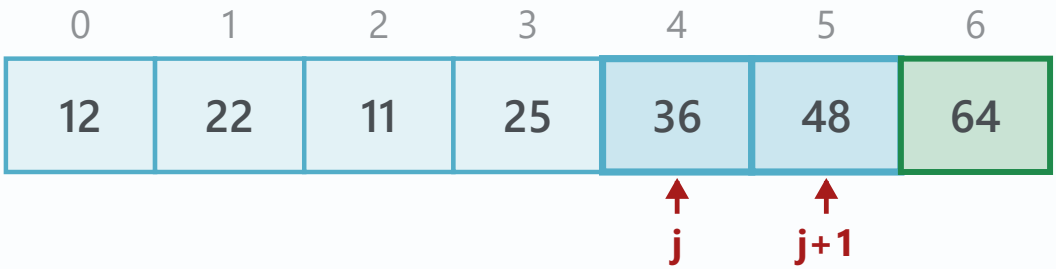
Kabarcık Sıralaması: Adım Adım



2. tur · değişiklik yok

Adım 10 (2. tur): 25 ve 36 karşılaştırılır; 25 < 36, değişiklik yok.

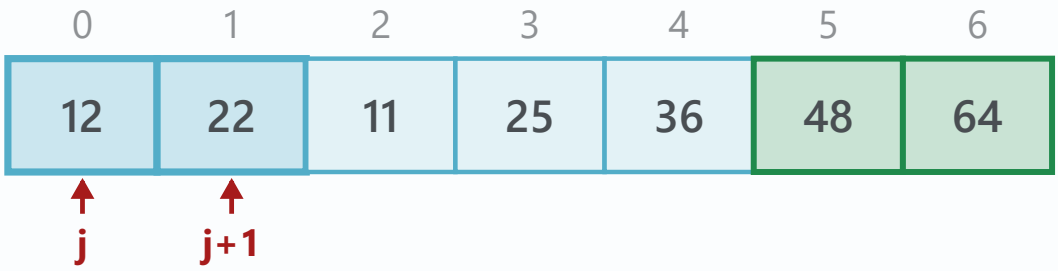
Kabarcık Sıralaması: Adım Adım



2. tur · değişiklik yok

Adım 11 (2. tur): 36 ve 48 karşılaştırılır; 36 < 48, değişiklik yok.

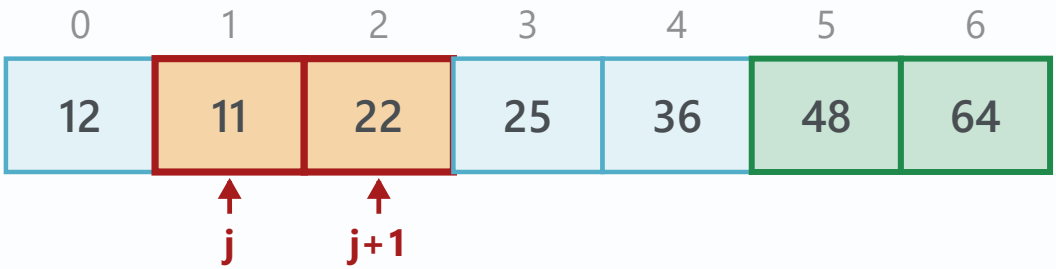
Kabarcık Sıralaması: Adım Adım



3. tur · değişiklik yok

Adım 12 (3. tur): 12 ve 22 karşılaştırılır; 12 < 22, değişiklik yok.

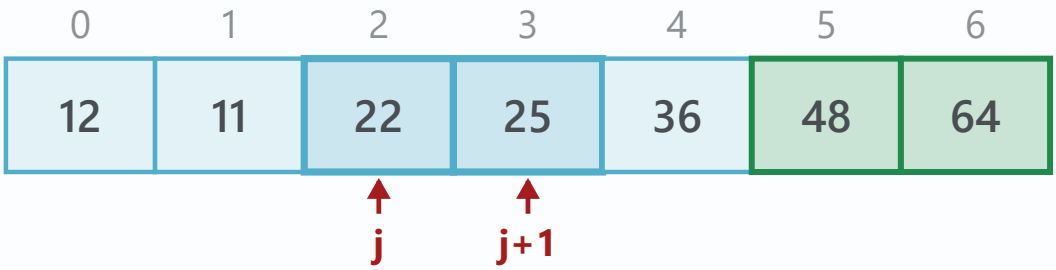
Kabarcık Sıralaması: Adım Adım



3. tur · yer değiştirdi

Adım 13 (3. tur): 22 ve 11 karşılaştırılır; 22 > 11 olduğu için yer değiştirir → [12, 11, 22, 25, 36, 48, 64].

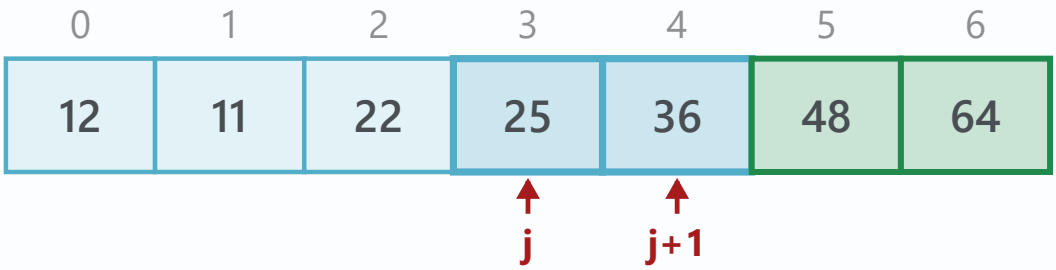
Kabarcık Sıralaması: Adım Adım



3. tur · değişiklik yok

Adım 14 (3. tur): 22 ve 25 karşılaştırılır; 22 < 25, değişiklik yok.

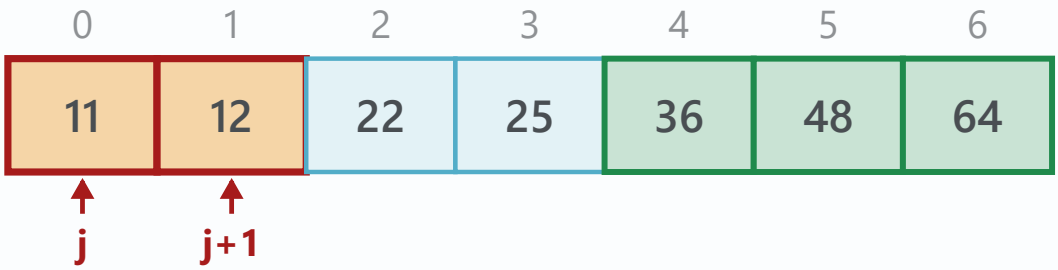
Kabarcık Sıralaması: Adım Adım



3. tur · değişiklik yok

Adım 15 (3. tur): 25 ve 36 karşılaştırılır; 25 < 36, değişiklik yok.

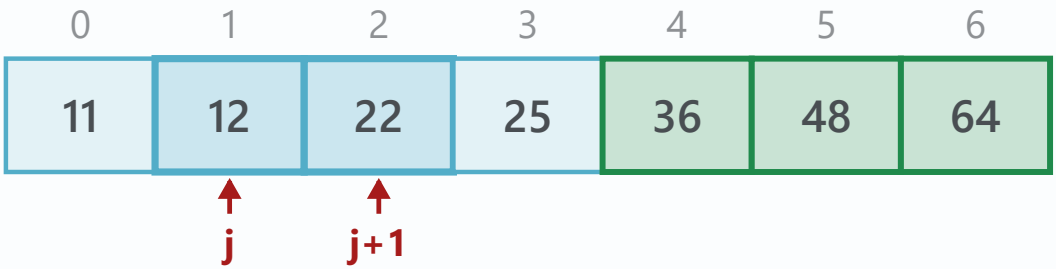
Kabarcık Sıralaması: Adım Adım



4. tur · yer değiştirdi

Adım 16 (4. tur): 12 ve 11 karşılaştırılır; 12 > 11 olduğu için yer değiştirir → [11, 12, 22, 25, 36, 48, 64].

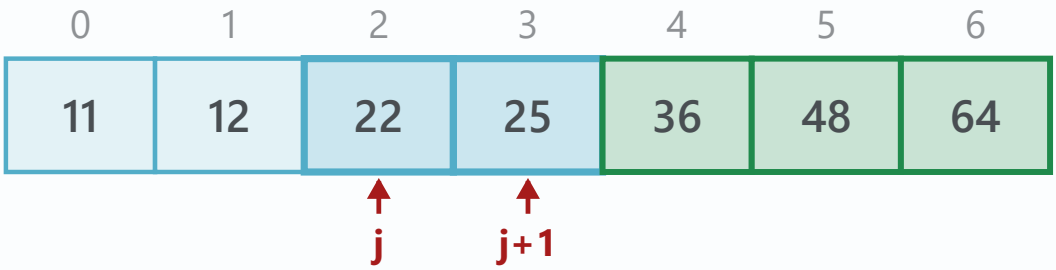
Kabarcık Sıralaması: Adım Adım



4. tur · değişiklik yok

Adım 17 (4. tur): 12 ve 22 karşılaştırılır; 12 < 22, değişiklik yok.

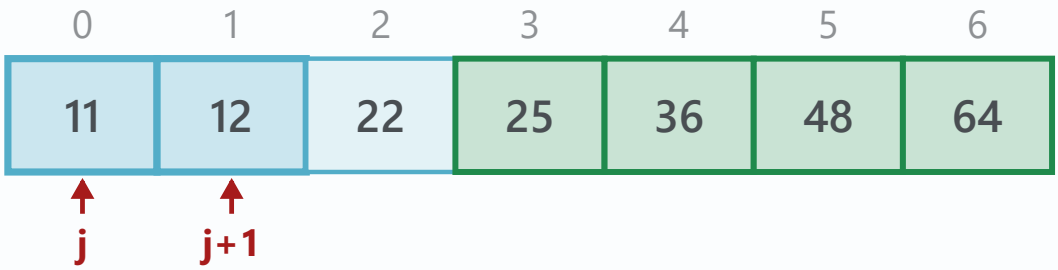
Kabarcık Sıralaması: Adım Adım



4. tur · değişiklik yok

Adım 18 (4. tur): 22 ve 25 karşılaştırılır; 22 < 25, değişiklik yok.

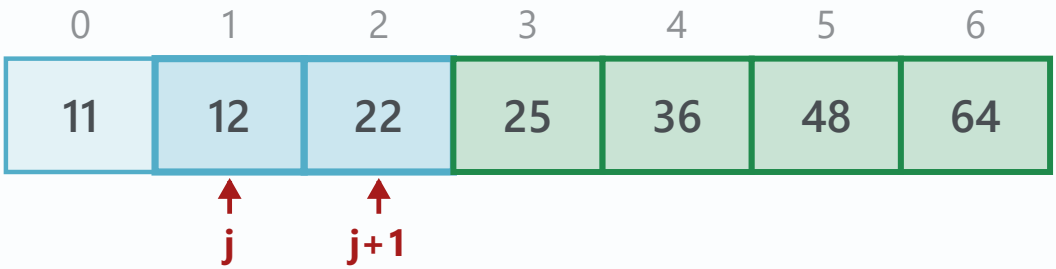
Kabarcık Sıralaması: Adım Adım



5. tur · değişiklik yok

Adım 19 (5. tur): 11 ve 12 karşılaştırılır; 11 < 12, değişiklik yok.

Kabarcık Sıralaması: Adım Adım



5. tur · değişiklik yok

Adım 20 (5. tur): 12 ve 22 karşılaştırılır; 12 < 22, değişiklik yok.

Kabarcık Sıralaması: Adım Adım



dizi sıralı

5. turda hiç yer değiştirme olmadı: dizi sıralıdır, algoritma durur. Toplam **20 karşılaştırma**, 11 yer değiştirme.

Sıralama Yöntemlerinin Karşılaştırması

	Rastgele (karıştırma)	**Seçimli**	**Kabarcık**
Fikir	Rastgele diziliş üret	En küçüğü seç, başa koy	Komşuları karşılaştır, büyüğü sona taşı
Sonuç sıralı mı?	Garanti yok	Evet	Evet
Karşılaştırma (n öğe)	—	her zaman $n(n-1)/2$	en kötü $n(n-1)/2$, sıralı dizide $n-1$
Yer değiştirme	$n-1$	en fazla $n-1$	en kötü $n(n-1)/2$
Örnek dizi ($n=7$)	—	21 karşılaştırma, 5 gerçek takas	20 karşılaştırma, 11 takas

Seçimli sıralamanın ve kabarcık sıralamasının işi n^2 ile büyür: 1 000 öğe için yaklaşık 500 000, 1 000 000 öğe için yaklaşık 500 milyar karşılaştırma. Bu yüzden büyük veride daha akıllı algoritmalar (ör. birleştirmeli sıralama, $n \cdot \log_2 n$) kullanılır.

Beş Sıralama Algoritmasının Karşılaştırması

Algoritma	Fikir	Ortalama	En kötü	Not
Kabarcık (Bubble Sort)	komşu iki öğe karşılaştırılır, gerekirse yer değiştirir	$O(n^2)$	$O(n^2)$	basit; büyük verilerde etkisiz
Seçimli (Selection Sort)	en küçük bulunur, başa yerleşir; kalanla tekrarlanır	$O(n^2)$	$O(n^2)$	büyük verilerde yavaş
Eklemeli (Insertion Sort)	sıradaki öğe sıralı kısımda uygun yere eklenir	$O(n^2)$	$O(n^2)$	küçük veya neredeyse sıralı veride etkili
Birleştirmeli (Merge Sort)	ikiye böl, her yarıyı özyinelemeli sırala, birleştir	$O(n \log n)$	$O(n \log n)$	kararlı; büyük verilerde etkili
Hızlı (Quick Sort)	pivot seçilir, küçükler sola büyükler sağa (böl ve fethet)	$O(n \log n)$	$O(n^2)$	ortalamada çok hızlı

$n \cdot \log n$ ile büyüyen algoritmalar 1 000 000 öğe için yaklaşık 20 milyon karşılaştırma yapar; n^2 ile büyüyenler için bu sayı yaklaşık 500 milyardır.

SIRALAMA ALGORİTMALARI

Seçimli ve Kabarcık Sıralaması: Etkileşimli Deney

Yöntem [seçimli sıralama] [bir adım] [sonuna kadar] [yeni dizi]

0	1	2	3	4	5	6	7
88	64	55	50	38	88	89	69

Seçmeli sıralama hazır. "bir adım" ile ilerleyin.

Her "bir adım" tek bir karşılaştırma yapar. Turuncu: karşılaştırılan öğeler, yeşil: kesin yerine oturmuş öğeler. Aynı dizide iki yöntemin karşılaştırma ve takas sayıları kıyaslanabilir.

ÖZET

Özet

Algoritma	Temel fikir	Kullanılan yapı	Örnek
İki / üç sayı sıralama	Karşılaştır, küçüğü önce yaz	Eğer … Değilse, iç içe karar	13, 11 → 11 13
Faktöriyel	1'den n'e kadar çarp	Döngü + biriktirici (1 ile başlar)	5! = 120
Fibonacci	Her terim önceki ikisinin toplamı	Döngü + değişken kaydırma	0 1 1 2 3
Palindrom	Temizle, ters çevir, karşılaştır	Metin işlemleri, karar	kayak → evet
Asal sayı	2 … n/2 arasında bölen ara	Bayrak + Döngüyü Kır	21 → değil
Üs alma	Tabanı üs kez çarp	Döngü + biriktirici	$3^4 = 81$
Rastgele / doğrusal / ikili arama	Rastgele dene / sırayla bak / yarıyı ele	Döngü, karar; ikilide sıralı dizi	9 öğede en çok 4 karşılaştırma (ikili)
Karıştırma / seçimli / kabarcık	Rastgele diz / en küçüğü seç / komşuları değiştir	İç içe döngü, takas	$n(n - 1)/2$ karşılaştırma

Her problemde aynı yol: sorunu ve testleri tanımla → sözde kod → elle izle → Java'ya çevir → testleri çalıştır.

ÖZET

Sık Yapılan Hatalar: Algoritma

Biriktiriciyi yanlış başlatmak

`faktoriyel = 0` ile başlanırsa her çarpım 0 olur. Toplama için 0, çarpma için 1 ile başlanır.

Döngü sınırında bir fazla/eksik

Fibonacci'de i = 2 … n − 1 yerine 2 … n yazılırsa n + 1 terim çıkar (off-by-one). Sınırlar küçük bir testle elle izlenir.

İkili aramayı sırasız dizide kullanmak

[77, 42, 64, 12, 23, 55, 7, 48, 31] içinde 77 aranınca `-1` döner: hata mesajı yok, sonuç yanlış.

Sınır durumları unutmak

Asal kontrolünde 0 ve 1, faktöriyelde 0!, Fibonacci'de n = 1 ayrıca test edilmeli.

Rastgele aramada durma koşulu yok

Hedef koleksiyonda yoksa döngü sonsuza kadar sürer; bir deneme sınırı gerekir.

Takası geçici değişkensiz yapmak

`a = b; b = a;` iki değişkeni de b yapar. Önce `gecici = a` saklanmalı.

ÖZET

Sık Yapılan Hatalar: Java

Hata	Ne olur?
`import java.util.Scanner;` unutuldu	Derleme hatası: `error: cannot find symbol … symbol: class Scanner`
Satır sonunda `;` yok: `int toplam = 0`	Derleme hatası: `error: ';' expected`
Karşılaştırmada tek `=` : `if (sayi % bolen = 0)`	Derleme hatası: `error: unexpected type … required: variable, found: value`. Eşitlik `==` ile sınanır.
Metinleri `==` ile karşılaştırmak	Derlenir ama "abcba" ile tersi için `false` verir. `girdi.equals(ters)` kullanılmalı.
`i <= dizi.length` ile döngü	Çalışma hatası: `ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5`
13! için `int` kullanmak	Hata mesajı yok, sonuç yanlış: 1932053504 (doğrusu 6227020800). `long` kullanılmalı.

ÖZET

Örnek Problemler

1. Faktöriyel algoritmasını **sayı = 4** için izleyin: her turda `i` ve `faktoriyel` değerleri nedir?
2. Fibonacci algoritması **n = 7** için ne yazdırır? Döngü kaç kez döner?
3. Asal sayı algoritması **25** için kaç bölen dener ve ne yazdırır?
4. [42, 17, 8, 23, 31] dizisinde doğrusal arama **31** ve **50** için kaç karşılaştırma yapar?
5. [7, 12, 23, 31, 42, 48, 55, 64, 77] dizisinde ikili arama **12** ve **77** için hangi öğelere bakar?
6. Üs alma algoritmasında taban = 2, üs = 10 için döngü kaç kez döner, sonuç nedir?

▶ **Cevapları göster**

ÖZET

Örnek Problemler: Sıralama

1. [29, 10, 14, 37, 13] dizisini seçimli sıralama ile sıralayın; her adımdan sonra diziyi yazın.
2. [5, 1, 4, 2, 8] dizisine kabarcık sıralaması uygulayın. Birinci turdan sonra dizi nasıldır? Erken durma ile kaç tur çalışır?
3. Seçimli sıralama 7 öğeli bir dizide kaç karşılaştırma yapar? Dizi zaten sıralı olsaydı sayı değişir miydi?
4. Kabarcık sıralaması sıralı bir dizide (erken durma ile) kaç karşılaştırma yapar?

▶ **Cevapları göster**

ÖZET

Alıştırmalar

1. Üç tam sayının **en büyüğünü** bulan sözde kodu ve Java programını yazın; 6 farklı sıra ile test edin.
2. 1'den n'e kadar sayıların toplamını döngüyle hesaplayın. n = 100 için sonucu n(n + 1)/2 formülüyle karşılaştırın.
3. Bir sayının basamakları toplamını bulun (ipucu: `% 10` ve `/ 10`). 1234 için sonuç?
4. Doğrusal aramayı, hedefin dizide **kaç kez** geçtiğini sayacak şekilde değiştirin.
5. Kabarcık sıralamasını **büyükten küçüğe** sıralayacak şekilde değiştirin.
6. Fibonacci algoritmasını Java'da `int` ile yazıp n = 50 için çalıştırın. Hangi terimden sonra sonuçlar bozulur? Neden?

▶ **Cevaplar / ipuçları**

Kaynaklar

- P. Deitel ve H. Deitel, *Java: How to Program (Early Objects)*, 11. baskı, Pearson, 2017 (Bölüm 4: Algoritmalar ve sözde kod; Bölüm 19: Arama ve sıralama).
- A. B. Downey ve C. Mayfield, *Think Java*, 2. baskı, O'Reilly, 2019 (çevrim içi ücretsiz).
- R. Sedgewick ve K. Wayne, *Algorithms*, 4. baskı, Addison-Wesley, 2011 (Bölüm 1.1 ikili arama, 2.1 temel sıralamalar).
- T. H. Cormen, C. E. Leiserson, R. L. Rivest ve C. Stein, *Introduction to Algorithms*, 4. baskı, MIT Press, 2022.
- D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, 2. baskı, Addison-Wesley, 1998.
- Oracle, *The Java Tutorials*: Language Basics, docs.oracle.com/javase/tutorial.

Son

Bölüm 1.3: Algoritmalar

Sonraki bölüm: Bölüm 1.4: Akış Diyagramları · Terim sözlüğü