

Bölüm 1: Algoritmalar

Java ile Nesne Yönelimli Programlama

Problem çözme adımları

Sözde kod

Koda dönüştürme

Fonksiyonlar

Örnek algoritmalar

Elle izleme

Arama algoritmaları

Sıralama algoritmaları

Sercan KÜLCÜ · Giresun Üniversitesi · Elektrik-Elektronik Mühendisliği

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; programı fonksiyonlara bölmek.

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 ve kabarcık sıralamayı bir dizi üzerinde adım adım uygulamak.

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 dönlür.

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 *çıkı*tıdır. "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.

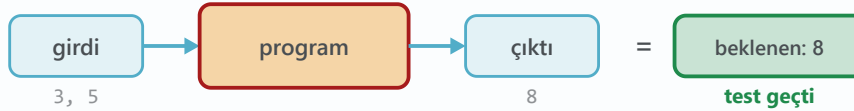
Etkinlik

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 durumlarıdır (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	Ekrana
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

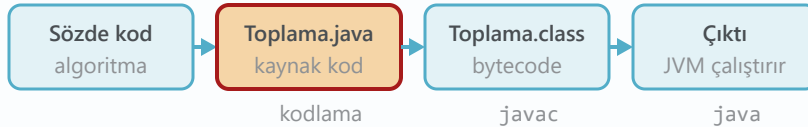
Algoritmayı İfade Etmek: Sözde Kod (Pseudocode)

Algoritma doğal dille, **sözde kodla**, akış diyagramıyla (sonraki bölüm) veya doğrudan bir programlama diliyle yazılabilir. Sözde kod, programlama diline benzeyen ama derleyici kuralları olmayan, insan için yazılmış bir gösterimdir.

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 çalıştırılabilir) 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	<code>sayi1 = tarayici.nextDouble();</code>
İkinci sayıyı al	<code>sayi2 = tarayici.nextDouble();</code>
İki sayıyı topla	<code>toplam = sayi1 + sayi2;</code>
Toplamı yazdır	<code>System.out.println(toplam);</code>

Java ayrıntıları (değişken tipleri, **Scanner**) sonraki bölümlerde ayrıca işlenecek; burada amaç algoritma ile kod arasındaki eşleşmeyi görmek.

Örnek Kod: İki Sayının Toplamı

```
import java.util.Scanner;

public class SumCalculator {
    public static void main(String[] args) {
        // Create a Scanner object to read user input
        Scanner scanner = new Scanner(System.in);
        // Prompt the user to enter the numbers
        System.out.print("Enter the first number: ");
        double num1 = scanner.nextDouble();
        System.out.print("Enter the second number: ");
        double num2 = scanner.nextDouble();
        // Calculate the sum of the two numbers
        double sum = num1 + num2;
        // Display the result
        System.out.println("The sum of " + num1 + " and " + num2 + " is: " + sum);
        // Close the Scanner
        scanner.close();
    }
}
```

```
Enter the first number: 3
Enter the second number: 5
The sum of 3.0 and 5.0 is: 8.0
```

Kütüphanelerin çoğu İngilizce adlar kullanır. Bu derste kendi adlarımızı Türkçe ama Türkçe karakter kullanmadan seçeceğiz.

Aynı Program, Türkçe Adlarla

```
import java.util.Scanner;

public class ToplamaHesaplama {
    public static void main(String[] args) {
        // Kullanıcı girdisini okumak için bir Scanner nesnesi oluştur
        Scanner tarayici = new Scanner(System.in);
        // Kullanıcıdan sayıları girmesini iste
        System.out.print("Birinci sayıyı giriniz: ");
        double sayi1 = tarayici.nextDouble();
        System.out.print("İkinci sayıyı giriniz: ");
        double sayi2 = tarayici.nextDouble();
        // İki sayının toplamını hesapla
        double toplam = sayi1 + sayi2;
        // Sonucu görüntüle
        System.out.println(sayi1 + " ile " + sayi2 + " toplamı: " + toplam);
        // Tarayıcıyı kapat
        tarayici.close();
    }
}
```

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

`import` satırı yazılmazsa derleyici `Scanner` 'ı bulamaz. `double` ondalıklı sayı tuttuğu için 3 girilse de `3.0` yazılır.

Toplama Programını Adım Adım İzle

```
► 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();
```

double sayi1

double sayi2

double toplam

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

Adım 1 / 8

DEĞİŞKENLER

tarayici = **Scanner**

ÇIKTI

Toplama Programını Adım Adım İzle

```
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();
```

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:

Toplama Programını Adım Adım İzle

```
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();
```

double sayi1

3.0

double sayi2

double toplam

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

Toplama Programını Adım Adım İzle

```
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();
```

double sayi1

3.0

double sayi2

double toplam

İkinci mesaj yazılır.

Adım 4 / 8

DEĞİŞKENLER

tarayici = **Scanner**

sayi1 = **3.0**

ÇIKTI

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

Toplama Programını Adım Adım İzle

```
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();
```

double sayi1

3.0

double sayi2

5.0

double toplam

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
```

Toplama Programını Adım Adım İzle

```
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();
```

double sayi1

3.0

double sayi2

5.0

double toplam

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
```

Toplama Programını Adım Adım İzle

```
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();
```

double sayi1

3.0

double sayi2

5.0

double toplam

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
```

Toplama Programını Adım Adım İzle

```
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

3.0

double sayi2

5.0

double toplam

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

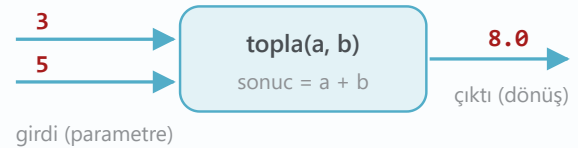
toplam = 8.0

ÇIKTI

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

Fonksiyonlar (Functions)

- Programları daha anlaşılır, sürdürülebilir ve verimli hale getirir.
- Bir grup deyim (*statements*) bir **ad verme** fırsatı sunar.
- Tekrarlı kodları ortadan kaldırarak programı küçük ve düzenli hale getirir.
- Uzun bir programı fonksiyonlara bölmek, her parçanın **ayrı ayrı test edilebilmesine** ve sonra çalışan bir bütün haline getirilmesine olanak tanır.
- İyi tasarlanmış fonksiyonlar başka projelerde **yeniden kullanılabilir**.



Fonksiyonu bir makine gibi düşünün: girdileri alır, işini yapar, sonucu geri verir. Java'da fonksiyonlara **metot** (method) denir.

Fonksiyon Örneği: topla(a, b)

```
1 static double topla(double a, double b) {  
2     double sonuc = a + b;  
3     return sonuc;  
4 }  
5  
▶ public static void main(String[] args) {  
7     double t1 = topla(3, 5);  
8     double t2 = topla(-2, 7);  
9     System.out.println("3 + 5 = " + t1);  
10    System.out.println("-2 + 7 = " + t2);  
11 }
```

Program her zaman `main` metodundan başlar; çağrı yığığında yalnızca `main` var.

Adım 1 / 10

ÇAĞRI YIĞINI

`main()`

DEĞİŞKENLER

—

ÇIKTI

Fonksiyon Örneği: topla(a, b)

```
1 static double topla(double a, double b) {  
2     double sonuc = a + b;  
3     return sonuc;  
4 }  
5  
6 public static void main(String[] args) {  
7     ▶ double t1 = topla(3, 5);  
8     double t2 = topla(-2, 7);  
9     System.out.println("3 + 5 = " + t1);  
10    System.out.println("-2 + 7 = " + t2);  
11 }
```

`topla(3, 5)` çağrılır: argümanlar 3 ve 5, parametreler `a` ve `b` 'ye kopyalanır.

Adım 2 / 10

ÇAĞRI YİĞİNİ

`topla(3.0, 5.0)`

`main()`

DEĞİŞKENLER

`a = 3.0`

`b = 5.0`

ÇIKTI

Fonksiyon Örneği: topla(a, b)

```
1 static double topla(double a, double b) {  
2     double sonuc = a + b;  
3     return sonuc;  
4 }  
5  
6 public static void main(String[] args) {  
7     double t1 = topla(3, 5);  
8     double t2 = topla(-2, 7);  
9     System.out.println("3 + 5 = " + t1);  
10    System.out.println("-2 + 7 = " + t2);  
11 }
```

`topla` 'nın içinde `sonuc` = $3.0 + 5.0 = 8.0$ hesaplanır.

Adım 3 / 10

ÇAĞRI YİĞİNİ

`topla(3.0, 5.0)`

`main()`

DEĞİŞKENLER

`a = 3.0`

`b = 5.0`

`sonuc = 8.0`

ÇIKTI

Fonksiyon Örneği: topla(a, b)

```
1 static double topla(double a, double b) {  
2     double sonuc = a + b;  
3     return sonuc;  
4 }  
5  
6 public static void main(String[] args) {  
7     double t1 = topla(3, 5);  
8     double t2 = topla(-2, 7);  
9     System.out.println("3 + 5 = " + t1);  
10    System.out.println("-2 + 7 = " + t2);  
11 }
```

return sonucu çağırana geri gönderir;
topla 'nın çerçevesi yığından kalkar.

Adım 4 / 10

ÇAĞRI YİĞİNİ

topla(3.0, 5.0)

main()

DEĞİŞKENLER

a = 3.0

b = 5.0

sonuc = 8.0

ÇIKTI

Fonksiyon Örneği: topla(a, b)

```
1 static double topla(double a, double b) {  
2     double sonuc = a + b;  
3     return sonuc;  
4 }  
5  
6 public static void main(String[] args) {  
7     ▶ double t1 = topla(3, 5);  
8     double t2 = topla(-2, 7);  
9     System.out.println("3 + 5 = " + t1);  
10    System.out.println("-2 + 7 = " + t2);  
11 }
```

Dönen 8.0 değeri `t1` 'e atanır. `a`, `b`,
`sonuc` artık yok.

Adım 5 / 10

ÇAĞRI YİĞİNİ

main()

DEĞİŞKENLER

t1 = 8.0

ÇIKTI

Fonksiyon Örneği: topla(a, b)

```
1 static double topla(double a, double b) {  
2     double sonuc = a + b;  
3     return sonuc;  
4 }  
5  
6 public static void main(String[] args) {  
7     double t1 = topla(3, 5);  
8     double t2 = topla(-2, 7);  
9     System.out.println("3 + 5 = " + t1);  
10    System.out.println("-2 + 7 = " + t2);  
11 }
```

Aynı fonksiyon bu kez -2 ve 7 ile çağrılır:
kod bir kez yazıldı, iki kez kullanılıyor.

Adım 6 / 10

ÇAĞRI YİĞİNİ

topla(-2.0, 7.0)

main()

DEĞİŞKENLER

t1 = 8.0

a = -2.0

b = 7.0

ÇIKTI

Fonksiyon Örneği: topla(a, b)

```
1 static double topla(double a, double b) {  
2     double sonuc = a + b;  
3     return sonuc;  
4 }  
5  
6 public static void main(String[] args) {  
7     double t1 = topla(3, 5);  
8     double t2 = topla(-2, 7);  
9     System.out.println("3 + 5 = " + t1);  
10    System.out.println("-2 + 7 = " + t2);  
11 }
```

sonuc = $-2.0 + 7.0 = 5.0$ hesaplanır ve döndürülür.

Adım 7 / 10

ÇAĞRI YİĞİNİ

topla(-2.0, 7.0)

main()

DEĞİŞKENLER

t1 = 8.0

a = -2.0

b = 7.0

sonuc = 5.0

ÇIKTI

Fonksiyon Örneği: topla(a, b)

```
1 static double topla(double a, double b) {  
2     double sonuc = a + b;  
3     return sonuc;  
4 }  
5  
6 public static void main(String[] args) {  
7     double t1 = topla(3, 5);  
8     double t2 = topla(-2, 7);  
9     System.out.println("3 + 5 = " + t1);  
10    System.out.println("-2 + 7 = " + t2);  
11 }
```

t2 = 5.0.

Adım 8 / 10

ÇAĞRI YİĞİNİ

main()

DEĞİŞKENLER

t1 = 8.0

t2 = 5.0

ÇIKTI

Fonksiyon Örneği: topla(a, b)

```
1 static double topla(double a, double b) {  
2     double sonuc = a + b;  
3     return sonuc;  
4 }  
5  
6 public static void main(String[] args) {  
7     double t1 = topla(3, 5);  
8     double t2 = topla(-2, 7);  
9     ▶ System.out.println("3 + 5 = " + t1);  
10    System.out.println("-2 + 7 = " + t2);  
11 }
```

Birinci sonuç yazdırılır.

Adım 9 / 10

ÇAĞRI YİĞİNİ

main()

DEĞİŞKENLER

t1 = 8.0

t2 = 5.0

ÇIKTI

3 + 5 = 8.0

Fonksiyon Örneği: topla(a, b)

```
1 static double topla(double a, double b) {  
2     double sonuc = a + b;  
3     return sonuc;  
4 }  
5  
6 public static void main(String[] args) {  
7     double t1 = topla(3, 5);  
8     double t2 = topla(-2, 7);  
9     System.out.println("3 + 5 = " + t1);  
▶ System.out.println("-2 + 7 = " + t2);  
11 }
```

İkinci sonuç yazdırılır; `main` biter ve program sonlanır.

Adım 10 / 10

ÇAĞRI YİĞİNİ

`main()`

DEĞİŞKENLER

`t1 = 8.0`

`t2 = 5.0`

ÇIKTI

`3 + 5 = 8.0`

`-2 + 7 = 5.0`

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, kabarcık sıralama.

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
```

birinciTamsayı

4

ikinciTamsayı

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
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

4

ikinciTamsayı

11

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

Adım 3 / 13

DEĞİŞKENLER

birinciTamsayı = 4

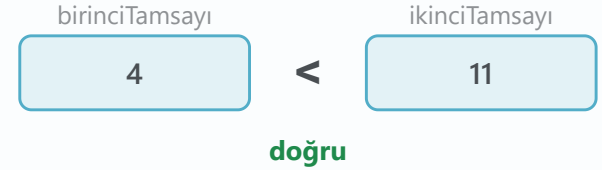
ikinciTamsayı = 11

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
6   ► Eğer birinciTamsayı < ikinciTamsayı ise
7       Yaz birinciTamsayı, " ", ikinciTamsayı
8   Değilse
9       Yaz ikinciTamsayı, " ", birinciTamsayı
10  Eğer Sonu
11 Bitir
```



Koşul: $4 < 11 \rightarrow$ **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
7     ► Yaz birinciTamsayı, " ", ikinciTamsayı
8   Değilse
9     Yaz ikinciTamsayı, " ", birinciTamsayı
10  Eğer Sonu
11 Bitir
```



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
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

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
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

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
6   ► Eğer birinciTamsayı < ikinciTamsayı ise
7       Yaz birinciTamsayı, " ", ikinciTamsayı
8   Değilse
9       Yaz ikinciTamsayı, " ", birinciTamsayı
10  Eğer Sonu
11 Bitir
```



Koşul: $13 < 11 \rightarrow$ 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
8 Değilse
9     Yaz ikinciTamsayı, " ", birinciTamsayı
10 Eğer Sonu
11 Bitir
```



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
9       Yaz ikinciTamsayı, " ", birinciTamsayı
10  Eğer Sonu
11 Bitir
```



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

13

ikinciTamsayı

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
```

İki Tam Sayı: Java Kodu

```
import java.util.Scanner;

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

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

```
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.

ÖRNEK ALGORİTMALAR

Üç Tam Sayı: Java Kodu ve Testler

```
import java.util.Scanner;

public class UcSayiSiralama {
    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 Kodunu İzle (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

7

ikinci

3

ucuncu

5

 $7 \leq 3$? yanlışGirdi 7, 3, 5. İlk koşul: $7 \leq 3$ **yanlış**;

&& 'nin sağ tarafına bakılmaz bile.

Adım 1 / 5

DEĞİŞKENLER

birinci = 7

ikinci = 3

ucuncu = 5

ÇIKTI

Üç Tam Sayı: Java Kodunu İzle (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);  
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

7

ikinci

3

ucuncu

5

 $3 \leq 7$ ve $3 \leq 5$: doğru

İkinci koşul: $3 \leq 7$ **doğru** ve $3 \leq 5$
doğru → en küçük **ikinci**.

Adım 2 / 5

DEĞİŞKENLER

birinci = 7

ikinci = 3

ucuncu = 5

ÇIKTI

Üç Tam Sayı: Java Kodunu İzle (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);
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

7

ikinci

3

ucuncu

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 Kodunu İzle (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);
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

7

ikinci

3

ucuncu

5

$7 \leq 5$? yanlış

Kalan iki sayı karşılaştırılır: $7 \leq 5$
yanlış.

Adım 4 / 5

DEĞİŞKENLER

birinci = 7

ikinci = 3

ucuncu = 5

ÇIKTI

3

Üç Tam Sayı: Java Kodunu İzle (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);
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

7

ikinci

3

ucuncu

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 girin
```

```
Oku sayı
```

```
// Sonucu saklamak için bir değişkeni başlatın
```

```
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: $n! = 1 \times 2 \times 3 \times \dots \times n$,
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

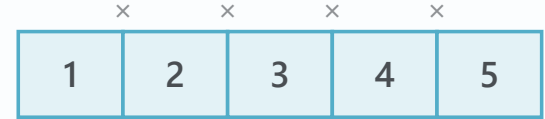
4 Döngü i = 1'den başlayarak sayı'ya kadar

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 = -

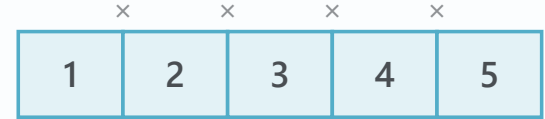
i = -

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
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

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

Adım 3 / 14

DEĞİŞKENLER

sayı = 5

faktoriyel = 1

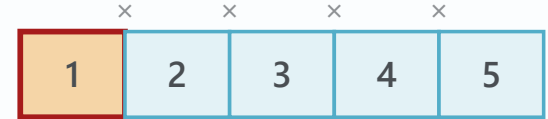
i = 1

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

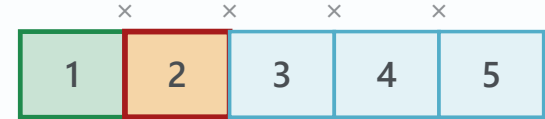
faktoriyel = 1

i = 1

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

```
1 Başla
2 Oku sayı
3 faktoriyel = 1
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

Döngü: i = 2 ($i \leq 5$ 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
5     faktoriyel = faktoriyel * i
6 Döngü Sonu
7 Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



$$\text{faktoriyel} = 1 \times 2 = 2$$

$$\text{faktoriyel} = 1 \times 2 = 2.$$

Adım 6 / 14

DEĞİŞKENLER

sayı = 5

faktoriyel = 2

i = 2

ÇIKTI



Faktöriyel: Elle İzleme (sayı = 5)

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



$$\text{faktoriyel} = 1 \times 2 = 2$$

Döngü: $i = 3$ ($i \leq 5$ 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
5     faktoriyel = faktoriyel * i
6 Döngü Sonu
7 Yaz "Girilen sayının faktöriyeli: ", faktoriyel
8 Bitir
```



$$\text{faktoriyel} = 1 \times 2 \times 3 = 6$$

$$\text{faktoriyel} = 2 \times 3 = 6.$$

Adım 8 / 14

DEĞİŞKENLER

sayı = 5

faktoriyel = 6

i = 3

ÇIKTI



Faktöriyel: Elle İzleme (sayı = 5)

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



$$\text{faktoriyel} = 1 \times 2 \times 3 = 6$$

Döngü: $i = 4$ ($i \leq 5$ olduğu için gövde çalışır).

Adım 9 / 14

DEĞİŞKENLER

sayı = 5

faktoriyel = 6

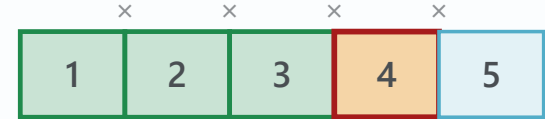
i = 4

ÇIKTI



Faktöriyel: Elle İzleme (sayı = 5)

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



$$\text{faktoriyel} = 1 \times 2 \times 3 \times 4 = 24$$

$$\text{faktoriyel} = 6 \times 4 = 24.$$

Adım 10 / 14

DEĞİŞKENLER

sayı = 5

faktoriyel = 24

i = 4

ÇIKTI

Faktöriyel: Elle İzleme (sayı = 5)

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



$$\text{faktoriyel} = 1 \times 2 \times 3 \times 4 = 24$$

Döngü: $i = 5$ ($i \leq 5$ 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
```



$$\text{faktoriyel} = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

$$\text{faktoriyel} = 24 \times 5 = \mathbf{120}.$$

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
```



$$\text{faktoriyel} = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

$i = 6$ olur; $6 \leq 5$ 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
```



$$\text{faktoriyel} = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

Sonuç yazdırılır: 5! = 120.

Adım 14 / 14

DEĞİŞKENLER

sayı = 5

faktoriyel = 120

i = 6

ÇIKTI

Girilen sayının faktöriyeli: 120

Faktöriyel: Java Kodu

```
import java.util.Scanner;

public class Faktoriyel {
    public static void main(String[] args) {
        Scanner tarayici = new Scanner(System.in);
        System.out.print("Bir sayı girin: ");
        int sayi = tarayici.nextInt();
        long faktoriyel = 1;
        for (int i = 1; i <= sayi; i++) {
            faktoriyel = faktoriyel * i;
        }
        System.out.println(sayi + "! = " + faktoriyel);
        tarayici.close();
    }
}
```

Bir sayı girin: 5
5! = 120

Bir sayı girin: 20
20! = 2432902008176640000

Neden long? int en fazla 2 147 483 647 tutar. 13! = 6 227 020 800 bu sınırı aşar; int ile hata vermeden yanlış sonuç (1 932 053 504) çıkar. long 20!'e kadar yeter.

ÖRNEK ALGORİTMALAR

Fibonacci Serisi Bulma

Başla

```
// Kaç terim istendiğini girin
Oku n
// İlk iki Fibonacci sayısını başlatın
a = 0
b = 1
Eğer n ≥ 1 ise Yaz a
Eğer n ≥ 2 ise Yaz b
// Kalan terimleri hesaplayın
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,

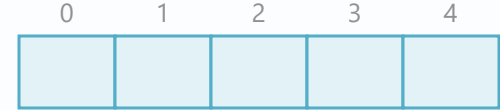
...

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 $i = 2 \dots n - 1$ 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
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
```



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

Adım 1 / 20

DEĞİŞKENLER

n = 5

a = -

b = -

i = -

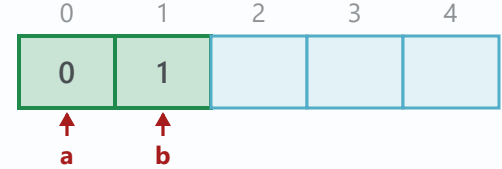
sonuc = -

ÇIKTI



Fibonacci: Elle İzleme (n = 5)

```
1 Başla
2 Oku n
3 ▶ a = 0
3 ▶ 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 = -

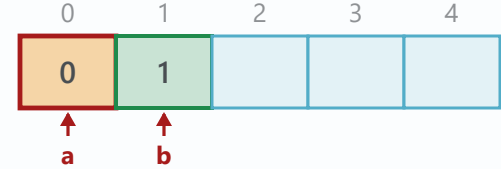
sonuc = -

ÇIKTI



Fibonacci: Elle İzleme (n = 5)

```
1 Başla
2 Oku n
3 a = 0
4 b = 1
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 ≥ 1: a yazılır.

Adım 3 / 20

DEĞİŞKENLER

n = 5

a = 0

b = 1

i = -

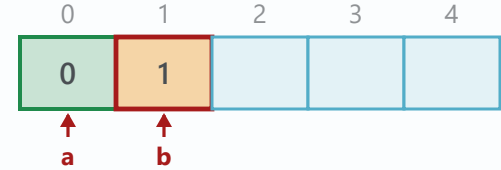
sonuc = -

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 = -

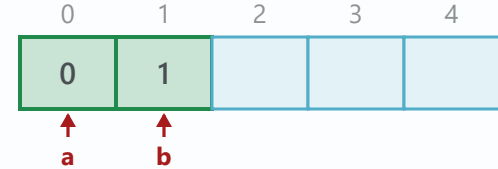
sonuc = -

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
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
```



Döngü: $i = 2$ ($i \leq n - 1 = 4$).

Adım 5 / 20

DEĞİŞKENLER

n = 5

a = 0

b = 1

i = 2

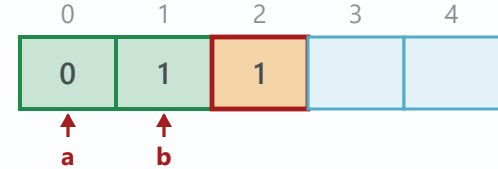
sonuc = -

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
8   ▶ 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

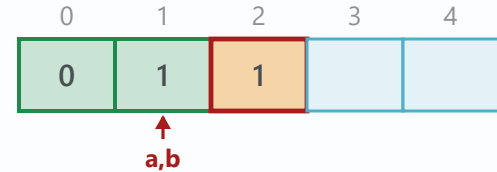
sonuc = 1

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
9   ▶ a = b
10  b = sonuc
11  Yaz sonuc
12 Döngü Sonu
13 Bitir
```



$a = b \rightarrow a = 1$ (bir adım ileri kaydı).

Adım 7 / 20

DEĞİŞKENLER

n = 5

a = 1

b = 1

i = 2

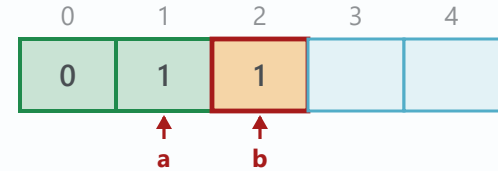
sonuc = 1

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
10    ► 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

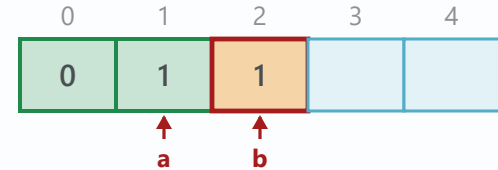
sonuc = 1

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
11    Yaz sonuc
12 Döngü Sonu
13 Bitir
```



Yeni terim yazılır: 1.

Adım 9 / 20

DEĞİŞKENLER

n = 5

a = 1

b = 1

i = 2

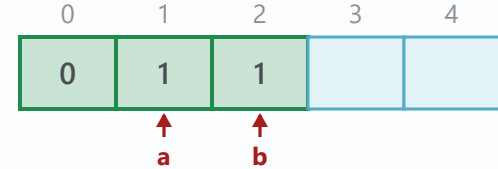
sonuc = 1

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
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
```



Döngü: $i = 3$ ($i \leq n - 1 = 4$).

Adım 10 / 20

DEĞİŞKENLER

n = 5

a = 1

b = 1

i = 3

sonuc = 1

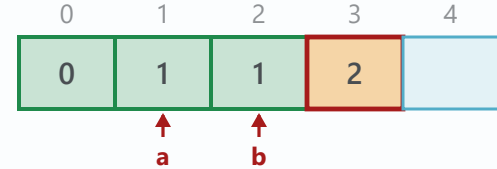
ÇIKTI

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
8   ▶ 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

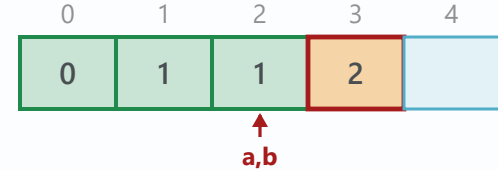
sonuc = 2

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
9   ► a = b
10  b = sonuc
11  Yaz sonuc
12 Döngü Sonu
13 Bitir
```



$a = b \rightarrow a = 1$ (bir adım ileri kaydı).

Adım 12 / 20

DEĞİŞKENLER

n = 5

a = 1

b = 1

i = 3

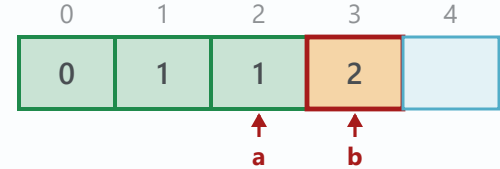
sonuc = 2

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
10    ► 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

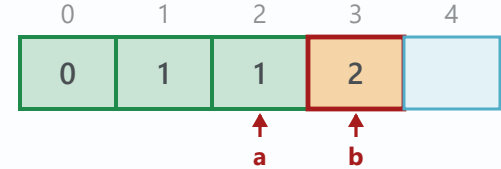
sonuc = 2

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
11    Yaz sonuc
12 Döngü Sonu
13 Bitir
```



Yeni terim yazılır: 2.

Adım 14 / 20

DEĞİŞKENLER

n = 5

a = 1

b = 2

i = 3

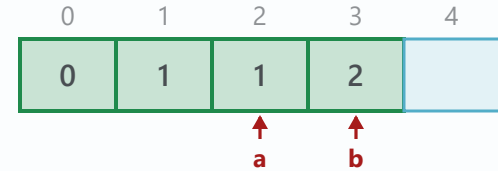
sonuc = 2

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
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
```



Döngü: $i = 4$ ($i \leq n - 1 = 4$).

Adım 15 / 20

DEĞİŞKENLER

n = 5

a = 1

b = 2

i = 4

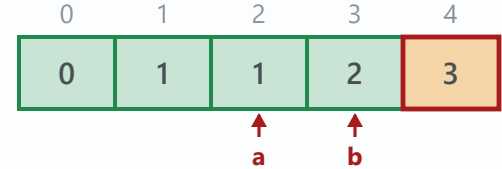
sonuc = 2

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
8   ▶ 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

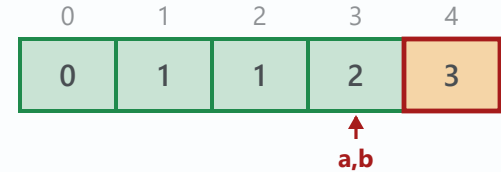
sonuc = 3

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
9     ► a = b
10    b = sonuc
11    Yaz sonuc
12 Döngü Sonu
13 Bitir
```



$a = b \rightarrow a = 2$ (bir adım ileri kaydı).

Adım 17 / 20

DEĞİŞKENLER

n = 5

a = 2

b = 2

i = 4

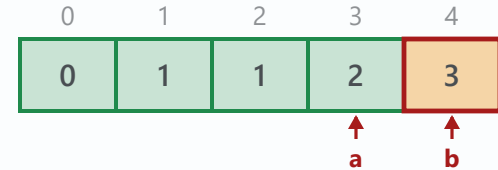
sonuc = 3

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
10    ► 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

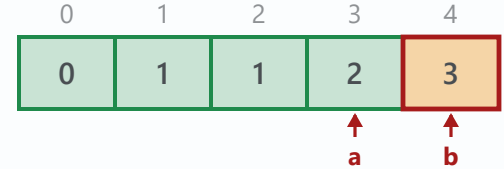
sonuc = 3

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
11    Yaz sonuc
12 Döngü Sonu
13 Bitir
```



Yeni terim yazılır: 3.

Adım 19 / 20

DEĞİŞKENLER

n = 5

a = 2

b = 3

i = 4

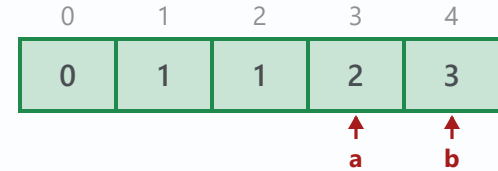
sonuc = 3

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; $5 \leq 4$ 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

Fibonacci: Java Kodu

```
import java.util.Scanner;

public class Fibonacci {
    public static void main(String[] args) {
        Scanner tarayici = new Scanner(System.in);
        System.out.print("Kaç terim? ");
        int n = tarayici.nextInt();
        int a = 0, b = 1, sonuc;
        if (n >= 1) System.out.print(a + " ");
        if (n >= 2) System.out.print(b + " ");
        for (int i = 2; i <= n - 1; i++) {
            sonuc = a + b;
            a = b;
            b = sonuc;
            System.out.print(sonuc + " ");
        }
        System.out.println();
        tarayici.close();
    }
}
```

Kaç terim? 5
0 1 1 2 3

Kaç terim? 10
0 1 1 2 3 5 8 13 21 34

`int a = 0, b = 1, sonuc;`

aynı tipten üç değişkeni tek satırda tanımlar. `print` terimleri yan yana yazar; en sondaki `println()` yalnızca satırı bitirir.

Palindrom Kontrolü

Başla

```
// Bir kelime veya cümlenin girilmesini isteyin
Oku girdiMetin
// Girdiyi temizleyin: küçük harf, boşluk ve noktalama yok
temizMetin = Temizle(girdiMetin)
// Temizlenmiş girdiyi ters çevirin
tersGirdi = TersÇevir(temizMetin)
// Temizlenmiş girdi ile ters girdiyi karşılaştırın
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
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
---	---	---	---	---

temizMetin

a	b	c	b	a
---	---	---	---	---

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)
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
```

temizMetin

a	b	c	b	a
---	---	---	---	---

tersGirdi

a	b	c	b	a
---	---	---	---	---

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)
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
```

temizMetin

a	b	c	b	a
---	---	---	---	---

tersGirdi

a	b	c	b	a
---	---	---	---	---

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
```

temizMetin

a	b	c	b	a
---	---	---	---	---

tersGirdi

a	b	c	b	a
---	---	---	---	---

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
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

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
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
temizMetin	k	i	t	a	p

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)
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
```

temizMetin

k	i	t	a	p
---	---	---	---	---

tersGirdi

p	a	t	i	k
---	---	---	---	---

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)
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
```

temizMetin

k	i	t	a	p
---	---	---	---	---

tersGirdi

p	a	t	i	k
---	---	---	---	---

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."
7 Değilse
8     Yaz "Girilen metin bir palindrom değildir."
9 Eğer Sonu
10 Bitir
```

temizMetin

k	i	t	a	p
---	---	---	---	---

tersGirdi

p	a	t	i	k
---	---	---	---	---

Koşul yanlışı: **Değilse** kolu çalışır.

Adım 10 / 10

DEĞİŞKENLER

girdiMetin = "Kitap"

temizMetin = "kitap"

tersGirdi = "patik"

ÇIKTI

Girilen metin bir palindrom
değildir.

ÖRNEK ALGORİTMALAR

Palindrom: Java Kodu

```
import java.util.Scanner;

public class Palindrom {
    public static void main(String[] args) {
        Scanner tarayici = new Scanner(System.in);
        System.out.print("Bir kelime veya cümle girin: ");
        String girdiMetin = tarayici.nextLine();
        String temiz = girdiMetin.toLowerCase().replaceAll("[^a-zçğışöü0-9]", "");
        String tersGirdi = new StringBuilder(temiz).reverse().toString();
        if (temiz.equals(tersGirdi)) {
            System.out.println("Girilen metin bir palindromdur.");
        } else {
            System.out.println("Girilen metin bir palindrom değildir.");
        }
        tarayici.close();
    }
}
```

```
Bir kelime veya cümle girin: Ey Edip Adana'da pide ye
Girilen metin bir palindromdur.
Bir kelime veya cümle girin: java
Girilen metin bir palindrom değildir.
```

`replaceAll("[^a-zçğışöü0-9]", "")` harf ve rakam dışındaki her şeyi siler, `StringBuilder` ters çevirir. Metinler `==` ile değil, `equals` ile karşılaştırılır.

Palindrom: Java Kodunu İzle

```
► String girdiMetin = "Kayak!";
2 String temiz = girdiMetin.toLowerCase()
3   .replaceAll("[^a-zçğışöşüö-9]", "");
4 String tersGirdi = new StringBuilder(temiz)
5   .reverse().toString();
6 if (temiz.equals(tersGirdi)) {
7   System.out.println("Girilen metin bir palindromdur.");
8 } else {
9   System.out.println("Girilen metin bir palindrom değildir.");
10 }
```

girdiMetin

K a y a k !

Girdi büyük harf ve noktalama içeriyor: "Kayak!". Doğrudan ters çevrilirse "!kayaK" olur.

Adım 1 / 6

DEĞİŞKENLER

girdiMetin = "Kayak!"

ÇIKTI

Palindrom: Java Kodunu İzle

```
1 String girdiMetin = "Kayak!";
2 String temiz = girdiMetin.toLowerCase()
3     .replaceAll("[^a-zçğiöşüö-9]", "");
4 String tersGirdi = new StringBuilder(temiz)
5     .reverse().toString();
6 if (temiz.equals(tersGirdi)) {
7     System.out.println("Girilen metin bir palindromdur.");
8 } else {
9     System.out.println("Girilen metin bir palindrom değildir.");
10 }
```

girdiMetin

K	a	y	a	k	!
---	---	---	---	---	---

küçük

k	a	y	a	k	!
---	---	---	---	---	---

`toLowerCase()` yeni bir dizgi
döndürür: `"kayak!"` . K → k.

Adım 2 / 6

DEĞİŞKENLER

`girdiMetin = "Kayak!"`

ÇIKTI

Palindrom: Java Kodunu İzle

```
1 String girdiMetin = "Kayak!";
2 String temiz = girdiMetin.toLowerCase()
3     .replaceAll("[^a-zçğışöşüö-9]", "");
4 String tersGirdi = new StringBuilder(temiz)
5     .reverse().toString();
6 if (temiz.equals(tersGirdi)) {
7     System.out.println("Girilen metin bir palindromdur.");
8 } else {
9     System.out.println("Girilen metin bir palindrom değildir.");
10 }
```

küçük

k a y a k !

temiz

k a y a k

replaceAll : harf ve rakam olmayan her karakter (!) silinir:

temiz = "kayak" .

Adım 3 / 6

DEĞİŞKENLER

girdiMetin = "Kayak!"

temiz = "kayak"

ÇIKTI



Palindrom: Java Kodunu İzle

```
1 String girdiMetin = "Kayak!";
2 String temiz = girdiMetin.toLowerCase()
3     .replaceAll("[^a-zçğiöşüö-9]", "");
▶ String tersGirdi = new StringBuilder(temiz)
▶     .reverse().toString();
6 if (temiz.equals(tersGirdi)) {
7     System.out.println("Girilen metin bir palindromdur.");
8 } else {
9     System.out.println("Girilen metin bir palindrom değildir.");
10 }
```

temiz

k a y a k

tersGirdi

k a y a k

StringBuilder karakterleri ters
sırayla dizer: "kayak".

Adım 4 / 6

DEĞİŞKENLER

temiz = "kayak"

tersGirdi = "kayak"

ÇIKTI



Palindrom: Java Kodunu İzle

```
1 String girdiMetin = "Kayak!";
2 String temiz = girdiMetin.toLowerCase()
3     .replaceAll("[^a-zçğıöşüö-9]", "");
4 String tersGirdi = new StringBuilder(temiz)
5     .reverse().toString();
▶ if (temiz.equals(tersGirdi)) {
7     System.out.println("Girilen metin bir palindromdur.");
8 } else {
9     System.out.println("Girilen metin bir palindrom değildir.");
10 }
```

temiz

k a y a k

tersGirdi

k a y a k

`equals` karakterleri tek tek karşılaştırır: hepsi aynı → `true`.
(`==` nesnelerin aynı olup olmadığına bakardı.)

Adım 5 / 6

DEĞİŞKENLER

temiz = "kayak"

tersGirdi = "kayak"

ÇIKTI

Palindrom: Java Kodunu İzle

```
1 String girdiMetin = "Kayak!";
2 String temiz = girdiMetin.toLowerCase()
3     .replaceAll("[^a-zçğışöü0-9]", "");
4 String tersGirdi = new StringBuilder(temiz)
5     .reverse().toString();
6 if (temiz.equals(tersGirdi)) {
7     System.out.println("Girilen metin bir palindromdur.");
8 } else {
9     System.out.println("Girilen metin bir palindrom değildir.");
10 }
```

temiz

k a y a k

Koşul doğru: ilk kol çalışır. **else**
kolu atlanır.

Adım 6 / 6

DEĞİŞKENLER

temiz = "kayak"

tersGirdi = "kayak"

ÇIKTI

Girilen metin bir
palindromdur.

Asal Sayı Kontrolü

Başla

```
// Bir sayıyı girin
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 yazdırın
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: $21 \% 3 = 0 \rightarrow$ 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
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
```

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
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

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
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:



$$7 \% 2 = 1$$

Döngü: bölen = 2 ($\leq 7 / 2 = 3$).

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
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

$7 \% 2 = 1 \rightarrow$ bölünmüyor

$7 \% 2 = 1 \neq 0 \rightarrow$ 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
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:



$$7 \% 3 = 1$$

Döngü: bölen = 3 ($\leq 7 / 2 = 3$).

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
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

$7 \% 3 = 1 \rightarrow$ bölünmüyor

$7 \% 3 = 1 \neq 0 \rightarrow$ 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
---	---

$7 \% 3 = 1 \rightarrow$ bölünmüyor

bölen = 4 olur; $4 \leq 3$ 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
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

$7 \% 3 = 1 \rightarrow$ 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
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
```

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
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

21 $\geq$ 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
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
---	---	---	---	---	---	---	---	----

$$21 \% 2 = 1$$

Döngü: bölen = 2 ($\leq 21 / 2 = 10$).

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
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
---	---	---	---	---	---	---	---	----

$21 \% 2 = 1 \rightarrow$ bölünmüyor

$21 \% 2 = 1 \neq 0 \rightarrow$ 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
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
---	---	---	---	---	---	---	---	----

$$21 \% 3 = 0$$

Döngü: bölen = 3 ($\leq 21 / 2 = 10$).

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
```

bölen:

2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	----

$21 \% 3 = 0 \rightarrow$ tam bölünüyor

$21 \% 3 = 0 \rightarrow$ 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
```

bölen:

2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	----

$21 \% 3 = 0 \rightarrow$ 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
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
---	---	---	---	---	---	---	---	----

$21 \% 3 = 0 \rightarrow$ 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
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
---	---	---	---	---	---	---	---	----

$21 \% 3 = 0 \rightarrow$ tam bölünüyor

Bayrak False: **Değilse** kolu çalışır.

Adım 17 / 17

DEĞİŞKENLER

sayı = 21

asalMi = False

bölen = 3

ÇIKTI

21 bir asal sayı değildir.

Asal Sayı: Java Kodu

```
import java.util.Scanner;

public class AsalSayi {
    public static void main(String[] args) {
        Scanner tarayici = new Scanner(System.in);
        System.out.print("Bir sayı girin: ");
        int sayi = tarayici.nextInt();
        boolean asalMi = true;           // bayrak (flag)
        if (sayi < 2) asalMi = false;     // 0 ve 1 asal değildir
        for (int bolen = 2; bolen <= sayi / 2; bolen++) {
            if (sayi % bolen == 0) {     // tam bölünüyor
                asalMi = false;
                break;                  // döngüyü kır
            }
        }
        if (asalMi) {
            System.out.println(sayi + " bir asal sayıdır.");
        } else {
            System.out.println(sayi + " bir asal sayı değildir.");
        }
        tarayici.close();
    }
}
```

Bir sayı girin: 21
21 bir asal sayı değildir.

Bir sayı girin: 97
97 bir asal sayıdır.

Daha hızlısı: bir sayının böleni varsa biri $\sqrt{\text{sayı}}$ 'dan küçük veya eşittir. Koşul

`bolen * bolen <= sayi`

olunca 97 için 47 yerine yalnızca 8 bölen denenir.

ÖRNEK ALGORİTMALAR

Üs Alma

Başla

```
// Bir sayıyı girin
```

```
Oku taban
```

```
// Bir üssü girin
```

```
Oku üs
```

```
// Sonucu saklamak için bir değişkeni başlatın
```

```
sonuc = 1
```

```
// Üssü kullanarak sonucu hesaplayın
```

```
Döngü i = 1'den başlayarak üs'e kadar
```

```
    sonuc = sonuc * taban
```

```
Döngü Sonu
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

Tanım: $\text{taban}^{\text{üs}} = \text{taban} \times \text{taban} \times \dots \times \text{taban}$ (üs kez); $\text{taban}^0 = 1$.

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 (3⁴)

1 Başla

► 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 (3⁴)

```
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 üs = 4 girer: 3⁴ hesaplanacak.

Adım 2 / 13

DEĞİŞKENLER

taban = 3

üs = 4

sonuc = -

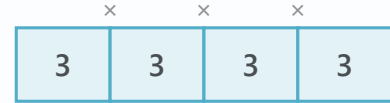
i = -

ÇIKTI



Üs Alma: Elle İzleme (3⁴)

```
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

Biriktirici 1 ile başlar.

Adım 3 / 13

DEĞİŞKENLER

taban = 3

üs = 4

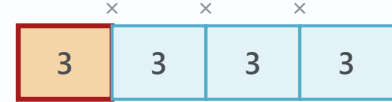
sonuc = 1

i = -

ÇIKTI

Üs Alma: Elle İzleme (3⁴)

```
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

Döngü: i = 1.

Adım 4 / 13

DEĞİŞKENLER

taban = 3

üs = 4

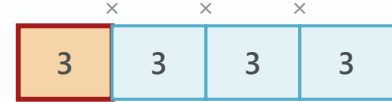
sonuc = 1

i = 1

ÇIKTI

Üs Alma: Elle İzleme (3⁴)

```
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 = 3

sonuc = 1 × 3 = 3.

Adım 5 / 13

DEĞİŞKENLER

taban = 3

üs = 4

sonuc = 3

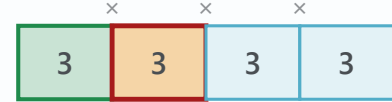
i = 1

ÇIKTI



Üs Alma: Elle İzleme (3⁴)

```
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 = 3

Döngü: i = 2.

Adım 6 / 13

DEĞİŞKENLER

taban = 3

üs = 4

sonuc = 3

i = 2

ÇIKTI



Üs Alma: Elle İzleme (3⁴)

```
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

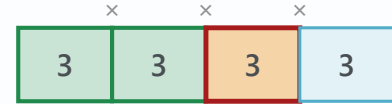
sonuc = 9

i = 2

ÇIKTI

Üs Alma: Elle İzleme (3⁴)

```
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

Döngü: i = 3.

Adım 8 / 13

DEĞİŞKENLER

taban = 3

üs = 4

sonuc = 9

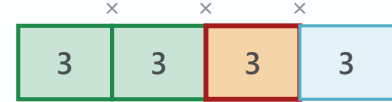
i = 3

ÇIKTI



Üs Alma: Elle İzleme (3⁴)

```
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

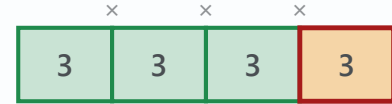
sonuc = 27

i = 3

ÇIKTI

Üs Alma: Elle İzleme (3⁴)

```
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

Döngü: i = 4.

Adım 10 / 13

DEĞİŞKENLER

taban = 3

üs = 4

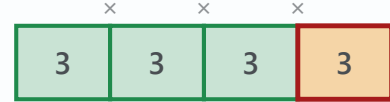
sonuc = 27

i = 4

ÇIKTI

Üs Alma: Elle İzleme (3⁴)

```
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 = 81

sonuc = 27 × 3 = **81**.

Adım 11 / 13

DEĞİŞKENLER

taban = 3

üs = 4

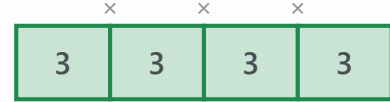
sonuc = **81**

i = 4

ÇIKTI

Üs Alma: Elle İzleme (3⁴)

```
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 = 81

i = 5 > 4: döngü biter.

Adım 12 / 13

DEĞİŞKENLER

taban = 3

üs = 4

sonuc = 81

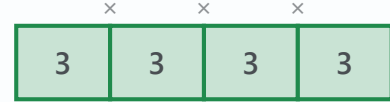
i = 5

ÇIKTI



Üs Alma: Elle İzleme (3⁴)

```
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 = 81

Sonuç yazdırılır: 3⁴ = 81.

Adım 13 / 13

DEĞİŞKENLER

taban = 3

üs = 4

sonuc = 81

i = 5

ÇIKTI

Sonuç: 81

ÖRNEK ALGORİTMALAR

Üs Alma: Java Kodu

```
import java.util.Scanner;

public class UsAlma {
    public static void main(String[] args) {
        Scanner tarayici = new Scanner(System.in);
        System.out.print("Taban ve üs girin: ");
        int taban = tarayici.nextInt();
        int us = tarayici.nextInt();
        long sonuc = 1;
        for (int i = 1; i <= us; i++) {
            sonuc = sonuc * taban;
        }
        System.out.println("Sonuç: " + sonuc);
        System.out.println("Math.pow: " + Math.pow(taban, us));
        tarayici.close();
    }
}
```

Taban ve üs girin: 3 4

Sonuç: 81

Math.pow: 81.0

Java'nın hazır

Math.pow(taban, us)

metodu aynı işi yapar ama sonucu **double** döndürür (81.0). Kendi döngümüz algoritmanın nasıl çalıştığını gösterir; gerçek programlarda hazır metotlar tercih edilir.

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.

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 $\approx \log_2 n$ 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

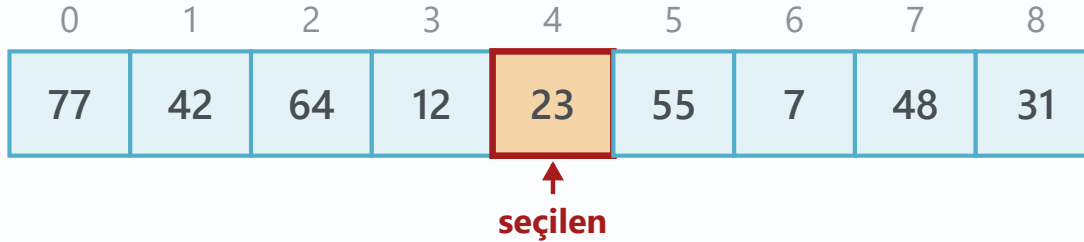
0	1	2	3	4	5	6	7	8
77	42	64	12	23	55	7	48	31

aranan öge: 48

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

Adım 1 / 5

Rastgele Arama: Adım Adım

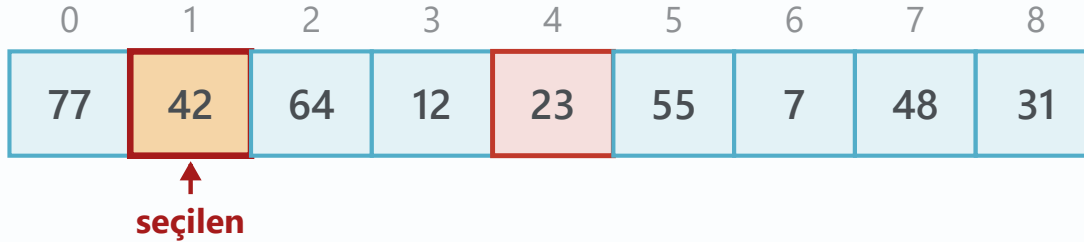


23 ile 48 karşılaştırılıyor

Adım 1: rastgele seçilen öge 23; $23 \neq 48$, henüz bulunamadı. Yeni bir rastgele öge seçilir.

Adım 2 / 5

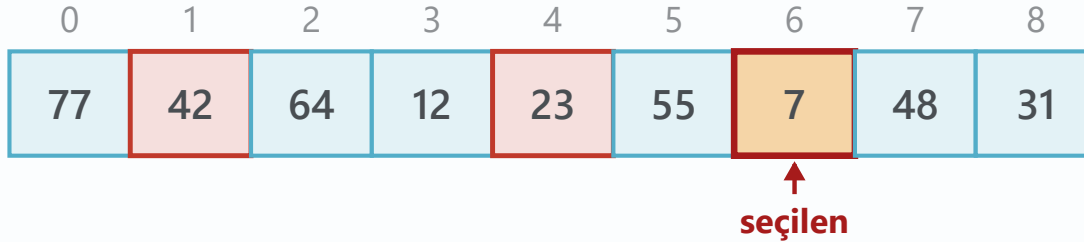
Rastgele Arama: Adım Adım



42 ile 48 karşılaştırılıyor

Adım 2: rastgele seçilen öge 42; $42 \neq 48$, henüz bulunamadı. Yeni bir rastgele öge seçilir.

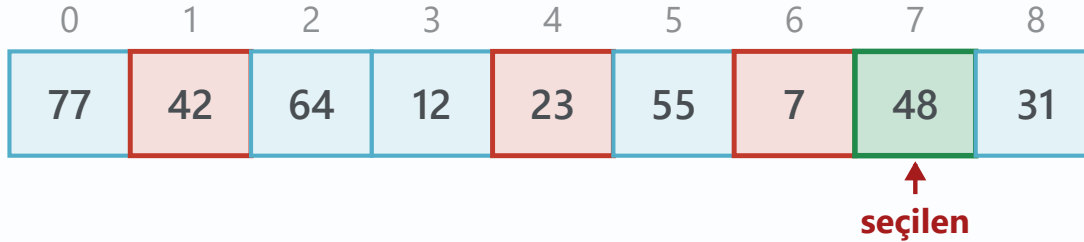
Rastgele Arama: Adım Adım



7 ile 48 karşılaştırılıyor

Adım 3: rastgele seçilen öge 7; $7 \neq 48$, henüz bulunamadı. Yeni bir rastgele öge seçilir.

Rastgele Arama: Adım Adım



48 ile 48 karşılaştırılıyor

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

Rastgele Arama: Java Kodu

```
import java.util.Random;

public class RastgeleArama {
    public static void main(String[] args) {
        int[] dizi = {77, 42, 64, 12, 23, 55, 7, 48, 31};
        int hedef = 48, deneme = 0;
        boolean bulundu = false;
        Random rastgele = new Random(18);    // sabit tohum
        while (!bulundu && deneme < 100) {    // deneme sınırı
            int i = rastgele.nextInt(dizi.length);    // 0..8
            deneme++;
            System.out.println(deneme + ". deneme: " + dizi[i]);
            if (dizi[i] == hedef) {
                bulundu = true;
            }
        }
        System.out.println(bulundu ? "Bulundu" : "Bulunamadı");
    }
}
```

```
1. deneme: 55
2. deneme: 31
3. deneme: 31
4. deneme: 7
5. deneme: 31
6. deneme: 48
Bulundu
```

`new Random(18)` sabit bir tohum (seed) kullandığı için çıktı hep aynıdır; `new Random()` her seferinde farklı sonuç verir. 31 **üç kez** seçildi: 9 ögeli dizide 6 deneme gerekti.

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) koyun.
2. İlk öğeden başlayarak sırayla hedef öğe ile karşılaştırın.
3. Hedef bulunursa işlemi sonlandırın ve konumunu döndürün.
4. Bulunmazsa bir sonraki öğeye geçin.
5. Listenin sonuna kadar devam edin; sona gelindiye öğe yoktur.

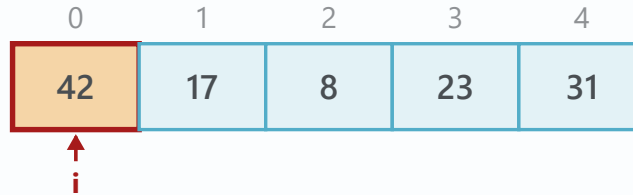
Avantajlar

- + 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).

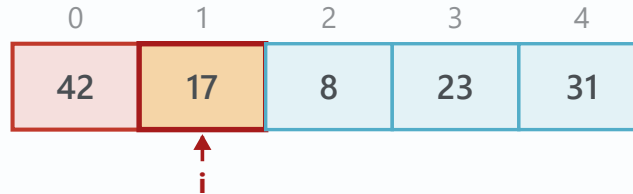
Doğrusal Arama: Adım Adım



dizi[0] = 42 ile 23 karşılaştırılıyor

Adım 1: aranan öge 23. Dizinin ilk ögesinden (42) başlanır ve aranan öge ile karşılaştırılır.

Doğrusal Arama: Adım Adım

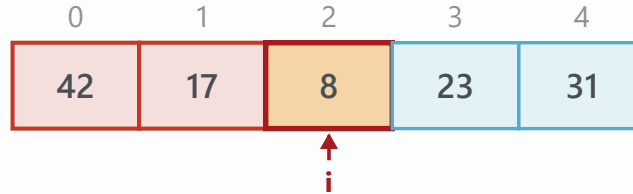


dizi[1] = 17 ile 23 karşılaştırılıyor

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

Adım 2 / 5

Doğrusal Arama: Adım Adım

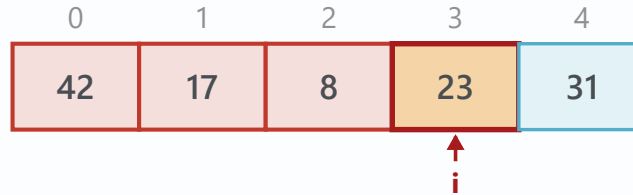


dizi[2] = 8 ile 23 karşılaştırılıyor

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

Adım 3 / 5

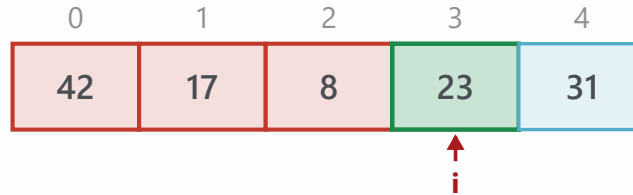
Doğrusal Arama: Adım Adım



$dizi[3] = 23$ ile 23 karşılaştırılıyor

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

Doğrusal Arama: Adım Adım



dizi[3] = 23: bulundu

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

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i;        // bulundu: indis  
5         }  
6     }  
7     return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     ▶ int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```

0	1	2	3	4
42	17	8	23	31

main diziye oluşturur.

Adım 1 / 16

ÇAĞRI YİĞİNİ

main()

DEĞİŞKENLER

dizi = [42, 17, 8, 23, 31]

ÇIKTI

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i;        // bulundu: indis  
5         }  
6     }  
7     return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     ▶ System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```

0	1	2	3	4
42	17	8	23	31

dogrusalAra(dizi, 23) çağrılır.

Adım 2 / 16

ÇAĞRI YİĞİNİ

dogrusalAra(dizi, 23)

main()

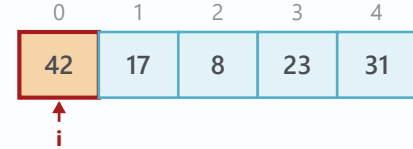
DEĞİŞKENLER

hedef = 23

ÇIKTI

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i;        // bulundu: indis  
5         }  
6     }  
7     return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```



i = 0: dizi[0] = 42 ≠ 23.

Adım 3 / 16

ÇAĞRI YIĞINI

dogrusalAra(dizi, 23)

main()

DEĞİŞKENLER

hedef = 23

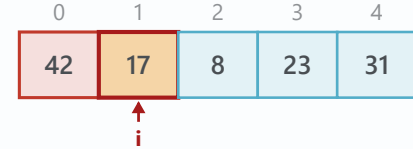
i = 0

dizi[i] = 42

ÇIKTI

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
▶   for (int i = 0; i < dizi.length; i++) {  
▶       if (dizi[i] == hedef) {  
4           return i;        // bulundu: indis  
5       }  
6   }  
7   return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```



i = 1: dizi[1] = 17 ≠ 23.

Adım 4 / 16

ÇAĞRI YIĞINI

dogrusalAra(dizi, 23)

main()

DEĞİŞKENLER

hedef = 23

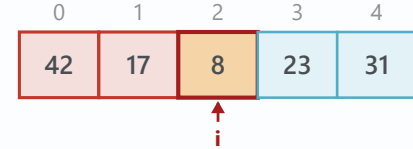
i = 1

dizi[i] = 17

ÇIKTI

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
▶   for (int i = 0; i < dizi.length; i++) {  
▶     if (dizi[i] == hedef) {  
4         return i;        // bulundu: indis  
5     }  
6 }  
7     return -1;           // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```



i = 2: dizi[2] = 8 ≠ 23.

Adım 5 / 16

ÇAĞRI YIĞINI

dogrusalAra(dizi, 23)

main()

DEĞİŞKENLER

hedef = 23

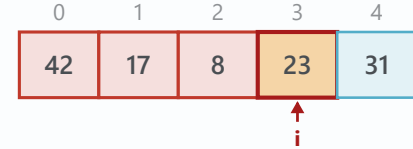
i = 2

dizi[i] = 8

ÇIKTI

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
▶   for (int i = 0; i < dizi.length; i++) {  
▶     if (dizi[i] == hedef) {  
4         return i;        // bulundu: indis  
5     }  
6 }  
7     return -1;           // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```



i = 3: dizi[3] = 23 == 23.

Adım 6 / 16

ÇAĞRI YİĞİNİ

dogrusalAra(dizi, 23)

main()

DEĞİŞKENLER

hedef = 23

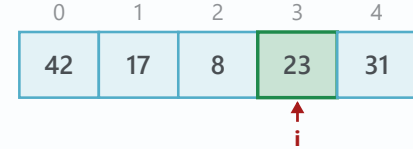
i = 3

dizi[i] = 23

ÇIKTI

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i; // bulundu: indis  
5         }  
6     }  
7     return -1; // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```



Eşleşme: `return 3` ; döngü ve metot hemen biter.

Adım 7 / 16

ÇAĞRI YİĞİNİ

`dogrusalAra(dizi, 23)`

`main()`

DEĞİŞKENLER

`hedef = 23`

`i = 3`

`dizi[i] = 23`

ÇIKTI

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i;        // bulundu: indis  
5         }  
6     }  
7     return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     ▶ System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```

0	1	2	3	4
42	17	8	23	31

Dönen 3 yazdırılır: 23, 3 numaralı indiste (dördüncü öge).

Adım 8 / 16

ÇAĞRI YİĞİNİ

main()

DEĞİŞKENLER

—

ÇIKTI

3

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {
2     for (int i = 0; i < dizi.length; i++) {
3         if (dizi[i] == hedef) {
4             return i;        // bulundu: indis
5         }
6     }
7     return -1;              // bulunamadı
8 }
9
10 public static void main(String[] args) {
11     int[] dizi = {42, 17, 8, 23, 31};
12     System.out.println(dogrusalAra(dizi, 23));
13     System.out.println(dogrusalAra(dizi, 99));
14 }
```

0	1	2	3	4
42	17	8	23	31

Bu kez listede olmayan 99 aranıyor.

Adım 9 / 16

ÇAĞRI YİĞİNİ

dogrusalAra(dizi, 99)

main()

DEĞİŞKENLER

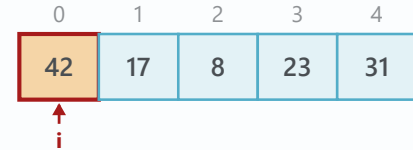
hedef = 99

ÇIKTI

3

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i;        // bulundu: indis  
5         }  
6     }  
7     return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```



i = 0: 42 ≠ 99.

Adım 10 / 16

ÇAĞRI YIĞINI

dogrusalAra(dizi, 99)

main()

DEĞİŞKENLER

hedef = 99

i = 0

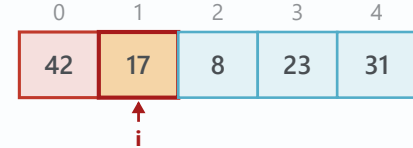
dizi[i] = 42

ÇIKTI

3

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i;        // bulundu: indis  
5         }  
6     }  
7     return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```



i = 1: 17 ≠ 99.

Adım 11 / 16

ÇAĞRI YIĞINI

dogrusalAra(dizi, 99)

main()

DEĞİŞKENLER

hedef = 99

i = 1

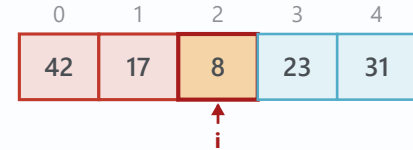
dizi[i] = 17

ÇIKTI

3

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i;        // bulundu: indis  
5         }  
6     }  
7     return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```



i = 2: 8 ≠ 99.

Adım 12 / 16

ÇAĞRI YIĞINI

dogrusalAra(dizi, 99)

main()

DEĞİŞKENLER

hedef = 99

i = 2

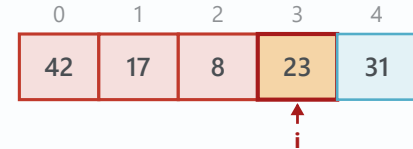
dizi[i] = 8

ÇIKTI

3

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i;        // bulundu: indis  
5         }  
6     }  
7     return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```



i = 3: 23 ≠ 99.

Adım 13 / 16

ÇAĞRI YİĞİNİ

dogrusalAra(dizi, 99)

main()

DEĞİŞKENLER

hedef = 99

i = 3

dizi[i] = 23

ÇIKTI

3

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i;        // bulundu: indis  
5         }  
6     }  
7     return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```

0	1	2	3	4
42	17	8	23	31
				↑ i

i = 4: 31 ≠ 99.

Adım 14 / 16

ÇAĞRI YİĞİNİ

dogrusalAra(dizi, 99)

main()

DEĞİŞKENLER

hedef = 99

i = 4

dizi[i] = 31

ÇIKTI

3

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i;        // bulundu: indis  
5         }  
6     }  
7     return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```

0	1	2	3	4
42	17	8	23	31

i = 5 olunca `5 < 5` yanlış; döngü biter, `-1` döndürülür (5 karşılaştırma).

Adım 15 / 16

ÇAĞRI YIĞINI

`dogrusalAra(dizi, 99)`

`main()`

DEĞİŞKENLER

`hedef = 99`

`i = 5`

ÇIKTI

3

Doğrusal Arama: Java Kodu

```
1 static int dogrusalAra(int[] dizi, int hedef) {  
2     for (int i = 0; i < dizi.length; i++) {  
3         if (dizi[i] == hedef) {  
4             return i;        // bulundu: indis  
5         }  
6     }  
7     return -1;                // bulunamadı  
8 }  
9  
10 public static void main(String[] args) {  
11     int[] dizi = {42, 17, 8, 23, 31};  
12     System.out.println(dogrusalAra(dizi, 23));  
13     System.out.println(dogrusalAra(dizi, 99));  
14 }
```

0	1	2	3	4
42	17	8	23	31

-1 "bulunamadı" anlamına gelir.

Adım 16 / 16

ÇAĞRI YİĞİNİ

main()

DEĞİŞKENLER

—

ÇIKTI

3

-1

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 seçin ve hedefle karşılaştırın.
2. Hedef bulunursa işlemi sonlandırın.
3. Hedef ortadakinden küçükse sol yarıya odaklanın.
4. Hedef ortadakinden büyükse sağ yarıya odaklanın.
5. Hedef bulunana veya arama alanı tükenene kadar tekrarlayın.

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 verir (hata vermeden!).

İkili Arama: Adım Adım

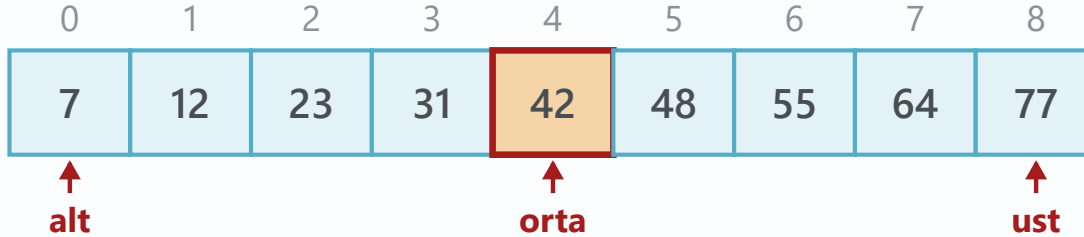
0	1	2	3	4	5	6	7	8
7	12	23	31	42	48	55	64	77
↑ alt								↑ üst

aranan öge: 48

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

Adım 1 / 6

İkili Arama: Adım Adım

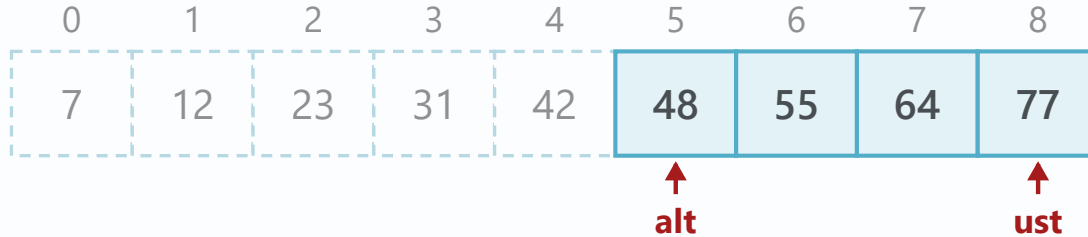


$$\text{orta} = (0 + 8) / 2 = 4 \rightarrow 42$$

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

Adım 2 / 6

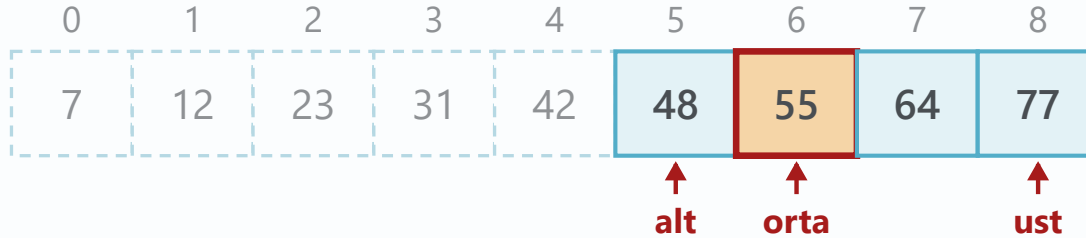
İkili Arama: Adım Adım



$48 > 42 \rightarrow$ sağ yarı

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

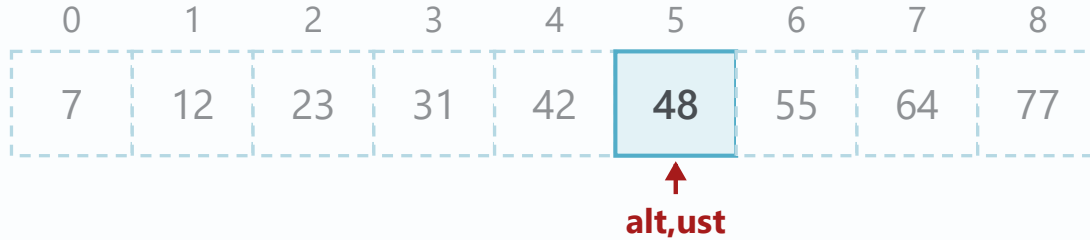
İkili Arama: Adım Adım



$$\text{orta} = (5 + 8) / 2 = 6 \rightarrow 55$$

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

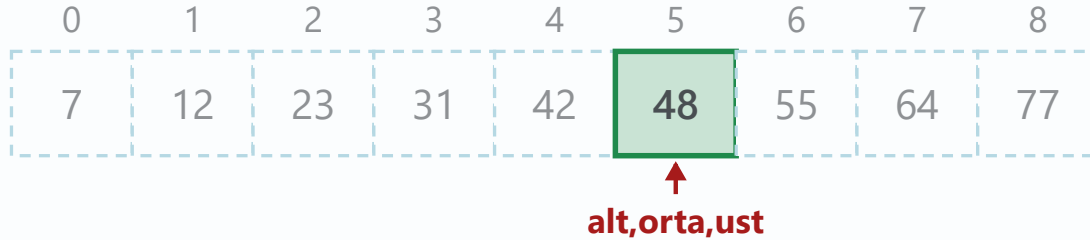
İkili Arama: Adım Adım



$48 < 55 \rightarrow$ 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 öge 48, aranan öğeye eşittir: **bulundu**. 9 öğeli dizide yalnızca **3 karşılaştırma** yapıldı.

0	1	2	3	4	5	6	7	8
7	12	23	31	42	48	55	64	77

↑ ↑
alt ust

Adım 1 / 15

```
ikiliAra(dizi, 48)
```

```
main()
```

hedef = 48

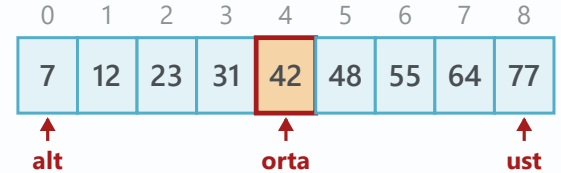
```
alt = 0
```

ust = 8

orta = -

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```



$alt \leq \text{üst} (0 \leq 8): orta = (0 + 8) / 2 = 4 \rightarrow dizi[4] = 42.$

Adım 2 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 48)

main()

DEĞİŞKENLER

hedef = 48

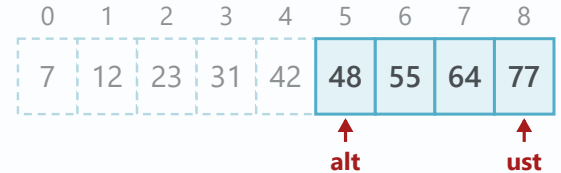
alt = 0

ust = 8

orta = 4

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            ▶ alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```



48 > 42: sağ yarıya geç, alt = orta + 1 = 5.

Adım 3 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 48)

main()

DEĞİŞKENLER

hedef = 48

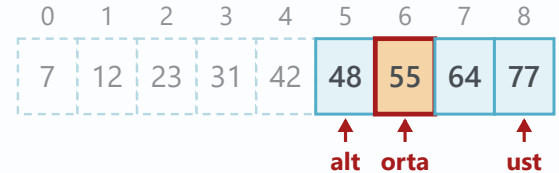
alt = 5

ust = 8

orta = 4

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```



$alt \leq \text{üst}$ ($5 \leq 8$): $orta = (5 + 8) / 2 = 6 \rightarrow dizi[6] = 55$.

Adım 4 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 48)

main()

DEĞİŞKENLER

hedef = 48

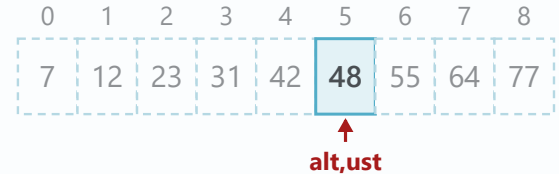
alt = 5

ust = 8

orta = 6

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```



48 < 55: sol yarıya geç, üst = orta - 1 = 5.

Adım 5 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 48)

main()

DEĞİŞKENLER

hedef = 48

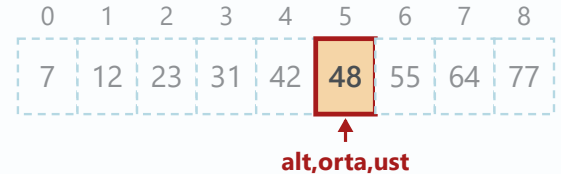
alt = 5

ust = 5

orta = 6

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```



$alt \leq \text{üst}$ ($5 \leq 5$): $\text{orta} = (5 + 5) / 2 = 5 \rightarrow \text{dizi}[5] = 48$.

Adım 6 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 48)

main()

DEĞİŞKENLER

hedef = 48

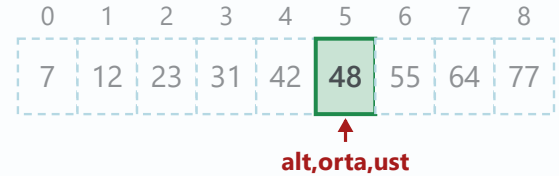
alt = 5

ust = 5

orta = 5

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```



48 == 48: **bulundu**, return 5 .

Adım 7 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 48)

main()

DEĞİŞKENLER

hedef = 48

alt = 5

ust = 5

orta = 5

0	1	2	3	4	5	6	7	8
7	12	23	31	42	48	55	64	77
↑ alt								↑ ust

Adım 8 / 15

```
ikiliAra(dizi, 50)
```

```
main()
```

hedef = 50

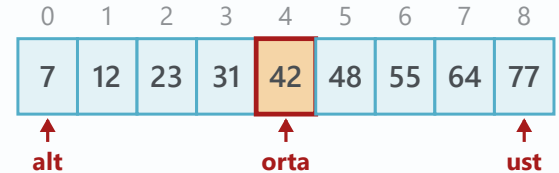
alt = 0

ust = 8

orta = -

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```



$alt \leq \text{üst}$ ($0 \leq 8$): $orta = (0 + 8) / 2 = 4 \rightarrow dizi[4] = 42$.

Adım 9 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 50)

main()

DEĞİŞKENLER

hedef = 50

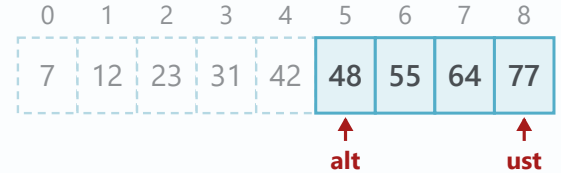
alt = 0

ust = 8

orta = 4

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```



50 > 42: sağ yarıya geç, alt = orta + 1 = 5.

Adım 10 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 50)

main()

DEĞİŞKENLER

hedef = 50

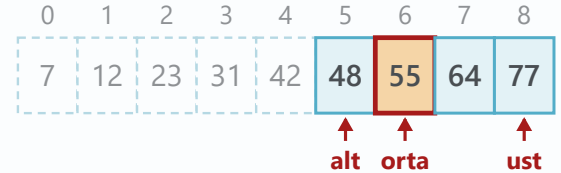
alt = 5

ust = 8

orta = 4

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```



$alt \leq \text{üst}$ ($5 \leq 8$): $orta = (5 + 8) / 2 = 6 \rightarrow dizi[6] = 55$.

Adım 11 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 50)

main()

DEĞİŞKENLER

hedef = 50

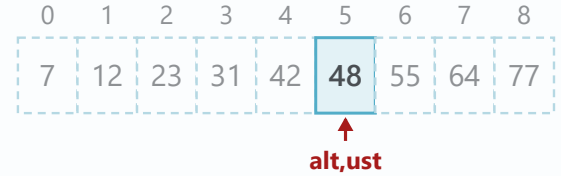
alt = 5

ust = 8

orta = 6

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```



50 < 55: sol yarıya geç, üst = orta - 1 = 5.

Adım 12 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 50)

main()

DEĞİŞKENLER

hedef = 50

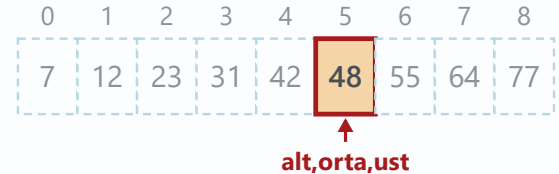
alt = 5

ust = 5

orta = 6

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```



$alt \leq \text{üst}$ ($5 \leq 5$): $orta = (5 + 5) / 2 = 5 \rightarrow dizi[5] = 48$.

Adım 13 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 50)

main()

DEĞİŞKENLER

hedef = 50

alt = 5

ust = 5

orta = 5

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta; // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1; // sol yarı  
9         } else {  
10            ▶ alt = orta + 1; // sağ yarı  
11        }  
12    }  
13    return -1; // alan tükendi  
14 }
```

0	1	2	3	4	5	6	7	8
7	12	23	31	42	48	55	64	77

↑ ↑
ust alt

50 > 48: sağ yarıya geç, alt = orta + 1 = 6.

Adım 14 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 50)

main()

DEĞİŞKENLER

hedef = 50

alt = 6

ust = 5

orta = 5

İkili Arama: Java Kodu

```
1 static int ikiliAra(int[] dizi, int hedef) {  
2     int alt = 0, ust = dizi.length - 1;  
3     while (alt <= ust) {  
4         int orta = (alt + ust) / 2;  
5         if (dizi[orta] == hedef) {  
6             return orta;           // bulundu  
7         } else if (hedef < dizi[orta]) {  
8             ust = orta - 1;         // sol yarı  
9         } else {  
10            alt = orta + 1;         // sağ yarı  
11        }  
12    }  
13    return -1;                     // alan tükendi  
14 }
```

0	1	2	3	4	5	6	7	8
7	12	23	31	42	48	55	64	77

↑ ↑
ust **alt**

alt > üst (6 > 5): arama alanı boş, **-1**
döndürülür.

Adım 15 / 15

ÇAĞRI YIĞINI

ikiliAra(dizi, 50)

main()

DEĞİŞKENLER

hedef = 50

alt = 6

ust = 5

orta = -

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

Öğesi 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.

Deneyin: Doğrusal ve İkili Arama

Aranan

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

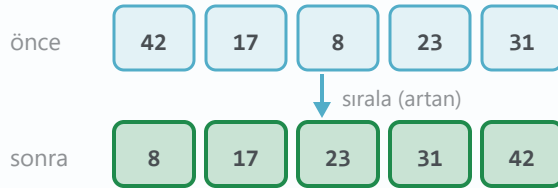
Sıralı bir dizi hazır. Bir sayı yazıp yöntem seçin.

Dizi sıralıdır, bu yüzden iki yöntem de doğru sonuç verir. Turuncu: incelenen öğe, kırmızı: elenen öğeler, yeşil: bulunan öğe. Karşılaştırma sayılarını farklı hedeflerle (ilk, son, olmayan bir sayı) karşılaştırın.

SIRALAMA ALGORİTMALARI

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.



Daha hızlı yöntemler (birleştirmeli, hızlı sıralama) ve Java'nın hazır `Arrays.sort` metodu ilerideki bölümlerde ele alınacak.

Rastgele (Random)

Öğeleri rastgele karıştırır; sıralamayı garanti etmez.

Seçimli (Selection Sort)

En küçüğü seçip başa koyar.

Kabarcık (Bubble Sort)

Komşuları karşılaştırıp büyüğü sona taşır.

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 hazırdir:
`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

0	1	2	3	4
42	17	8	23	31

başlangıç dizisi

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

Adım 1 / 4

Rastgele Sıralama: Adım Adım

0	1	2	3	4
31	17	42	8	23

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

0	1	2	3	4
8	23	31	42	17

2. karıştırma

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

Adım 3 / 4

Rastgele Sıralama: Adım Adım

0	1	2	3	4
8	23	17	31	42

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.

Karıştırma: Java Kodu (Fisher-Yates)

```
import java.util.Arrays;
import java.util.Random;

public class RastgeleSiralama {
    public static void main(String[] args) {
        int[] dizi = {42, 17, 8, 23, 31};
        Random rastgele = new Random(3);
        for (int tur = 1; tur <= 3; tur++) {
            // Fisher-Yates: sondan başa karıştır
            for (int i = dizi.length - 1; i > 0; i--) {
                int j = rastgele.nextInt(i + 1); // 0..i
                int gecici = dizi[i];           // takas
                dizi[i] = dizi[j];
                dizi[j] = gecici;
            }
            System.out.println(Arrays.toString(dizi));
        }
    }
}
```

```
[23, 17, 42, 8, 31]
[31, 17, 23, 42, 8]
[17, 23, 31, 42, 8]
```

Fisher-Yates yöntemi sondan başa giderek her öğeyi kendisi veya solundaki rastgele bir öğeyle değiştirir; her diziliş eşit olasılıkla çıkar. **gecici** değişkeni iki değeri **yer değiştirmek** (takas, swap) için gereklidir.

Karıştırma (Fisher–Yates): Kod İzleme

```
► int[] dizi = {42, 17, 8, 23, 31};  
► Random rastgele = new Random(3);  
3 for (int i = dizi.length - 1; i > 0; i--) {  
4     int j = rastgele.nextInt(i + 1);    // 0..i  
5     int gecici = dizi[i];                // takas  
6     dizi[i] = dizi[j];  
7     dizi[j] = gecici;  
8 }  
9 System.out.println(Arrays.toString(dizi));
```

0	1	2	3	4
42	17	8	23	31

Dizi ve sabit tohumlu (3) bir **Random** hazırlanır; tohum sabit olduğu için çekilen sayılar her çalıştırmada aynıdır.

Adım 1 / 9

DEĞİŞKENLER

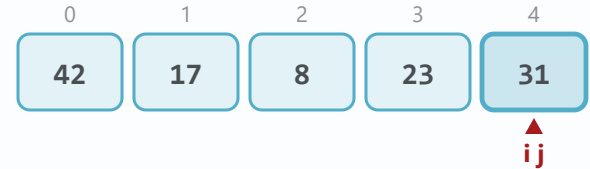
dizi = [42, 17, 8, 23, 31]

ÇIKTI



Karıştırma (Fisher–Yates): Kod İzleme

```
1 int[] dizi = {42, 17, 8, 23, 31};
2 Random rastgele = new Random(3);
▶ for (int i = dizi.length - 1; i > 0; i--) {
▶   int j = rastgele.nextInt(i + 1); // 0..i
5   int gecici = dizi[i];           // takas
6   dizi[i] = dizi[j];
7   dizi[j] = gecici;
8 }
9 System.out.println(Arrays.toString(dizi));
```



nextInt(5) → 4

i = 4 (son konum): 0..4 arasından **j**
= 4 çekildi.

Adım 2 / 9

DEĞİŞKENLER

i = 4

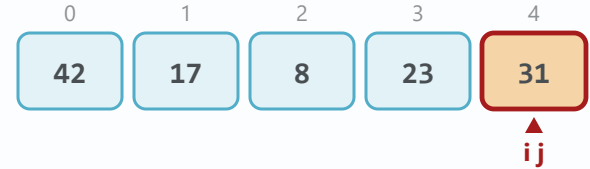
j = 4

ÇIKTI



Karıştırma (Fisher–Yates): Kod İzleme

```
1 int[] dizi = {42, 17, 8, 23, 31};
2 Random rastgele = new Random(3);
3 for (int i = dizi.length - 1; i > 0; i--) {
4     int j = rastgele.nextInt(i + 1);    // 0..i
5     ▶ int gecici = dizi[i];              // takas
6     ▶ dizi[i] = dizi[j];
7     ▶ dizi[j] = gecici;
8 }
9 System.out.println(Arrays.toString(dizi));
```



Öge kendisiyle takas edilir; 31 son konumda **kesinleşir**.

Adım 3 / 9

DEĞİŞKENLER

dizi = [42, 17, 8, 23, 31]

i = 4

j = 4

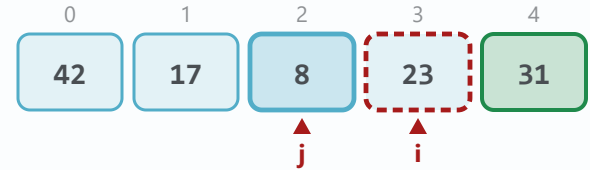
gecici = 31

ÇIKTI



Karıştırma (Fisher–Yates): Kod İzleme

```
1 int[] dizi = {42, 17, 8, 23, 31};
2 Random rastgele = new Random(3);
3 for (int i = dizi.length - 1; i > 0; i--) {
4     ▶ int j = rastgele.nextInt(i + 1); // 0..i
5     int gecici = dizi[i];           // takas
6     dizi[i] = dizi[j];
7     dizi[j] = gecici;
8 }
9 System.out.println(Arrays.toString(dizi));
```



nextInt(4) → 2

i = 3: 0..3 arasından j = 2.

Adım 4 / 9

DEĞİŞKENLER

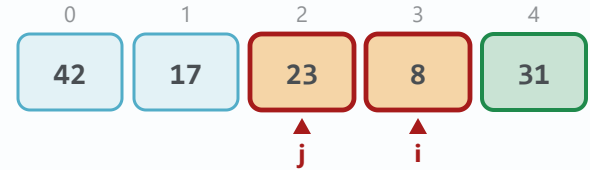
i = 3

j = 2

ÇIKTI

Karıştırma (Fisher–Yates): Kod İzleme

```
1 int[] dizi = {42, 17, 8, 23, 31};
2 Random rastgele = new Random(3);
3 for (int i = dizi.length - 1; i > 0; i--) {
4     int j = rastgele.nextInt(i + 1); // 0..i
5     ▶ int gecici = dizi[i];           // takas
6     ▶ dizi[i] = dizi[j];
7     ▶ dizi[j] = gecici;
8 }
9 System.out.println(Arrays.toString(dizi));
```



3. ve 2. öge (23 ile 8) yer değiştirir.

Adım 5 / 9

DEĞİŞKENLER

dizi = [42, 17, 23, 8, 31]

i = 3

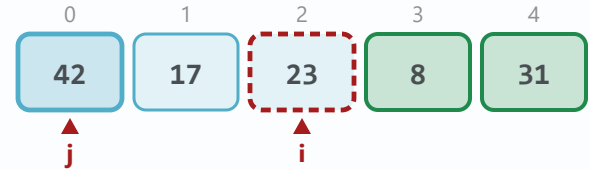
j = 2

gecici = 23

ÇIKTI

Karıştırma (Fisher–Yates): Kod İzleme

```
1 int[] dizi = {42, 17, 8, 23, 31};
2 Random rastgele = new Random(3);
3 for (int i = dizi.length - 1; i > 0; i--) {
4     ▶ int j = rastgele.nextInt(i + 1); // 0..i
5     int gecici = dizi[i];           // takas
6     dizi[i] = dizi[j];
7     dizi[j] = gecici;
8 }
9 System.out.println(Arrays.toString(dizi));
```



nextInt(3) → 0

i = 2: 0..2 arasından j = 0.

Adım 6 / 9

DEĞİŞKENLER

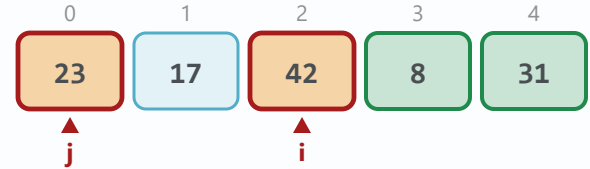
i = 2

j = 0

ÇIKTI

Karıştırma (Fisher–Yates): Kod İzleme

```
1 int[] dizi = {42, 17, 8, 23, 31};
2 Random rastgele = new Random(3);
3 for (int i = dizi.length - 1; i > 0; i--) {
4     int j = rastgele.nextInt(i + 1); // 0..i
5     ▶ int gecici = dizi[i];           // takas
6     ▶ dizi[i] = dizi[j];
7     ▶ dizi[j] = gecici;
8 }
9 System.out.println(Arrays.toString(dizi));
```



2. ve 0. öge (23 ile 42) yer değiştirir.

Adım 7 / 9

DEĞİŞKENLER

dizi = [23, 17, 42, 8, 31]

i = 2

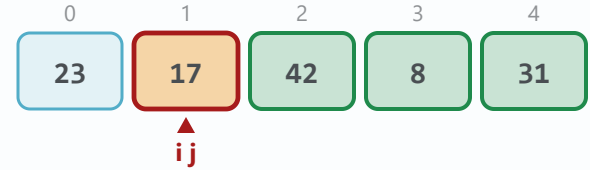
j = 0

gecici = 23

ÇIKTI

Karıştırma (Fisher–Yates): Kod İzleme

```
1 int[] dizi = {42, 17, 8, 23, 31};
2 Random rastgele = new Random(3);
3 for (int i = dizi.length - 1; i > 0; i--) {
4     ▶ int j = rastgele.nextInt(i + 1); // 0..i
5     ▶ int gecici = dizi[i];           // takas
6     ▶ dizi[i] = dizi[j];
7     ▶ dizi[j] = gecici;
8 }
9 System.out.println(Arrays.toString(dizi));
```



nextInt(2) → 1

i = 1: **j** = 1, kendisiyle takas. **i** 0 olunca döngü biter (4 takas).

Adım 8 / 9

DEĞİŞKENLER

dizi = [23, 17, 42, 8, 31]

i = 1

j = 1

gecici = 17

ÇIKTI



Karıştırma (Fisher–Yates): Kod İzleme

```
1 int[] dizi = {42, 17, 8, 23, 31};
2 Random rastgele = new Random(3);
3 for (int i = dizi.length - 1; i > 0; i--) {
4     int j = rastgele.nextInt(i + 1); // 0..i
5     int gecici = dizi[i];             // takas
6     dizi[i] = dizi[j];
7     dizi[j] = gecici;
8 }
▶ System.out.println(Arrays.toString(dizi));
```



Karıştırılmış dizi yazılır. Her konum için yalnız kendisi ve solundakiler arasından seçim yapıldığından $5!$ dizilişin her biri eşit olasılıklıdır.

Adım 9 / 9

DEĞİŞKENLER

dizi = [23, 17, 42, 8, 31]

ÇIKTI

[23, 17, 42, 8, 31]

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 bulun.
2. Bu öğeyi koleksiyonun başına taşıyın (ilk öğeyle yer değiştirin).
3. Geri kalan kısımdaki en küçük öğeyi bulun.
4. Onu ikinci pozisyona taşıyın.
5. Bu adımları koleksiyonun sonuna kadar tekrarlayın.

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

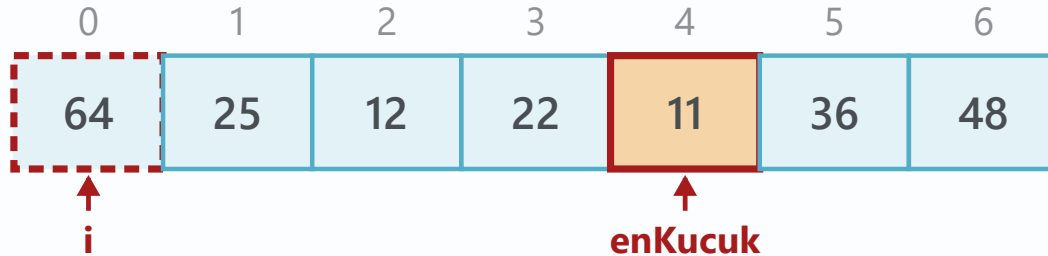
0	1	2	3	4	5	6
64	25	12	22	11	36	48

başlangıç

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

Adım 1 / 13

Seçimli Sıralama: Adım Adım



sıralanmamış kısmın en küçüğü: 11

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

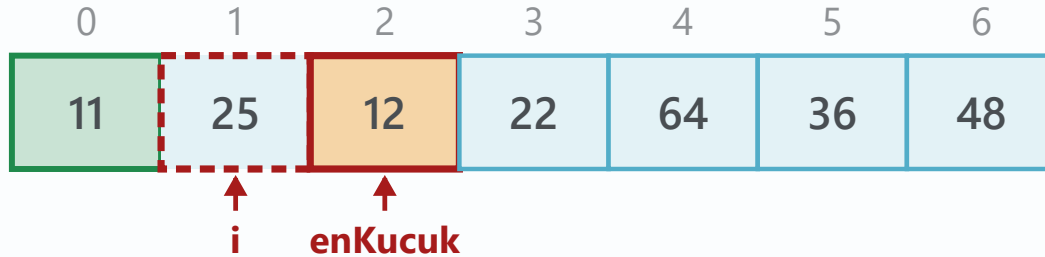
Seçimli Sıralama: Adım Adım

0	1	2	3	4	5	6
11	25	12	22	64	36	48

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. pozisyonadaki 25 ile yer değiştirecek.

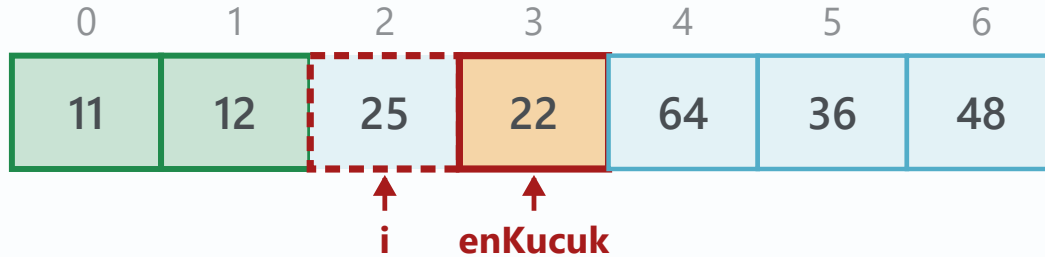
Seçimli Sıralama: Adım Adım

0	1	2	3	4	5	6
11	12	25	22	64	36	48

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. pozisyonadaki 25 ile yer değiştirecek.

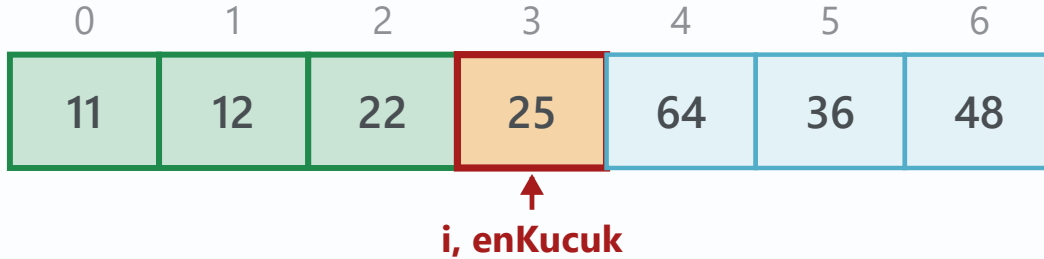
Seçimli Sıralama: Adım Adım

0	1	2	3	4	5	6
11	12	22	25	64	36	48

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.

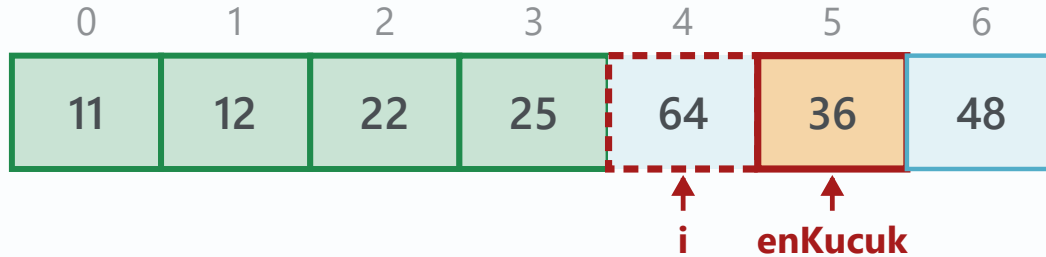
Seçimli Sıralama: Adım Adım

0	1	2	3	4	5	6
11	12	22	25	64	36	48

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. pozisyonadaki 64 ile yer değiştirecek.

Seçimli Sıralama: Adım Adım

0	1	2	3	4	5	6
11	12	22	25	36	64	48

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. pozisyonadaki 64 ile yer değiştirecek.

Seçimli Sıralama: Adım Adım

0	1	2	3	4	5	6
11	12	22	25	36	48	64

6. adımdan sonra

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

Seçimli Sıralama: Java Kodu

```
import java.util.Arrays;

public class SecmeliSiralama {
    static void secmeliSiralama(int[] dizi) {
        int n = dizi.length;
        for (int i = 0; i < n - 1; i++) {
            int enKucuk = i;           // en küçüğün indisi
            for (int j = i + 1; j < n; j++) {
                if (dizi[j] < dizi[enKucuk]) {
                    enKucuk = j;
                }
            }
            int gecici = dizi[i];       // yer değiştir
            dizi[i] = dizi[enKucuk];
            dizi[enKucuk] = gecici;
            System.out.println(Arrays.toString(dizi));
        }
    }

    public static void main(String[] args) {
        int[] dizi = {64, 25, 12, 22, 11, 36, 48};
        secmeliSiralama(dizi);
    }
}
```

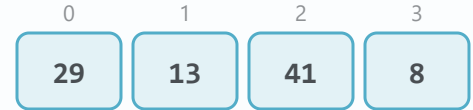
```
[11, 25, 12, 22, 64, 36, 48]
[11, 12, 25, 22, 64, 36, 48]
[11, 12, 22, 25, 64, 36, 48]
[11, 12, 22, 25, 64, 36, 48]
[11, 12, 22, 25, 36, 64, 48]
[11, 12, 22, 25, 36, 48, 64]
```

Her satır bir adımın sonucudur. Dış döngü yerleştirilecek konumu (**i**), iç döngü en küçüğün indisini (**enKucuk**) bulur.

Arrays.toString diziyi metne çevirir.

Seçimli Sıralama: Kod İzleme

```
1 static void secmeliSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int i = 0; i < n - 1; i++) {  
4         int enKucuk = i;           // aday indis  
5         for (int j = i + 1; j < n; j++) {  
6             if (dizi[j] < dizi[enKucuk]) {  
7                 enKucuk = j;  
8             }  
9         }  
10        int gecici = dizi[i];       // takas  
11        dizi[i] = dizi[enKucuk];  
12        dizi[enKucuk] = gecici;  
13    }  
14 }
```



Sıralanacak dizi [29, 13, 41, 8] ,
n = 4. Hiçbir öge henüz yerinde değil.

Adım 1 / 11

DEĞİŞKENLER

dizi = [29, 13, 41, 8]

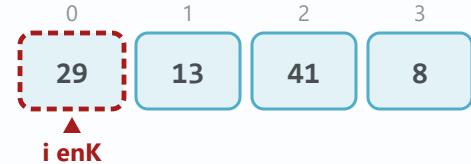
n = 4

ÇIKTI



Seçimli Sıralama: Kod İzleme

```
1 static void secmeliSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int i = 0; i < n - 1; i++) {  
4         int enKucuk = i;           // aday indis  
5         for (int j = i + 1; j < n; j++) {  
6             if (dizi[j] < dizi[enKucuk]) {  
7                 enKucuk = j;  
8             }  
9         }  
10        int gecici = dizi[i];      // takas  
11        dizi[i] = dizi[enKucuk];  
12        dizi[enKucuk] = gecici;  
13    }  
14 }
```



i = 0: 0. konuma gelecek en küçüğü arıyoruz. Aday: **enKucuk** = 0 (29).

Adım 2 / 11

DEĞİŞKENLER

n = 4

i = 0

enKucuk = 0

ÇIKTI



Seçimli Sıralama: Kod İzleme

```
1 static void secmeliSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int i = 0; i < n - 1; i++) {  
4         int enKucuk = i;           // aday indis  
5         for (int j = i + 1; j < n; j++) {  
6             if (dizi[j] < dizi[enKucuk]) {  
7                 enKucuk = j;  
8             }  
9         }  
10        int gecici = dizi[i];      // takas  
11        dizi[i] = dizi[enKucuk];  
12        dizi[enKucuk] = gecici;  
13    }  
14 }
```



13 < 29 → enKucuk = 1

j = 1: 13 < 29 → yeni aday enKucuk = 1.

Adım 3 / 11

DEĞİŞKENLER

n = 4

i = 0

j = 1

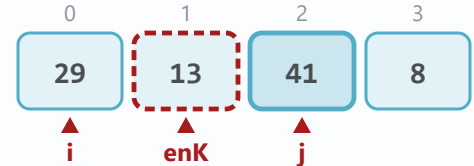
enKucuk = 1

ÇIKTI



Seçimli Sıralama: Kod İzleme

```
1 static void secmeliSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int i = 0; i < n - 1; i++) {  
4         int enKucuk = i;           // aday indis  
5         for (int j = i + 1; j < n; j++) {  
6             if (dizi[j] < dizi[enKucuk]) {  
7                 enKucuk = j;  
8             }  
9         }  
10        int gecici = dizi[i];      // takas  
11        dizi[i] = dizi[enKucuk];  
12        dizi[enKucuk] = gecici;  
13    }  
14 }
```



41 < 13 → false

j = 2: 41 < 13 yanlış; aday değişmez.

Adım 4 / 11

DEĞİŞKENLER

n = 4

i = 0

j = 2

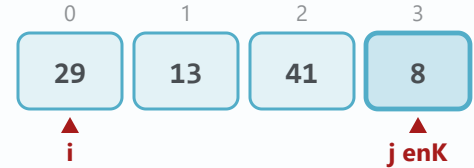
enKucuk = 1

ÇIKTI



Seçimli Sıralama: Kod İzleme

```
1 static void secmeliSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int i = 0; i < n - 1; i++) {  
4         int enKucuk = i;           // aday indis  
5         for (int j = i + 1; j < n; j++) {  
6             if (dizi[j] < dizi[enKucuk]) {  
7                 enKucuk = j;  
8             }  
9         }  
10        int gecici = dizi[i];      // takas  
11        dizi[i] = dizi[enKucuk];  
12        dizi[enKucuk] = gecici;  
13    }  
14 }
```



8 < 13 → enKucuk = 3

j = 3: 8 < 13 → enKucuk = 3. İç
döngü biter: en küçük 8.

Adım 5 / 11

DEĞİŞKENLER

n = 4

i = 0

j = 3

enKucuk = 3

ÇIKTI



Seçimli Sıralama: Kod İzleme

```
1 static void secmeliSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int i = 0; i < n - 1; i++) {  
4         int enKucuk = i;           // aday indis  
5         for (int j = i + 1; j < n; j++) {  
6             if (dizi[j] < dizi[enKucuk]) {  
7                 enKucuk = j;  
8             }  
9         }  
10        int gecici = dizi[i];      // takas  
11        dizi[i] = dizi[enKucuk];  
12        dizi[enKucuk] = gecici;  
13    }  
14 }
```



dizi[0] ↔ dizi[3]

gecici yardımıyla 0. ve 3. öge yer değiştirir. 8 artık **yerinde**.

Adım 6 / 11

DEĞİŞKENLER

dizi = [8, 13, 41, 29]

i = 0

enKucuk = 3

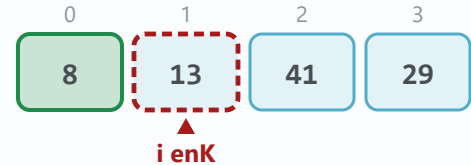
gecici = 29

ÇIKTI



Seçimli Sıralama: Kod İzleme

```
1 static void secmeliSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int i = 0; i < n - 1; i++) {  
4         int enKucuk = i;           // aday indis  
5         for (int j = i + 1; j < n; j++) {  
6             if (dizi[j] < dizi[enKucuk]) {  
7                 enKucuk = j;  
8             }  
9         }  
10        int gecici = dizi[i];      // takas  
11        dizi[i] = dizi[enKucuk];  
12        dizi[enKucuk] = gecici;  
13    }  
14 }
```



41 < 13 X, 29 < 13 X

i = 1: aday 13. 41 ve 29 ondan küçük değil → enKucuk 1 kalır.

Adım 7 / 11

DEĞİŞKENLER

n = 4

i = 1

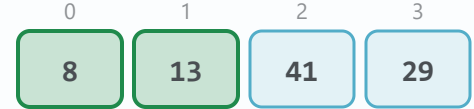
enKucuk = 1

ÇIKTI



Seçimli Sıralama: Kod İzleme

```
1 static void secmeliSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int i = 0; i < n - 1; i++) {  
4         int enKucuk = i;           // aday indis  
5         for (int j = i + 1; j < n; j++) {  
6             if (dizi[j] < dizi[enKucuk]) {  
7                 enKucuk = j;  
8             }  
9         }  
10        int gecici = dizi[i];      // takas  
11        dizi[i] = dizi[enKucuk];  
12        dizi[enKucuk] = gecici;  
13    }  
14 }
```



dizi[1] kendisiyle değişir

Öge kendisiyle yer değiştirir; dizi değişmez. 13 de yerinde.

Adım 8 / 11

DEĞİŞKENLER

dizi = [8, 13, 41, 29]

i = 1

enKucuk = 1

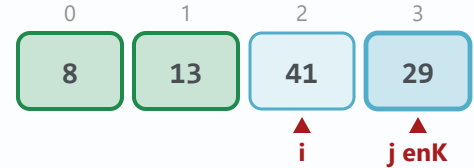
gecici = 13

ÇIKTI



Seçimli Sıralama: Kod İzleme

```
1 static void secmeliSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int i = 0; i < n - 1; i++) {  
4         int enKucuk = i;           // aday indis  
5         for (int j = i + 1; j < n; j++) {  
6             if (dizi[j] < dizi[enKucuk]) {  
7                 enKucuk = j;  
8             }  
9         }  
10        int gecici = dizi[i];      // takas  
11        dizi[i] = dizi[enKucuk];  
12        dizi[enKucuk] = gecici;  
13    }  
14 }
```



29 < 41 → enKucuk = 3

i = 2: aday 41; j = 3'te 29 < 41 →
enKucuk = 3.

Adım 9 / 11

DEĞİŞKENLER

n = 4

i = 2

j = 3

enKucuk = 3

ÇIKTI



Seçimli Sıralama: Kod İzleme

```
1 static void secmeliSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int i = 0; i < n - 1; i++) {  
4         int enKucuk = i;           // aday indis  
5         for (int j = i + 1; j < n; j++) {  
6             if (dizi[j] < dizi[enKucuk]) {  
7                 enKucuk = j;  
8             }  
9         }  
10        int gecici = dizi[i];      // takas  
11        dizi[i] = dizi[enKucuk];  
12        dizi[enKucuk] = gecici;  
13    }  
14 }
```



2. ve 3. öge yer değiştirir: [8, 13, 29, 41] .

Adım 10 / 11

DEĞİŞKENLER

dizi = [8, 13, 29, 41] i = 2

enKucuk = 3 gecici = 41

ÇIKTI



Seçimli Sıralama: Kod İzleme

```
1 static void secmeliSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int i = 0; i < n - 1; i++) {  
4         int enKucuk = i;           // aday indis  
5         for (int j = i + 1; j < n; j++) {  
6             if (dizi[j] < dizi[enKucuk]) {  
7                 enKucuk = j;  
8             }  
9         }  
10        int gecici = dizi[i];       // takas  
11        dizi[i] = dizi[enKucuk];  
12        dizi[enKucuk] = gecici;  
13    }  
14 }
```



$i = 3 < 3 \rightarrow \text{false}$

$i = 3$: $3 < 3$ yanlış, dış döngü biter.
Son öğe kendiliğinden yerindedir; dizi
sıralı. Toplam 6 karşılaştırma.

Adım 11 / 11

DEĞİŞKENLER

dizi = [8, 13, 29, 41]

n = 4

i = 3

ÇIKTI



Kabarcık Sıralama (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 başlayın.
2. İki yan yana öğeyi karşılaştırın; gerekiyorsa yer değiştirip büyük olanı sağa taşıyın.
3. Koleksiyonun sonuna kadar bu adımı tekrarlayın (bir **tur**).
4. En büyük öğe sona geldiği için son öğeyi sıralanmış kabul edin.
5. İlk dört adımı kalan kısım için tekrarlayın; bir turda hiç yer değiştirme olmazsa durun.

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 Sıralama: Adım Adım

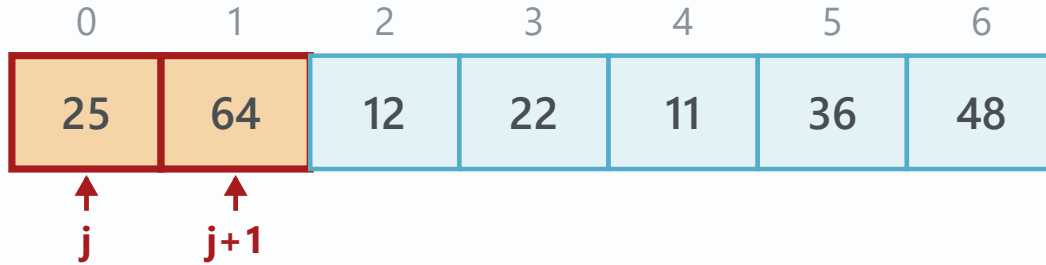
0	1	2	3	4	5	6
64	25	12	22	11	36	48

başlangıç

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

Adım 1 / 22

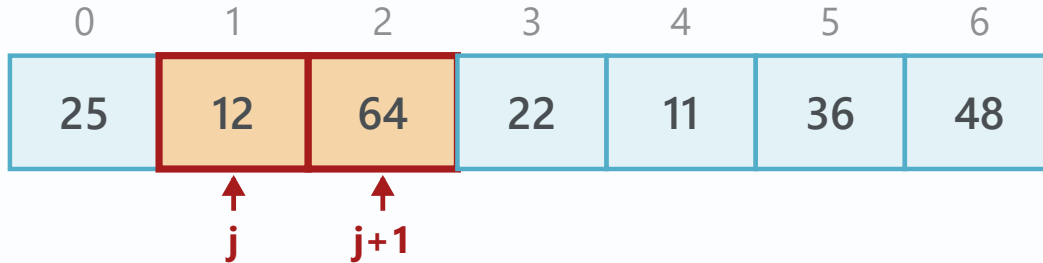
Kabarcık Sıralama: Adım Adım



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].

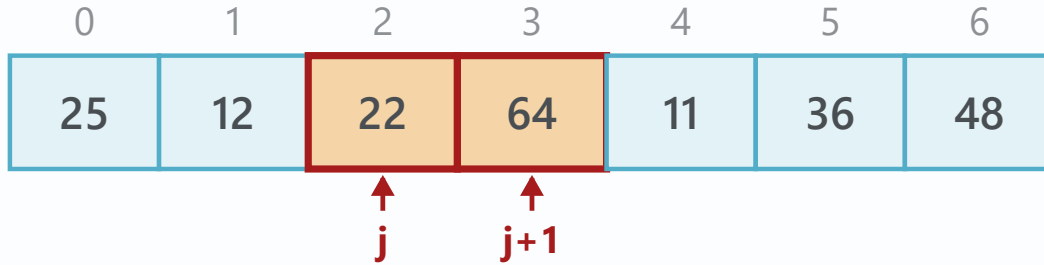
Kabarcık Sıralama: Adım Adım



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 Sıralama: Adım Adım

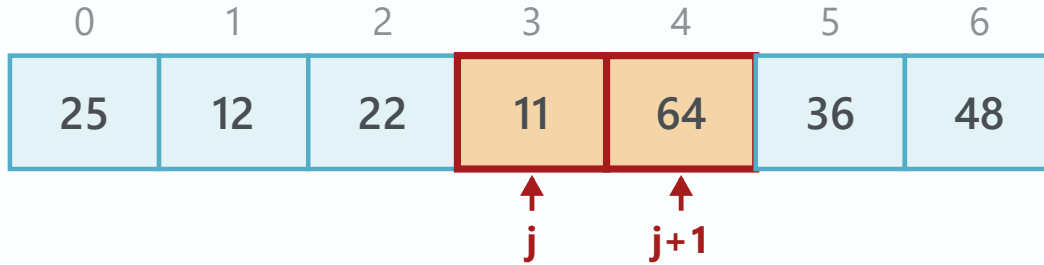


1. tur · yer değiştirdi

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

Adım 4 / 22

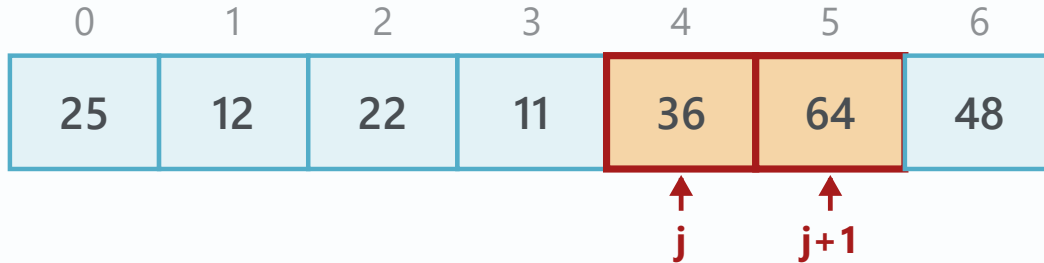
Kabarcık Sıralama: Adım Adım



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].

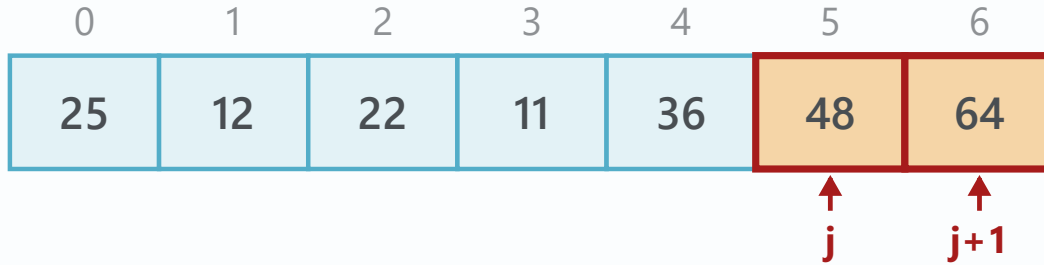
Kabarcık Sıralama: Adım Adım



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].

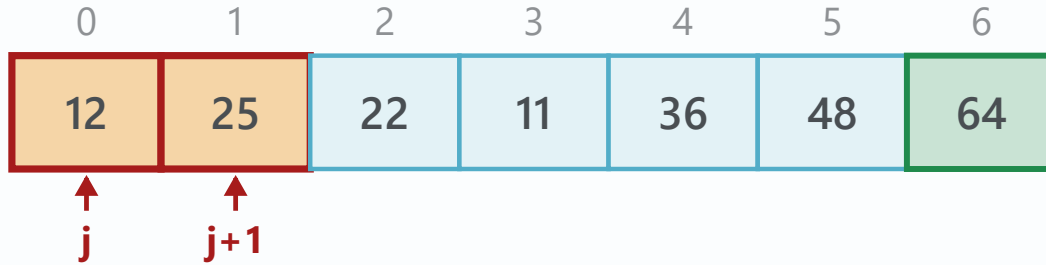
Kabarcık Sıralama: Adım Adım



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].

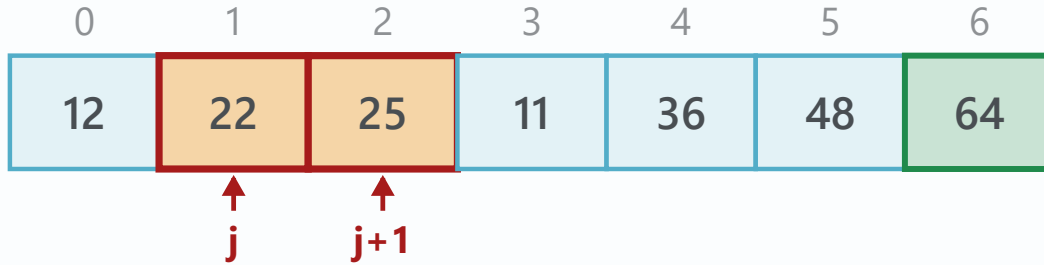
Kabarcık Sıralama: Adım Adım



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].

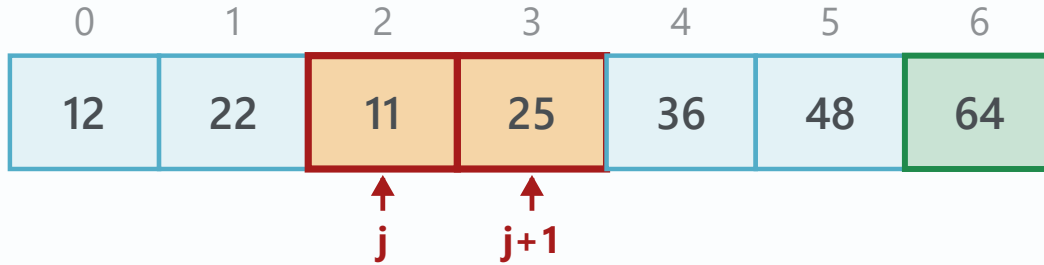
Kabarcık Sıralama: Adım Adım



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].

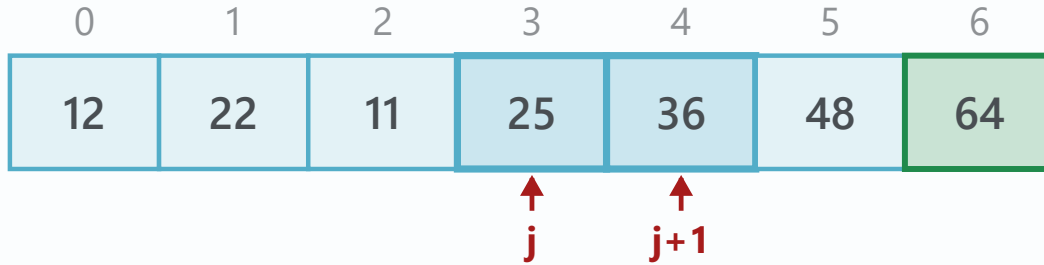
Kabarcık Sıralama: Adım Adım



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].

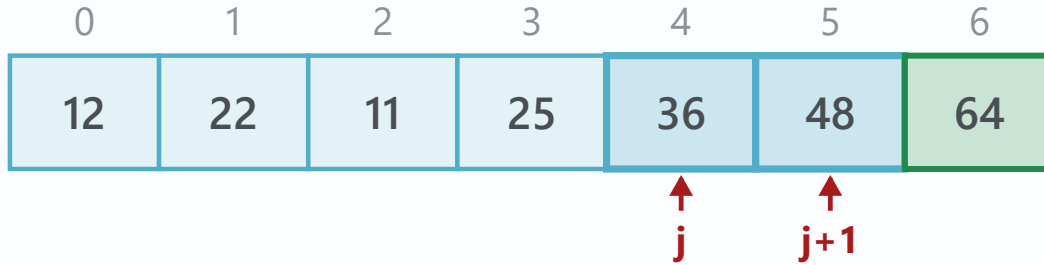
Kabarcık Sıralama: Adım Adım



2. tur · değişiklik yok

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

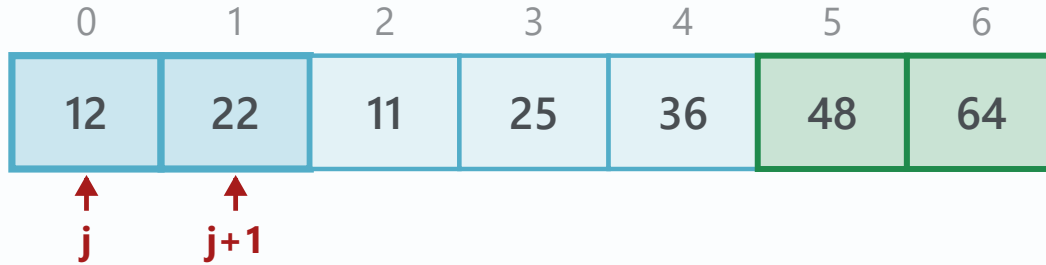
Kabarcık Sıralama: Adım Adım



2. tur · değişiklik yok

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

Kabarcık Sıralama: Adım Adım

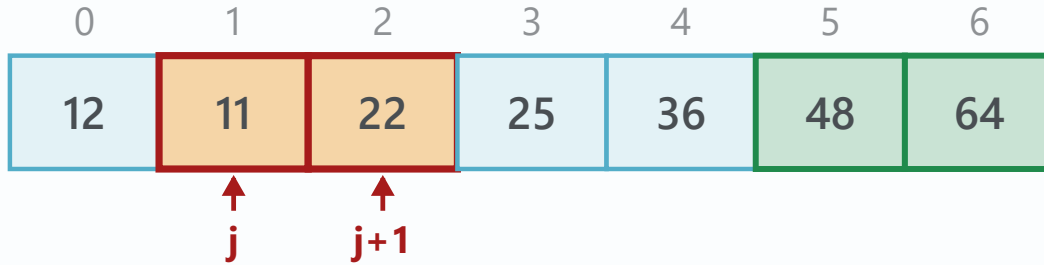


3. tur · değişiklik yok

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

Adım 13 / 22

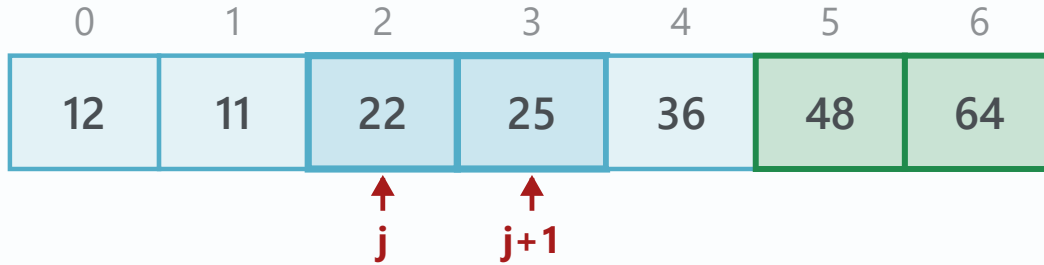
Kabarcık Sıralama: Adım Adım



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 Sıralama: Adım Adım

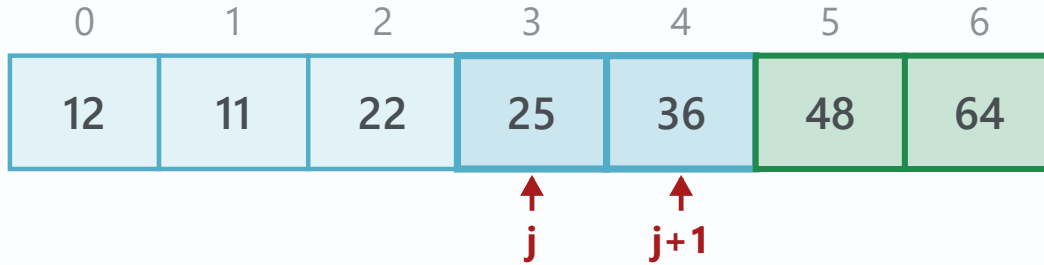


3. tur · değişiklik yok

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

Adım 15 / 22

Kabarcık Sıralama: Adım Adım

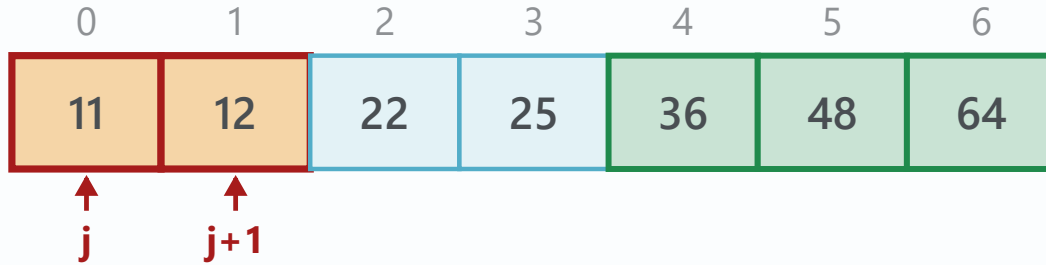


3. tur · değişiklik yok

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

Adım 16 / 22

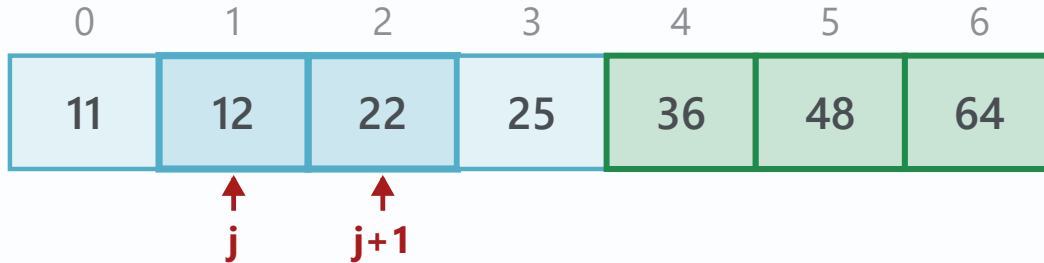
Kabarcık Sıralama: Adım Adım



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 Sıralama: Adım Adım

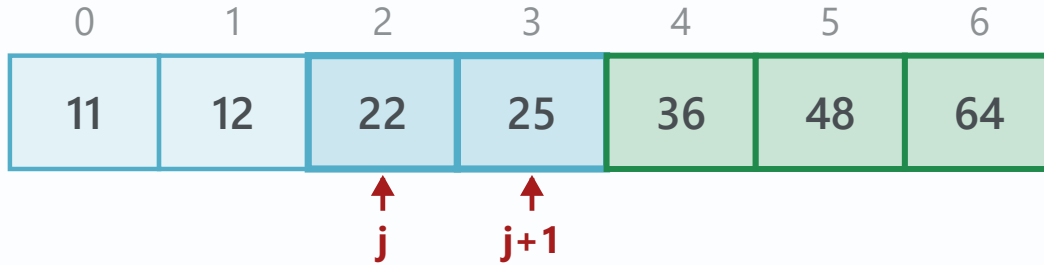


4. tur · değişiklik yok

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

Adım 18 / 22

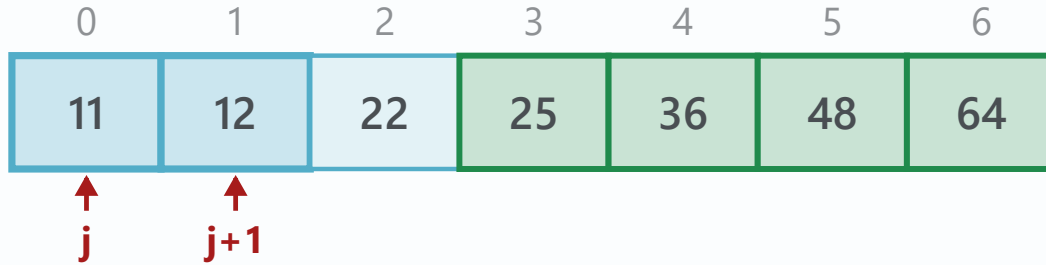
Kabarcık Sıralama: Adım Adım



4. tur · değişiklik yok

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

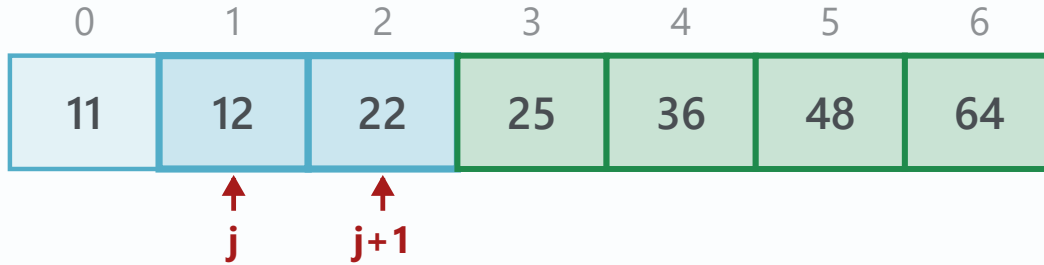
Kabarcık Sıralama: Adım Adım



5. tur · değişiklik yok

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

Kabarcık Sıralama: Adım Adım



5. tur · değişiklik yok

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

Adım 21 / 22

Kabarcık Sıralama: Adım Adım

0	1	2	3	4	5	6
11	12	22	25	36	48	64

dizi sıralı

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

Kabarcık Sıralama: Java Kodu

```
import java.util.Arrays;

public class KabarcikSiralama {
    static void kabarcikSiralama(int[] dizi) {
        int n = dizi.length;
        for (int tur = 0; tur < n - 1; tur++) {
            boolean degisti = false;
            for (int j = 0; j < n - 1 - tur; j++) {
                if (dizi[j] > dizi[j + 1]) { // ters sıra
                    int gecici = dizi[j];
                    dizi[j] = dizi[j + 1];
                    dizi[j + 1] = gecici;
                    degisti = true;
                }
            }
            System.out.println(Arrays.toString(dizi));
            if (!degisti) break; // takas yok: sıralı
        }
    }

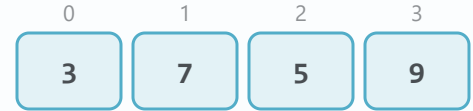
    public static void main(String[] args) {
        int[] dizi = {64, 25, 12, 22, 11, 36, 48};
        kabarcikSiralama(dizi);
    }
}
```

```
[25, 12, 22, 11, 36, 48, 64]
[12, 22, 11, 25, 36, 48, 64]
[12, 11, 22, 25, 36, 48, 64]
[11, 12, 22, 25, 36, 48, 64]
[11, 12, 22, 25, 36, 48, 64]
```

Her satır bir turun sonucudur.
Her turdan sonra en büyük
öğe sona yerleştiği için iç
döngü `n - 1 - tur` 'a
kadar gider. `degisti`
bayrağı, takas olmayan 5.
turda algoritmayı erken
durdurur.

Kabarcık Sıralama: Kod İzleme

```
1 static void kabarcikSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int tur = 0; tur < n - 1; tur++) {  
4         boolean degisti = false;  
5         for (int j = 0; j < n - 1 - tur; j++) {  
6             if (dizi[j] > dizi[j + 1]) { // ters sıra  
7                 int gecici = dizi[j];  
8                 dizi[j] = dizi[j + 1];  
9                 dizi[j + 1] = gecici;  
10                degisti = true;  
11            }  
12        }  
13        if (!degisti) break; // takas yok:  
14    }  
15 }
```



Dizi [3, 7, 5, 9], n = 4. Yalnızca 7 ile 5 ters sırada.

Adım 1 / 9

DEĞİŞKENLER

dizi = [3, 7, 5, 9]

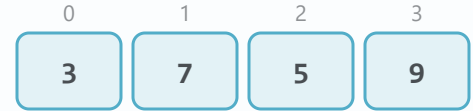
n = 4

ÇIKTI



Kabarcık Sıralama: Kod İzleme

```
1 static void kabarcikSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int tur = 0; tur < n - 1; tur++) {  
4         boolean degisti = false;  
5         for (int j = 0; j < n - 1 - tur; j++) {  
6             if (dizi[j] > dizi[j + 1]) { // ters sıra  
7                 int gecici = dizi[j];  
8                 dizi[j] = dizi[j + 1];  
9                 dizi[j + 1] = gecici;  
10                degisti = true;  
11            }  
12        }  
13        if (!degisti) break; // takas yok:  
14    }  
15 }
```



1. tur (**tur** = 0): bayrak **degisti** = false. **j** , 0'dan 2'ye kadar gidecek.

Adım 2 / 9

DEĞİŞKENLER

n = 4

tur = 0

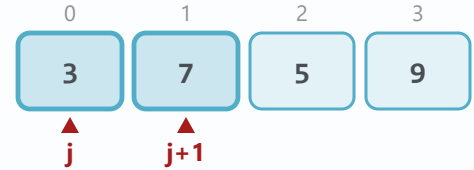
degisti = false

ÇIKTI



Kabarcık Sıralama: Kod İzleme

```
1 static void kabarcikSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int tur = 0; tur < n - 1; tur++) {  
4         boolean degisti = false;  
5         for (int j = 0; j < n - 1 - tur; j++) {  
6             if (dizi[j] > dizi[j + 1]) { // ters sıra  
7                 int gecici = dizi[j];  
8                 dizi[j] = dizi[j + 1];  
9                 dizi[j + 1] = gecici;  
10                degisti = true;  
11            }  
12        }  
13        if (!degisti) break; // takas yok:  
14    }  
15 }
```



3 > 7 → false

j = 0: 3 > 7 yanlış, komşular sırada.

Adım 3 / 9

DEĞİŞKENLER

n = 4

tur = 0

j = 0

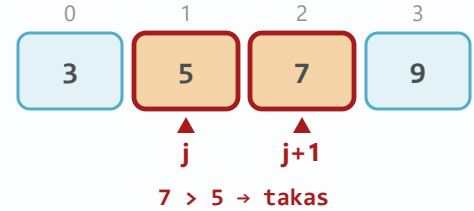
degisti = false

ÇIKTI



Kabarcık Sıralama: Kod İzleme

```
1 static void kabarcikSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int tur = 0; tur < n - 1; tur++) {  
4         boolean degisti = false;  
5         for (int j = 0; j < n - 1 - tur; j++) {  
6             if (dizi[j] > dizi[j + 1]) { // ters sıra  
7                 int gecici = dizi[j];  
8                 dizi[j] = dizi[j + 1];  
9                 dizi[j + 1] = gecici;  
10                degisti = true;  
11            }  
12        }  
13        if (!degisti) break; // takas yok:  
14        sıralı  
15    }  
16 }
```



j = 1: 7 > 5 → takas; degisti = true.

Adım 4 / 9

DEĞİŞKENLER

dizi = [3, 5, 7, 9]

tur = 0

j = 1

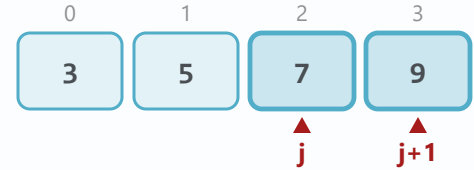
degisti = true

ÇIKTI



Kabarcık Sıralama: Kod İzleme

```
1 static void kabarcikSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int tur = 0; tur < n - 1; tur++) {  
4         boolean degisti = false;  
5         for (int j = 0; j < n - 1 - tur; j++) {  
6             if (dizi[j] > dizi[j + 1]) { // ters sıra  
7                 int gecici = dizi[j];  
8                 dizi[j] = dizi[j + 1];  
9                 dizi[j + 1] = gecici;  
10                degisti = true;  
11            }  
12        }  
13        if (!degisti) break; // takas yok:  
14        sıralı  
15    }
```



7 > 9 → false

j = 2: 7 > 9 yanlış. Tur bitti: en büyük öge (9) sona.

Adım 5 / 9

DEĞİŞKENLER

dizi = [3, 5, 7, 9]

tur = 0

j = 2

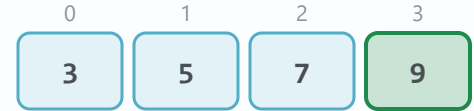
degisti = true

ÇIKTI



Kabarcık Sıralama: Kod İzleme

```
1 static void kabarcikSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int tur = 0; tur < n - 1; tur++) {  
4         boolean degisti = false;  
5         for (int j = 0; j < n - 1 - tur; j++) {  
6             if (dizi[j] > dizi[j + 1]) { // ters sıra  
7                 int gecici = dizi[j];  
8                 dizi[j] = dizi[j + 1];  
9                 dizi[j + 1] = gecici;  
10                degisti = true;  
11            }  
12        }  
13        if (!degisti) break; // takas yok:  
14    }  
15 }
```



degisti = true → devam

Bu turda takas oldu → **break** yok, yeni tura geçilir.

Adım 6 / 9

DEĞİŞKENLER

tur = 0

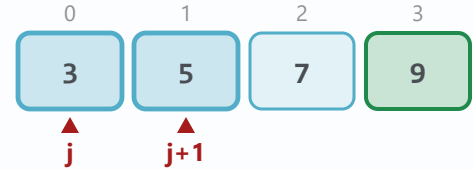
degisti = true

ÇIKTI



Kabarcık Sıralama: Kod İzleme

```
1 static void kabarcikSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int tur = 0; tur < n - 1; tur++) {  
4         boolean degisti = false;  
5         for (int j = 0; j < n - 1 - tur; j++) {  
6             if (dizi[j] > dizi[j + 1]) { // ters sıra  
7                 int gecici = dizi[j];  
8                 dizi[j] = dizi[j + 1];  
9                 dizi[j + 1] = gecici;  
10                degisti = true;  
11            }  
12        }  
13        if (!degisti) break; // takas yok:  
14        sıralı  
15    }  
16 }
```



3 > 5 → false

2. tur: `degisti` = false. `j` = 0: 3 > 5
yanlış.

Adım 7 / 9

DEĞİŞKENLER

`tur = 1`

`j = 0`

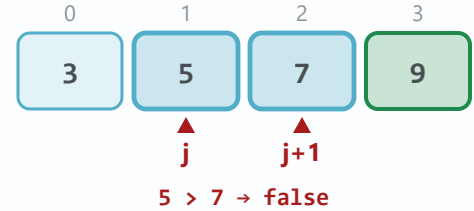
`degisti = false`

ÇIKTI



Kabarcık Sıralama: Kod İzleme

```
1 static void kabarcikSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int tur = 0; tur < n - 1; tur++) {  
4         boolean degisti = false;  
5         for (int j = 0; j < n - 1 - tur; j++) {  
6             if (dizi[j] > dizi[j + 1]) { // ters sıra  
7                 int gecici = dizi[j];  
8                 dizi[j] = dizi[j + 1];  
9                 dizi[j + 1] = gecici;  
10                degisti = true;  
11            }  
12        }  
13        if (!degisti) break; // takas yok:  
14        sıralı  
15    }
```



j = 1: 5 > 7 yanlış. İç döngü j < 2 olduğu için biter.

Adım 8 / 9

DEĞİŞKENLER

tur = 1 j = 1

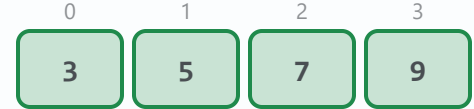
degisti = false

ÇIKTI



Kabarcık Sıralama: Kod İzleme

```
1 static void kabarcikSiralama(int[] dizi) {  
2     int n = dizi.length;  
3     for (int tur = 0; tur < n - 1; tur++) {  
4         boolean degisti = false;  
5         for (int j = 0; j < n - 1 - tur; j++) {  
6             if (dizi[j] > dizi[j + 1]) { // ters sıra  
7                 int gecici = dizi[j];  
8                 dizi[j] = dizi[j + 1];  
9                 dizi[j + 1] = gecici;  
10                degisti = true;  
11            }  
12        }  
13        if (!degisti) break; // takas yok:  
14        }  
15 }
```



degisti = false → break

Hiç takas olmadı: dizi sıralı. **break** ile 3. tur hiç yapılmaz; erken durma 1 karşılaştırma kazandırdı.

Adım 9 / 9

DEĞİŞKENLER

dizi = [3, 5, 7, 9]

tur = 1

degisti = false

ÇIKTI



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

Deneyin: Seçimli ve Kabarcık Sıralama

Yöntem

seçimli sıralama

▼

bir adım

sonuna kadar

yeni dizi

0

1

2

3

4

5

6

7

30

80

11

17

38

69

50

34

Seçmeli sıralama hazır. "bir adım" ile ilerleyin.

Her "bir adım" tek bir karşılaştırma yapar. Turuncu: karşılaştırılan öğeler, yeşil: kesin yerine oturmuş öğeler. İki yöntemin karşılaştırma ve takas sayılarını aynı dizi için kıyaslayın.

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 $i = 2 \dots n - 1$ yerine $2 \dots n$ yazılırsa $n + 1$ terim çıkar (off-by-one). Sınırları küçük bir testle elle izleyin.

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

ÖZET

Örnek Problemler

1. Faktöriyel algoritmasını **sayı = 4** için izleyin: her turda **i** ve **faktöriyel** 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 sıralama 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 sıralama 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 sıralamayı **büyükten küçüğe** sıralayacak şekilde değiştirin.
6. Fibonacci programını `int` ile $n = 50$ 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 (çevrimiçi 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: Algoritmalar

Sonraki bölüm: Bölüm 1: Akış Diyagramları · [Terim sözlüğü](#)