



Bölüm 1: Algoritmalar

JAVA ile Nesne Yönelimli Programlama

Program Geliştirme Metodolojisi: İlk Algoritma, Sonra Uygulama



- **Adım 1 - Sorunu Tanımlayın:** İlk adım, program geliştirme sürecinin temel taşıdır. Sorunu net bir şekilde tanımlamak, projenin başarısının anahtarıdır. Sorunun ne olduğunu anlamadan, bir çözüm bulmak imkansızdır.
- **Adım 2 - Bir Algoritma Belirleyin:** Sorunu tanımladıktan sonra sıradaki adım, bir çözüm yolu yaratmaktır. Bir algoritma, sorunu çözmek için adım adım yönergeler sunar. Doğru algoritmayı seçmek, programın verimli çalışmasını sağlar.
- **Adım 3 - Algoritmayı Kodlamaya Çevirin:** Algoritma belirlendikten sonra, seçilen programlama dilinde kodlama bu adımda gerçekleştirilir. Bu aşamada dikkatli ve sistematik bir şekilde çalışmak önemlidir.



Sorunu Tanımlama

- Program geliştirme sürecinin ilk ve en önemli adımlarından biri, sorunu doğru bir şekilde tanımlamaktır.
- Bu adım, projenin başarılı bir şekilde ilerlemesi için hayati bir öneme sahiptir.
- Doğru bir şekilde sorunu tanımlamak, programınızın başarılı bir şekilde ilerlemesine ve doğru sonuçlar üretmesine yardımcı olacaktır.
- Ayrıca, oluşturduğunuz test senaryoları, programınızı doğrulamak ve hataları belirlemek için son derece önemlidir.



Sorunu Tanımlama

▪ Adım A - Sorun Tanımını Yazın

- Bu adımda, programın girdi ve çıktıları doğal bir dilde açıklanır. Hangi verilerin programa gireceği ve programın ne tür bir çıktı üreteceği anlatılır.

▪ Adım B - Tüm Program İçin Test Senaryoları Oluşturun

- Bu adımda, programın doğru çalışıp çalışmadığını doğrulamak için kullanılacak test durumları oluşturulur. Bu test senaryoları, programa ne tür verilerin gireceğini ve programın bu verilere nasıl tepki vermesi gerektiği belirtilmelidir.



Örnek Test Senaryoları

- **Sorun Tanımı:** Bir kullanıcının girdiği iki sayının toplamını hesaplamak.
 - Test Senaryosu 1: Girdi : 3, 5 - Beklenen Çıktı : 8
 - Test Senaryosu 2: Girdi : -2, 7 - Beklenen Çıktı : 5
- **Sorun Tanımı:** Bir metin belgesinin içindeki kelime sayısını bulmak.
 - Test Senaryosu 1: Girdi : "Merhaba, dünya!" - Beklenen Çıktı : 2
 - Test Senaryosu 2: Girdi : "Bu bir denemedir." - Beklenen Çıktı : 3



Bir Algoritma Belirleme

- Program geliştirmenin ikinci önemli adımı, bir algoritma oluşturmayı içerir.
- Bu adım, sorunu mantıklı bir şekilde çözmek için gerekli adımları belirlemeyi içerir.
- Bir algoritma oluşturmak ve test etmek, program geliştirme sürecinin önemli bir parçasıdır.
- Algoritma, test senaryolarına göre, adım adım uygulanarak, problem daha iyi anlaşılabilir ve algoritma daha doğru hale getirilebilir.



Bir Algoritma Belirleme

- **Adım A - Algoritmayı Algoritmik Bir Şekilde Uygulayın**
- Algoritma oluşturmanın ilk alt adımı, algoritmayı algoritmik bir şekilde uygulamaktır. Bu, adım adım talimatlar veya tarifler yazmayı içerir.
- **Sorun:** İki sayının toplamını hesaplayın.
- **Algoritma:**
 - İlk sayıyı al.
 - İkinci sayıyı al.
 - İki sayıyı topla.
 - Toplamı ekrana yazdır.
- Bu adımlar, problemi mantıklı bir şekilde çözmek için bir rehber sağlar.



Bir Algoritma Belirleme

- **Adım B - Kağıt ve Kalem Kullanarak Test Edin**
- Algoritmanızı oluşturduktan sonra, bir sonraki adım onu test etmektir. Kağıt ve kalem kullanarak test etmek, algoritmanızın mantıklı ve işlevsel olduğunu doğrulamanıza yardımcı olabilir.
 - Küçük ama anlamlı test senaryoları oluşturun.
 - Algoritmayı adım adım takip edin ve her adımın sonucunu yazın.
 - Algoritmayı uygularken düşündüklerinizi fark edin.
 - Adımların gerçekleştirdiklerini gözlemleyin ve gerekirse daha hassas hale getirin.
- Bu adım, algoritmanın gerçek dünyada nasıl çalıştığını anlamaya yardımcı olacaktır.



Koda Dönüştürme

- Algoritmanızı bir programlama dilinde kodlamak, fikirlerinizi gerçeğe dönüştürmenin en heyecan verici kısmıdır.
- Mantıklı bir şekilde kodlama yapmak ve fonksiyonları kullanmak, programı daha anlaşılır, bakımı daha kolay ve genel olarak daha etkili hale getirecektir.



Koda Dönüştürme

- **Adım A - Bir Programlama Dilini Kullanarak Uygulama**
- Algoritmayı oluşturduktan sonra, sıradaki adım onu bir programlama dilinde kodlamaktır.
- Programlama dili, bilgisayarın anlayabileceği bir dilde algoritmanızı ifade etmenizi sağlar.
 - Algoritmayı mantıklı bir şekilde kodlayın, her adımı yazılı kod haline getirin.
 - Kodu okunaklı ve düzenli bir şekilde yazmaya özen gösterin.
 - Programlama dilinin özgün özelliklerini kullanarak algoritmayı en etkili şekilde ifade edin.



Koda Dönüştürme

- **Adım B - Mantıksal Ünitelere (Fonksiyonlar) Ayırma**
- Bir programı daha yönetilebilir ve anlaşılır hale getirmenin önemli bir yolu, kodu mantıksal parçalara ayırmaktır.
- Bu parçalar, genellikle "fonksiyonlar" olarak adlandırılır ve belirli bir görevi yerine getiren kod parçalarını içerirler.
 - Programdaki farklı işlevleri tanımlamak için fonksiyonları kullanın.
 - Her fonksiyonun ne yapacağını açıkça belirtin ve adını anlamlı bir şekilde seçin.
 - Fonksiyonlar, kodu daha modüler ve bakımı daha kolay hale getirir.



Örnek Kod Parçacıkları

```
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 number
        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();
    }
}
```



Örnek Kod Parçacıkları

```
import java.util.Scanner;

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



Neden Fonksiyonlar?

- Fonksiyonlar, programları daha anlaşılır, daha sürdürülebilir ve daha verimli hale getiren güçlü bir araçtır.
- Kodu daha kolay düzenlemek, daha hızlı hata ayıklamak ve başka projelerde yeniden kullanabilmek için fonksiyonların etkili bir şekilde kullanılması gerekir.
- **Neden 1 - Kod Parçalarını İsimlendirmek**
 - Yeni bir fonksiyon oluşturmak, bir grup ifadeyi (statements) adlandırma fırsatı sunar. Bu, programı daha okunabilir ve hata ayıklamasını daha kolay hale getirir. İsimlendirme, kodun amacını ve işlevini daha açık bir şekilde tanımlar.



Neden Fonksiyonlar?

▪ Neden 2 - Tekrarlı Kodları Ortadan Kaldırmak

- Fonksiyonlar, tekrarlı kodları ortadan kaldırarak programı daha küçük ve düzenli hale getirebilir. Sonradan bir değişiklik gerektiğinde, bu değişikliğin sadece bir kez yapılması yeterli olur. Bu, bakımı kolaylaştırır ve oluşabilecek hataları azaltır.

▪ Neden 3 - Uzun Programları Parçalara Ayırmak

- Uzun bir programı fonksiyonlara bölmek, her parçanın ayrı ayrı test edilebilmesine ve sonra çalışabilir bir bütün haline getirmeye olanak tanır. Karmaşık projeleri daha yönetilebilir hale getirir ve hata izleme sürecini kolaylaştırır.

▪ Neden 4 - İyi Tasarlanmış Fonksiyonların Yeniden Kullanılabilirliği

- İyi tasarlanmış fonksiyonlar genellikle birçok farklı program için kullanışlıdır. Bir kez yazılıp test edildiğinde, başka projelerde yeniden kullanılabilir. Bu, kodun verimli bir şekilde kullanılmasına ve zaman tasarrufuna yardımcı olur.



Su Düşüşü Geliştirme Stratejisi (Waterfall)

- Projeleri aşamalı bir şekilde yönetmek ve denetlemek için güçlü bir araçtır. Bu strateji, projeyi beş ana aşamaya böler ve her aşamanın tamamlanmasının ardından bir sonraki aşamaya geçilir.
- Beş ana aşamadan oluşur:
 - **İhtiyaç Analizi:** Proje gereksinimleri ve hedefleri belirlenir. Müşteri veya kullanıcı ihtiyaçları anlaşılır.
 - **Tasarım:** Proje tasarlanır. Yazılımın nasıl çalışacağı ve nasıl görüneceği planlanır.
 - **Geliştirme:** Kodlama aşamasıdır. Yazılımın aslı oluşturulur ve test edilir.
 - **Test ve Kalite Kontrol:** Yazılım, hataları düzeltmek ve istenen işlevselliği sağlamak için test edilir.
 - **Dağıtım ve Bakım:** Yazılım kullanıma sunulur ve kullanıcıların geri bildirimlerine yanıt vermek için bakım yapılır.



Su Düşüşü Geliştirme Stratejisi

■ Avantajlar

- Proje ilerlemesi belirli aşamalara ayrıldığı için daha iyi kontrol edilebilir.
- Her aşama tamamlandığında, sonraki aşamaya geçilmeden önce gereksinimler ve tasarım doğrulanabilir.
- İyi belirlenmiş gereksinimler, tasarım ve kodlama aşamaları hataların daha erken tanımlanmasını sağlar.

■ Dezavantajlar

- Geri dönüş zor olabilir. Bir aşama tamamlandığında, değişiklikler maliyetli ve karmaşık olabilir.
- Proje gereksinimleri başlangıçta eksik veya yanlış anlaşılabilir, bu da projeyi etkileyebilir.
- Müşteri gereksinimlerinde değişikliklerle başa çıkmak zor olabilir.



Döngülü Geliştirme Stratejisi (Iterative)

- Yazılım projelerini küçük parçalara böler ve her parçayı ayrı ayrı geliştirir.
- Her aşama tamamlandığında, geri bildirim alınır ve projenin bir sonraki aşamasına geçilir.
- Bu strateji, projenin daha esnek ve değişken gereksinimlere yanıt verebilir olmasını sağlar.



Ana Prensipler

- Döngülü Geliştirme Stratejisi, şu ana prensiplere dayanır:
 - **Küçük İterasyonlar:** Projeyi küçük ve yönetilebilir parçalara bölme. Her bir iterasyon, belirli bir işlevselliği veya özelliği ele alır.
 - **Geri Bildirim Odaklı:** Her iterasyon sonrası geri bildirim alınır ve bu geri bildirim, projenin gelişimini şekillendirir.
 - **Esneklik:** Değişen gereksinimlere hızlı ve etkili bir şekilde uyum sağlama yeteneği.



Döngülü Geliştirme Süreci

- Döngülü geliştirme stratejisi genellikle aşağıdaki süreci takip eder:
 - **İlk İterasyon:** Temel gereksinimler belirlenir ve ilk prototip oluşturulur.
 - **İterasyonlar:** Küçük geliştirme döngüleri ile yazılımın işlevselliği artırılır ve iyileştirilir.
 - **Geri Bildirim:** Her iterasyon sonrası, kullanıcılar ve paydaşlardan geri bildirim alınır.
 - **Yeniden Değerlendirme ve İterasyon:** Geri bildirimlere dayalı olarak, projenin hedefleri ve gereksinimleri yeniden değerlendirilir ve yeni iterasyonlar planlanır.



Döngülü Geliştirme Stratejisi

■ Avantajlar

- Değişen gereksinimlere kolayca uyum sağlayabilir.
- Kullanıcılar ve paydaşlar sürekli olarak projeye dahil edilebilir.
- Hatalar ve sorunlar daha erken tespit edilir ve düzeltilir.
- İyileştirmeler ve yeni özellikler daha hızlı eklenir.

■ Dezavantajlar

- İterasyonlar arasındaki geçişler zaman ve kaynak gerektirebilir.
- İterasyonlar çok sayıda olabilir ve yönetim karmaşıklığı yaratabilir.
- Proje süreci daha esnek olduğu için bazen belirsizlik yaratabilir.



Örnek Algoritmalar

- İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Faktöriyel Bulma
- Fibonacci Serisi Bulma
- Palindrom Kontrolü
- Asal Sayı Kontrolü
- Üs Alma



İki Tamsayıyı Küçükten Büyüğe Doğru 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
```

Bitir



İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

Başla

Yaz "Birinci tamsayıyı girin: "

Oku `birinciTamsayı`

Yaz "İkinci tamsayıyı girin: "

Oku `ikinciTamsayı`

birinci	ikinci
4	

Eğer `birinciTamsayı < ikinciTamsayı` ise

Yaz `birinciTamsayı, " ", ikinciTamsayı`

Değilse

Yaz `ikinciTamsayı, " ", birinciTamsayı`

Bitir



İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

Başla

Yaz "Birinci tamsayıyı girin: "
Oku birinciTamsayı

Yaz "İkinci tamsayıyı girin: "
Oku ikinciTamsayı

birinci	ikinci
4	11

Eğer birinciTamsayı < ikinciTamsayı ise
Yaz birinciTamsayı, " ", ikinciTamsayı
Değilse
Yaz ikinciTamsayı, " ", birinciTamsayı

Bitir



İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

Başla

Yaz "Birinci tamsayıyı girin: "
Oku birinciTamsayı

Yaz "İkinci tamsayıyı girin: "
Oku ikinciTamsayı

birinci	ikinci
4	11

Eğer birinciTamsayı < ikinciTamsayı ise

Yaz birinciTamsayı, " ", ikinciTamsayı

Değilse

Yaz ikinciTamsayı, " ", birinciTamsayı

Bitir



İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

Başla

Yaz "Birinci tamsayıyı girin: "
Oku birinciTamsayı

Yaz "İkinci tamsayıyı girin: "
Oku ikinciTamsayı

birinci	ikinci
4	11
4	11

Eğer birinciTamsayı < ikinciTamsayı ise
Yaz birinciTamsayı, " ", ikinciTamsayı
Değilse
Yaz ikinciTamsayı, " ", birinciTamsayı

Bitir



İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

Başla

Yaz "Birinci tamsayıyı girin: "

Oku **birinciTamsayı**

Yaz "İkinci tamsayıyı girin: "

Oku **ikinciTamsayı**

birinci	ikinci
13	

Eğer **birinciTamsayı < ikinciTamsayı** ise

Yaz **birinciTamsayı, " ", ikinciTamsayı**

Değilse

Yaz **ikinciTamsayı, " ", birinciTamsayı**

Bitir



İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

Başla

Yaz "Birinci tamsayıyı girin: "
Oku birinciTamsayı

birinci	ikinci
13	11

Yaz "İkinci tamsayıyı girin: "
Oku ikinciTamsayı

Eğer birinciTamsayı < ikinciTamsayı ise
Yaz birinciTamsayı, " ", ikinciTamsayı
Değilse
Yaz ikinciTamsayı, " ", birinciTamsayı

Bitir



İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

Başla

Yaz "Birinci tamsayıyı girin: "
Oku birinciTamsayı

Yaz "İkinci tamsayıyı girin: "
Oku ikinciTamsayı

birinci	ikinci
13	11

Eğer birinciTamsayı < ikinciTamsayı ise

Yaz birinciTamsayı, " ", ikinciTamsayı

Değilse

Yaz ikinciTamsayı, " ", birinciTamsayı

Bitir



İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

Başla

Yaz "Birinci tamsayıyı girin: "
Oku birinciTamsayı

birinci	ikinci
13	11

Yaz "İkinci tamsayıyı girin: "
Oku ikinciTamsayı

Eğer birinciTamsayı < ikinciTamsayı ise
Yaz birinciTamsayı, " ", ikinciTamsayı

Değilse

Yaz ikinciTamsayı, " ", birinciTamsayı

Bitir



İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

Başla

Yaz "Birinci tamsayıyı girin: "
Oku birinciTamsayı

Yaz "İkinci tamsayıyı girin: "
Oku ikinciTamsayı

birinci	ikinci
13	11
11 13	

Eğer birinciTamsayı < ikinciTamsayı ise
Yaz birinciTamsayı, " ", ikinciTamsayı
Değilse
Yaz ikinciTamsayı, " ", birinciTamsayı

Bitir



Örnek Algoritmalar

- İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- **Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma**
- Faktöriyel Bulma
- Fibonacci Serisi Bulma
- Palindrom Kontrolü
- Asal Sayı Kontrolü
- Üs Alma



Üç Tamsayıyı Küçükten Büyüğe Doğru 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
```



Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
// Tamsayıları 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
End Eğer
```



Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

Değilse Eğer ikinciTamsayı <= birinciTamsayı ve ikinciTamsayı <= üçüncüTamsayı ise

Yaz ikinciTamsayı

Eğer birinciTamsayı <= üçüncüTamsayı ise

Yaz birinciTamsayı, " ", üçüncüTamsayı

Değilse

Yaz üçüncüTamsayı, " ", birinciTamsayı

End Eğer



Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

Değilse

Yaz üçüncüTamsayı

Eğer birinciTamsayı \leq ikinciTamsayı ise

Yaz birinciTamsayı, " ", ikinciTamsayı

Değilse

Yaz ikinciTamsayı, " ", birinciTamsayı

End Eğer

End Eğer

Bitir



Örnek Algoritmalar

- İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- **Faktöriyel Bulma**
- Fibonacci Serisi Bulma
- Palindrom Kontrolü
- Asal Sayı Kontrolü
- Üs Alma



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü  
Döngü i = 1'den başlayarak sayı'ya kadar  
    faktoriyel = faktoriyel * i  
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

Input: sayı

sayı	faktoriyel	i
5		

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü  
Döngü i = 1'den başlayarak sayı'ya kadar  
    faktoriyel = faktoriyel * i  
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	1	

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

```
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü
```

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

```
    faktoriyel = faktoriyel * i
```

```
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	1	1

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü
```

Döngü i = 1'den başlayarak sayı'ya kadar

```
    faktoriyel = faktoriyel * i
```

```
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	1	1

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü  
Döngü i = 1'den başlayarak sayı'ya kadar  
    faktoriyel = faktoriyel * i  
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	1	2

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü
```

Döngü i = 1'den başlayarak sayı'ya kadar

```
    faktoriyel = faktoriyel * i
```

```
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	2	2

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü  
Döngü i = 1'den başlayarak sayı'ya kadar  
    faktoriyel = faktoriyel * i  
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	2	3

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü
```

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

```
    faktoriyel = faktoriyel * i
```

```
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	6	3

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü  
Döngü i = 1'den başlayarak sayı'ya kadar  
    faktoriyel = faktoriyel * i  
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	6	4

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü
```

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

```
    faktoriyel = faktoriyel * i
```

```
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	24	4

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü  
Döngü i = 1'den başlayarak sayı'ya kadar  
    faktoriyel = faktoriyel * i  
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	24	5

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü
```

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

```
    faktoriyel = faktoriyel * i
```

```
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	120	5

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü  
Döngü i = 1'den başlayarak sayı'ya kadar  
    faktoriyel = faktoriyel * i  
Döngüyü Bitir
```

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Faktöriyel Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

sayı	faktoriyel	i
5	120	5

```
// Sonucu saklamak için bir değişkeni başlatın  
faktoriyel = 1
```

```
// Faktöriyel hesabı için döngü  
Döngü i = 1'den başlayarak sayı'ya kadar  
    faktoriyel = faktoriyel * i
```

Döngüyü Bitir

```
Yaz "Girilen sayının faktöriyeli: ", faktoriyel
```

Bitir



Örnek Algoritmalar

- İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Faktöriyel Bulma
- **Fibonacci Serisi Bulma**
- Palindrom Kontrolü
- Asal Sayı Kontrolü
- Üs Alma



Fibonacci Serisi Bulma

Başla

```
// Bir sayıyı girin
```

```
Input: n
```

```
// İlk iki Fibonacci sayısını başlatın
```

```
a = 0
```

```
b = 1
```

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

```
sonuc = 0
```

```
Eğer n = 0 ise
```

```
Yaz 0
```

```
Değilse
```



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

Bitir



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

n	a	b	i	sonuc
5	0	1		

Bitir



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

n	a	b	i	sonuc
5	0	1		
0				

Bitir



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

n	a	b	i	sonuc
5	0	1		
0 1				

Bitir



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

n	a	b	i	sonuc
5	0	1	2	
0 1				

Bitir



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

n	a	b	i	sonuc
5	0	1	2	1
0 1				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	1	1	2	1
0 1				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	1	1	2	1
0 1				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	1	1	2	1
0 1 1				

Bitir



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

n	a	b	i	sonuc
5	1	1	3	1
0 1 1				

Bitir



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

n	a	b	i	sonuc
5	1	1	3	2
0 1 1				

Bitir



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

n	a	b	i	sonuc
5	1	1	3	2
0 1 1				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	1	2	3	2
0 1 1				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	1	2	3	2
0 1 1 2				

Bitir



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

n	a	b	i	sonuc
5	1	2	4	2
0 1 1 2				

Bitir



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

n	a	b	i	sonuc
5	1	2	4	3
0 1 1 2				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	2	2	4	3
0 1 1 2				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	2	3	4	3
0 1 1 2				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	2	3	4	3
0	1	1	2	3

Bitir



Fibonacci Serisi Bulma

Değilse

// İlk n Fibonacci sayısını hesaplayın

Yaz a

Yaz b

Döngü i = 2'den başlayarak n - 1'e kadar

sonuc = a + b

a = b

b = sonuc

Yaz sonuc

Döngüyü Bitir

End Eğer

n	a	b	i	sonuc
5	2	3	5	3
0 1 1 2 3				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	2	3	5	5
0 1 1 2 3				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	3	3	5	5
0 1 1 2 3				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	3	5	5	5
0 1 1 2 3				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

```
Döngüyü Bitir
```

```
End Eğer
```

n	a	b	i	sonuc
5	3	5	5	5
0 1 1 2 3 5				

Bitir



Fibonacci Serisi Bulma

Değilse

```
// İlk n Fibonacci sayısını hesaplayın
```

```
Yaz a
```

```
Yaz b
```

```
Döngü i = 2'den başlayarak n - 1'e kadar
```

```
    sonuc = a + b
```

```
    a = b
```

```
    b = sonuc
```

```
    Yaz sonuc
```

Döngüyü Bitir

```
End Eğer
```

n	a	b	i	sonuc
5	3	5	5	5
0 1 1 2 3 5				

Bitir



Örnek Algoritmalar

- İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Faktöriyel Bulma
- Fibonacci Serisi Bulma
- **Palindrom Kontrolü**
- Asal Sayı Kontrolü
- Üs Alma



Palindrom Kontrolü

Başla

```
// Bir kelime veya cümlenin girilmesini isteyin
```

```
Input: girdiMetin
```

```
// Temizlenmiş girdiyi ters çevirin
```

```
tersGirdi = TersÇevir(girdiMetin)
```

```
// Temizlenmiş girdi ile ters girdiyi karşılaştırın
```

```
Eğer girdiMetin = tersGirdi ise
```

```
    Yaz "Girilen metin bir palindromdur."
```

```
Değilse
```

```
    Yaz "Girilen metin bir palindrom değildir."
```

```
End Eğer
```

Bitir



Palindrom Kontrolü

Başla

```
// Bir kelime veya cümlenin girilmesini isteyin
```

```
Input: girdiMetin
```

```
// Temizlenmiş girdiyi ters çevirin  
tersGirdi = TersÇevir(girdiMetin)
```

girdiMetin	tersGirdi

```
// Temizlenmiş girdi ile ters girdiyi karşılaştırın
```

```
Eğer girdiMetin = tersGirdi ise
```

```
Yaz "Girilen metin bir palindromdur."
```

```
Değilse
```

```
Yaz "Girilen metin bir palindrom değildir."
```

```
End Eğer
```

Bitir



Palindrom Kontrolü

Başla

```
// Bir kelime veya cümlenin girilmesini isteyin
```

Input: girdiMetin

```
// Temizlenmiş girdiyi ters çevirin  
tersGirdi = TersÇevir(girdiMetin)
```

girdiMetin	tersGirdi
abcba	

```
// Temizlenmiş girdi ile ters girdiyi karşılaştırın
```

```
Eğer girdiMetin = tersGirdi ise
```

```
Yaz "Girilen metin bir palindromdur."
```

```
Değilse
```

```
Yaz "Girilen metin bir palindrom değildir."
```

```
End Eğer
```

Bitir



Palindrom Kontrolü

Başla

```
// Bir kelime veya cümlenin girilmesini isteyin
```

```
Input: girdiMetin
```

```
// Temizlenmiş girdiyi ters çevirin
```

```
tersGirdi = TersÇevir(girdiMetin)
```

girdiMetin	tersGirdi
abcba	abcba

```
// Temizlenmiş girdi ile ters girdiyi karşılaştırın
```

```
Eğer girdiMetin = tersGirdi ise
```

```
Yaz "Girilen metin bir palindromdur."
```

```
Değilse
```

```
Yaz "Girilen metin bir palindrom değildir."
```

```
End Eğer
```

Bitir



Palindrom Kontrolü

Başla

```
// Bir kelime veya cümlenin girilmesini isteyin
```

```
Input: girdiMetin
```

```
// Temizlenmiş girdiyi ters çevirin  
tersGirdi = TersÇevir(girdiMetin)
```

girdiMetin	tersGirdi
abcba	abcba

```
// Temizlenmiş girdi ile ters girdiyi karşılaştırın
```

```
Eğer girdiMetin = tersGirdi ise
```

```
Yaz "Girilen metin bir palindromdur."
```

```
Değilse
```

```
Yaz "Girilen metin bir palindrom değildir."
```

```
End Eğer
```

Bitir



Palindrom Kontrolü

Başla

```
// Bir kelime veya cümlenin girilmesini isteyin
```

```
Input: girdiMetin
```

```
// Temizlenmiş girdiyi ters çevirin  
tersGirdi = TersÇevir(girdiMetin)
```

girdiMetin	tersGirdi
abcba	abcba
Girilen metin bir palindromdur	

```
// Temizlenmiş girdi ile ters girdiyi karşılaştırın  
Eğer girdiMetin = tersGirdi ise
```

```
    Yaz "Girilen metin bir palindromdur."
```

```
Değilse
```

```
    Yaz "Girilen metin bir palindrom değildir."
```

```
End Eğer
```

Bitir



Örnek Algoritmalar

- İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Faktöriyel Bulma
- Fibonacci Serisi Bulma
- Palindrom Kontrolü
- **Asal Sayı Kontrolü**
- Üs Alma



Asal Sayı Kontrolü

Başla

```
// Bir sayıyı girin
```

```
Input: sayı
```

```
// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
```

```
asalMi = True
```

```
// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
```

```
Döngü bölen = 2'den başlayarak sayı / 2'e kadar
```

```
    Eğer sayı % bölen = 0 ise
```

```
        // Bölünüyorsa, sayı asal değildir
```

```
        asalMi = False
```

```
        Döngüyü Kır
```

```
    End Eğer
```

```
Döngüyü Bitir
```




Asal Sayı Kontrolü

```
// Bayrağa (flag) göre sonucu yazdırın  
Eğer asalMi = True ise  
    Yaz sayı, " bir asal sayıdır."  
Değilse  
    Yaz sayı, " bir asal sayı değildir."  
End Eğer
```

Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
7		

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et

Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
7	True	

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın

asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et

Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
7	True	2

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et

Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
7	True	2

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
7	True	3

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et

Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
7	True	3

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
7	True	3

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et

Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
21		

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et

Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
21	True	

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın

asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et

Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
21	True	2

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et

Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
21	True	2

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
21	True	3

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et

Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
21	True	3

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

// Bir sayıyı girin

Input: sayı

sayı	asalMi	bölen
21	False	3

// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
asalMi = True

// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et

Döngü bölen = 2'den başlayarak sayı / 2'e kadar

Eğer sayı % bölen = 0 ise

// Bölünüyorsa, sayı asal değildir

asalMi = False

Döngüyü Kır

End Eğer

Döngüyü Bitir



Asal Sayı Kontrolü

Başla

```
// Bir sayıyı girin
```

Input: sayı

sayı	asalMi	bölen
7	False	3

```
// Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın  
asalMi = True
```

```
// 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
```

```
Döngü bölen = 2'den başlayarak sayı / 2'e kadar
```

```
Eğer sayı % bölen = 0 ise
```

```
    // Bölünüyorsa, sayı asal değildir
```

```
    asalMi = False
```

```
    Döngüyü Kır
```

```
End Eğer
```

Döngüyü Bitir



Örnek Algoritmalar

- İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Faktöriyel Bulma
- Fibonacci Serisi Bulma
- Palindrom Kontrolü
- Asal Sayı Kontrolü
- **Üs Alma**

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

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

```
sonuc = 1
```

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

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

```
    sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

Üs Alma



Başla

```
// Bir sayıyı girin
```

Input: taban

```
// Bir üssü girin
```

Input: üs

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

```
sonuc = 1
```

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

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

```
sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

taban	üs	sonuc	i
3			

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

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

```
sonuc = 1
```

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

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

```
    sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

taban	üs	sonuc	i
3	4		

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

taban	üs	sonuc	i
3	4	1	

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

```
sonuc = 1
```

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

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

```
sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

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

```
sonuc = 1
```

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

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

```
sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

taban	üs	sonuc	i
3	4	1	1

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

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

```
sonuc = 1
```

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

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

```
sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

taban	üs	sonuc	i
3	4	3	1

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

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

```
sonuc = 1
```

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

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

```
sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

taban	üs	sonuc	i
3	4	3	2

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

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

```
sonuc = 1
```

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

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

```
sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

taban	üs	sonuc	i
3	4	9	2

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

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

```
sonuc = 1
```

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

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

```
sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

taban	üs	sonuc	i
3	4	9	3

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

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

```
sonuc = 1
```

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

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

```
sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

taban	üs	sonuc	i
3	4	27	3

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

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

```
sonuc = 1
```

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

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

```
sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

taban	üs	sonuc	i
3	4	27	4

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

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

```
sonuc = 1
```

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

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

```
sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

taban	üs	sonuc	i
3	4	81	4

Üs Alma



Başla

```
// Bir sayıyı girin
```

```
Input: taban
```

```
// Bir üssü girin
```

```
Input: üs
```

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

```
sonuc = 1
```

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

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

```
sonuc = sonuc * taban
```

```
Yaz "Sonuç: ", sonuc
```

Bitir

taban	üs	sonuc	i
3	4	81	4



Arama Algoritmaları

- Arama algoritmaları, bir koleksiyon içerisinde belirli bir öğenin varlığını aramak veya bulmak için kullanılır.
 - Örneğin, bir listede belirli bir sayının varlığını kontrol etmek için arama algoritmaları kullanılır.
- Verileri etkili bir şekilde işlemek ve değerli bilgileri çıkarmak için temel bir işlemdir.
- Belirli öğelerin hızlı ve doğru bir şekilde bulunmasına yardımcı olur.
- **Örnek Kullanım Alanları:** Kitapların kütüphanedeki yerini bulma, İnternet üzerinde bir web sitesinde belirli bir sayfayı bulma, veritabanında belirli bir kaydı arama.



Arama Algoritmaları

- **Problem Tanımı**
- **Girdi:**
 - Nesnelerin bir koleksiyonu, «Sepet» diyelim.
 - Belirli bir nesne, «Elma» diyelim.
- **Çıktı:**
 - Eğer «Elma», "Sepet" içindeyse "True".
 - Eğer «Elma», "Sepet" içinde değilse "False".



Arama Algoritmaları

- Rastgele (Random)
- Doğrusal (Linear)
- İkili (Binary)



Rastgele Arama

- Rastgele arama, bir koleksiyon içerisindeki bir öğeyi bulmak için rastgele bir şekilde öğeleri kontrol eden bir arama algoritmasıdır.
- Her adımda bir öğeyi rastgele seçer ve hedefi bulana veya arama alanı tükenene kadar devam eder.
- Büyük veri koleksiyonlarında hızlı bir şekilde arama yapmak için kullanılabilir.
- **Avantajlar:** Basit ve hızlıdır, veri koleksiyonunun sırası önemli değildir.
- **Dezavantajlar:** En kötü durumda veri koleksiyonunun tamamını incelemek gerekebilir, etkili olmayabilir.



Rastgele Arama

▪ Nasıl Çalışır?

- Rastgele bir öge seç.
- Seçilen ögeyi hedefle karşılaştır.
- Eğer hedef bulunduysa işlemi sonlandır.
- Eğer hedef bulunamazsa adımları tekrarla.



Rastgele Arama

- **Soru:** Rastgele Arama, Arama Problemini Çözer mi?
 - Rastgele arama her zaman etkili bir çözüm sağlamaz.
 - Rastgele arama, öğeyi bulma olasılığını artırabilir, ancak kesin bir sonuç garantisi vermez.
- **Soru:** Eğer Öğe «Sepet» İçerisinde Değilse Ne Olur?
 - Eğer «Elma» öğesi "Sepet" içinde değilse, rastgele arama işlemi sona erebilir ve «Elma» bulunamayabilir.
 - Bu nedenle, rastgele arama kullanırken öğenin koleksiyon içinde olup olmadığını doğrulamak önemlidir.



Rastgele Arama

- Elimizde rastgele bir dizi olsun: [77, 42, 64, 12, **23**, 55, 7, 48, 31]
- Aramak istediğimiz öge: 48
- Adım 1:
 - Dizi üzerinde rastgele bir eleman seçin ve aranan öge (48) ile karşılaştırın.
 - Örneğin, ilk seçilen eleman 23'tür, bu nedenle 23 ile 48'i karşılaştırın.



Rastgele Arama

- Elimizde rastgele bir dizi olsun: [77, **42**, 64, 12, 23, 55, 7, 48, 31]
- Aramak istediğimiz öge: 48
- Adım 2:
 - Aranan öge (48) henüz bulunamadı, çünkü seçilen eleman 23'tür.
 - Dizi üzerinde bir başka rastgele eleman seçin ve tekrar karşılaştırın.
 - Bu sefer seçilen eleman 42'dir.



Rastgele Arama

- Elimizde rastgele bir dizi olsun: [77, 42, 64, 12, 23, 55, **7**, 48, 31]
- Aramak istediğimiz öge: 48
- Adım 3:
 - 42 ile 48'i karşılaştırın.
 - Aranan öge (48) henüz bulunamadı, çünkü seçilen eleman 42'dir.
 - Yeni bir rastgele eleman seçin.



Rastgele Arama

- Elimizde rastgele bir dizi olsun: [77, 42, 64, 12, 23, 55, 7, **48**, 31]
- Aramak istediğimiz öge: 48
- Adım 4:
- Bu adımları tekrarlayarak aranan öğeyi (48) bulana kadar devam edin.
- Örneğin, bir sonraki seçilen eleman 48 olduğunda aranan öge bulundu.



Arama Algoritmaları

- Rastgele (Random)
- **Doğrusal (Linear)**
- İkili (Binary)



Doğrusal Arama

- Doğrusal arama, bir koleksiyon içinde bir öğeyi bulmanın en temel yoludur.
- Öğeleri sırayla kontrol ederek aranan öğeyi bulmaya çalışır.
- Küçük koleksiyonlarda etkili bir çözümdür.
- Öğelerin sıralı olması gerekmez, herhangi bir sırayla bulunabilirler.
- **Avantajlar:** Basit ve anlaşılır, herhangi bir sırayla çalışabilir.
- **Dezavantajlar:** Büyük koleksiyonlarda yavaş olabilir, her öğeyi kontrol etmesi gerekebilir.



Doğrusal Arama

▪ Nasıl Çalışır?

- Öğeleri bir listeye koyun
- İlk öğeden başlayarak sırayla hedef öğe ile karşılaştırın.
- Eğer hedef öğe bulunursa işlemi sonlandırın ve bulunan öğeyi dönün.
- Eğer hedef öğe bulunmazsa bir sonraki öğeye geçin.
- Listenin sonuna kadar devam edin.



Doğrusal Arama

- **Soru:** Doğrusal Arama, Arama Problemini Çözer mi?
 - **Evet**, Doğrusal Arama herhangi bir liste için ve herhangi bir öğe için problemi çözecektir.
 - Doğrusal Arama, belirli bir öğeyi bulmak için her öğeyi sırayla kontrol eder.
 - Hedef öğeyi bulana veya tüm öğeleri kontrol edene kadar devam eder.



Doğrusal Arama

- Elimizde sıralanmamış bir dizi olsun: [42, 17, 8, 23, 31]
- Adım 1:
 - Aranacak öğeyi belirleyin, örneğin 23'ü aranıyor.
 - Dizinin ilk elemanından başlayın (42) ve aranan öğe ile karşılaştırın.



Doğrusal Arama

- Elimizde sıralanmamış bir dizi olsun: [42, **17**, 8, 23, 31]
- Adım 2:
- İlk eleman (42) aranan öge (23) ile eşleşmez.
- Bir sonraki elemana geçin ve 17 ile karşılaştırın.



Doğrusal Arama

- Elimizde sıralanmamış bir dizi olsun: [42, 17, **8**, 23, 31]
- Adım 3:
- İkinci eleman (17) de aranan öge ile eşleşmez.
- Bir sonraki elemana geçin ve 8 ile karşılaştırın.



Doğrusal Arama

- Elimizde sıralanmamış bir dizi olsun: [42, 17, 8, **23**, 31]
- Adım 4:
- Üçüncü eleman (8) de aranan öge ile eşleşmez.
- Bir sonraki elemana geçin ve 23 ile karşılaştırın.



Doğrusal Arama

- Elimizde sıralanmamış bir dizi olsun: [42, 17, 8, **23**, 31]
- Adım 5:
- Dördüncü eleman (23), aranan öge ile eşleşir.
- Aranan öğeyi bulduk ve bu adımda arama işlemi sona erer.



Arama Algoritmaları

- Rastgele (Random)
- Doğrusal (Lineer)
- **İkili (Binary)**



İkili Arama

- Bir koleksiyon içerisindeki bir öğeyi bulmanın hızlı ve etkili bir yoludur.
- Ancak, verilerin **sıralı** (küçükten büyüğe veya büyükten küçüğe) olması gerekmektedir.
- Büyük veri koleksiyonlarında hızlı bir şekilde arama yapmak için idealdir.
- Her adımda listenin yarısı kadar öğeyi elemeyi sağlar.
- **Avantajlar:** Hızlı, büyük veri koleksiyonlarında etkilidir.
- **Dezavantajlar:** Veriler sıralı olmalıdır, koleksiyonun yeniden düzenlenmesi gerekebilir.



İkili Arama

▪ Nasıl Çalışır?

- Ortadaki öğeyi seçin ve hedefle karşılaştırın.
- Eğer hedef öge bulunursa işlemi sonlandırın.
- Eğer hedef öge, ortadaki öğeden küçükse, sol yarıya odaklanın ve adımları tekrarlayın.
- Eğer hedef öge, ortadaki öğeden büyükse, sağ yarıya odaklanın ve adımları tekrarlayın.
- Hedef öge bulunana veya arama alanı tükenene kadar bu adımları tekrarlayın.



İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, **42**, 48, 55, 64, 77]
- Aramak istediğimiz öge: 48
- Adım 1:
- Dizinin ortasındaki elemanı (42) alın ve aranan öge (48) ile karşılaştırın.



İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, 48, 55, 64, 77]
- Aramak istediğimiz öge: 48
- Adım 2:
 - 48, ortadaki elemandan (42) büyüktür.
 - Bu nedenle, aranan öğenin dizinin sağ yarısında olduğunu anlarız.
 - Sol yarı artık arama için kullanılmayacak.



İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, 48, 55, 64, 77]
- Aramak istediğimiz öge: 48
- Adım 3:
- Sağ yarıyı ele alın: [48, 55, 64, 77]



İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, 48, 55, **64**, 77]
- Aramak istediğimiz öge: 48
- Adım 4:
- Sağ yarının ortasındaki elemanı (64) alın ve aranan öge (48) ile karşılaştırın.



İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, 48, 55, 64, 77]
- Aramak istediğimiz öge: 48
- Adım 5:
 - 48, ortadaki elemandan (64) küçüktür.
 - Bu nedenle, aranan öğenin dizinin sol yarısında olduğunu anlarız.
 - Sağ yarı artık arama için kullanılmayacak.



İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, 48, 55, 64, 77]
- Aramak istediğimiz öge: 48
- Adım 6:
- Sol yarıyı ele alın: [48, 55]



İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, **48**, 55, 64, 77]
- Aramak istediğimiz öge: 48
- Adım 7:
- Sol yarının ortasındaki elemanı (48) alın ve aranan öge (48) ile karşılaştırın.



İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, 48, 55, 64, 77]
- Aramak istediğimiz öge: 48
- Adım 8:
- Aranan öge (48) bulundu.
- Arama işlemi başarıyla tamamlandı.



İkili Arama vs. Lineer Arama: Karşılaştırma

- Lineer arama, bir koleksiyon içinde bir öğeyi sırayla kontrol ederek aramaya çalışır. Her adımda bir öğeyi kontrol eder ve hedefi bulana veya koleksiyon sonuna kadar devam eder.
- İkili arama, sıralı verilerde hedef öğeyi daha hızlı bulmak için kullanılır. Verilerin sıralı (küçükten büyüğe veya büyükten küçüğe) olması gerekir.
- **Lineer Arama:** Büyük koleksiyonlarda yavaş olabilir, her öğeyi kontrol etmesi gerekebilir. **İkili Arama:** Büyük koleksiyonlarda hızlıdır, her adımda yarıya kadar öğeleri eleme avantajı vardır.
- Örnek Kullanım:
 - Lineer Arama: Telefon rehberinde bir kişiyi bulma.
 - İkili Arama: Sözlük içinde bir kelimeyi bulma.



Sıralama Algoritmaları

- Sıralama algoritmaları, bir veri koleksiyonundaki öğeleri belirli bir kurala göre düzenlemek için kullanılır.
- Bu düzenleme genellikle öğeleri küçükten büyüğe veya büyükten küçüğe sıralamak şeklinde olur.
- Verileri düzenlemek ve sıralamak, veri analizi ve arama işlemleri için temel bir adımdır.
- Sıralı veriler, arama işlemlerini hızlandırabilir.



Sıralama Algoritmaları

- **Rastgele Sıralama (Random)**
- **Seçmeli Sıralama (Selection Sort)**
- **Kabarcık Sıralama (Bubble Sort)**



Rastgele Sıralama

- Bir veri koleksiyonunu rastgele bir düzende sıralama işlemidir.
- Veri koleksiyonunu rastgele hale getirerek verileri çeşitlendirmek ve analiz etmek için kullanılır.
- Bazı algoritmaların başlangıç verilerini rastgele sıralama işlemi ile iyileştirmesinde kullanılır.
- Nasıl Çalışır?
 - Veri koleksiyonundaki öğeleri rastgele bir şekilde karıştırır.
 - Öğelerin sıralamasını tamamen rastgele yapar, önceki sıralamayla ilgilenmez.
- **Avantajlar:** Verileri rastgele sıralayarak daha çeşitli sonuçlar elde edilebilir.
- **Dezavantajlar:** Verilerin doğru bir şekilde sıralanmasını sağlamaz.



Rastgele Sıralama

- Elimizde sıralanmamış bir dizi olsun: [42, 17, 8, 23, 31]
- Adım 1:
- Dizi elemanlarını rastgele bir sırayla düzenleyin.
- Her düzenleme farklı bir sonuç üretecektir. Örneğin, dizi aşağıdaki gibi bir düzende olabilir: [31, 17, 42, 8, 23]



Rastgele Sıralama

- Elimizde sıralanmamış bir dizi olsun: [31, 17, 42, 8, 23]
- Adım 2:
- Dizi elemanlarını tekrar rastgele bir sırayla düzenleyin.
- Bu sefer farklı bir düzenleme olabilir: [8, 23, 31, 42, 17]



Rastgele Sıralama

- Elimizde sıralanmamış bir dizi olsun: [8, 23, 31, 42, 17]
- Adım 3:
- Dizi elemanlarını tekrar rastgele bir sırayla düzenleyin.
- Bu sefer farklı bir düzenleme olabilir: [8, 23, 17, 31, 42]



Sıralama Algoritmaları

- Rastgele Sıralama (Random)
- **Seçmeli Sıralama (Selection Sort)**
- Kabarcık Sıralama (Bubble Sort)



Seçmeli Sıralama

- Bir veri koleksiyonundaki öğeleri sırayla karşılaştırarak sıralamak için kullanılan basit bir sıralama algoritmasıdır.
- Bu algoritma, en küçük veya en büyük öğeyi bulup sırayla yer değiştirerek sıralamayı gerçekleştirir.
- **Avantajlar:** Basit ve anlaşılır, küçük veri koleksiyonlarında etkili.
- **Dezavantajlar:** Büyük veri koleksiyonlarında yavaş, daha verimli algoritmalar mevcut.



Seçmeli Sıralama

▪ Nasıl Çalışır?

- Veri koleksiyonundaki en küçük (veya en büyük) öğeyi bulun.
- Bu öğeyi sıralı koleksiyonun başına taşıyın.
- Koleksiyonun bir sonraki bölümünü ele alın ve en küçük (veya en büyük) öğeyi bulun.
- Bu öğeyi sıralı koleksiyonun ikinci pozisyonuna taşıyın.
- Bu adımları koleksiyonun sonuna kadar tekrarlayın.



Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [64, 25, 12, 22, **11**, 36, 48]
- Adım 1:
 - Sıralanmamış kısmın içindeki en küçük elemanı bulun (bu durumda 11).
 - İlk elemanla (64) yer değiştirin.
 - Dizi birinci adımdan sonra: [**11**, 25, 12, 22, 64, 36, 48]



Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [11, 25, 12, 22, 64, 36, 48]
- Adım 2:
- İkinci en küçük elemanı bulun (bu durumda 12) ve ikinci elemanla yer değiştirin.
- Dizi ikinci adımdan sonra: [11, 12, 25, 22, 64, 36, 48]



Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [11, 12, 25, 22, 64, 36, 48]
- Adım 3:
- Üçüncü en küçük elemanı bulun (bu durumda 22) ve üçüncü elemanla yer değiştirin.
- Dizi üçüncü adımdan sonra: [11, 12, 22, 25, 64, 36, 48]



Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [11, 12, 22, 25, 64, 36, 48]
- Adım 4:
- Dördüncü en küçük elemanı bulun (bu durumda 25) ve dördüncü elemanla yer değiştirin.
- Dizi dördüncü adımdan sonra: [11, 12, 22, 25, 64, 36, 48]



Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [11, 12, 22, 25, 64, 36, 48]
- Adım 5:
- Beşinci en küçük elemanı bulun (bu durumda 36) ve beşinci elemanla yer değiştirin.
- Dizi beşinci adımdan sonra: [11, 12, 22, 25, 36, 64, 48]



Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [11, 12, 22, 25, 36, 64, 48]
- Adım 6:
- Altıncı en küçük elemanı bulun (bu durumda 48) ve altıncı elemanla yer değiştirin.
- Dizi altıncı adımdan sonra: [11, 12, 22, 25, 36, 48, 64]



Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [11, 12, 22, 25, 36, 48, 64]
- Adım 7:
- Dizinin sonuna ulaşıldığı için değişiklik yok.
- Dizi yedinci adımdan sonra: [11, 12, 22, 25, 36, 48, 64]



Sıralama Algoritmaları

- Rastgele Sıralama (Random)
- Seçmeli Sıralama (Selection Sort)
- **Kabarcık Sıralama (Bubble Sort)**



Kabarcık Sıralama

- Bir veri koleksiyonundaki öğeleri sırayla karşılaştırarak sıralamak için kullanılan basit bir sıralama algoritmasıdır.
- Bu algoritma, yan yana duran öğeleri sırayla karşılaştırarak en büyük öğeyi koleksiyonun sonuna doğru "kabarcık" gibi taşır.
- **Avantajlar:** Basit ve anlaşılır, küçük veri koleksiyonlarında etkili.
- **Dezavantajlar:** Büyük veri koleksiyonlarında çok yavaş, daha verimli algoritmalar mevcut.



Kabarcık Sıralama

■ Nasıl Çalışır?

- Koleksiyonun başından itibaren başlayın.
- İki yan yana öğeyi karşılaştırın ve gerektiğinde yerlerini değiştirin, büyük olan öğeyi sağa taşıyın.
- Koleksiyonun sonuna kadar bu adımları tekrarlayın.
- En büyük öğeyi, koleksiyonun sonuna taşındığı için son öğeyi sıralanmış kabul edin.
- İlk dört adımı koleksiyonun sonuna kadar tekrarlayarak sıralama işlemi tamamlanır.



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [64, 25, 12, 22, 11, 36, 48]
- Adım 1:
 - Dizinin ilk iki elemanını karşılaştırın (64 ve 25).
 - 25, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
 - Dizi birinci adımdan sonra: [25, 64, 12, 22, 11, 36, 48]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [25, **64**, **12**, 22, 11, 36, 48]
- Adım 2:
- Şimdi ikinci ve üçüncü elemanları karşılaştırın (64 ve 12).
- 12, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi ikinci adımdan sonra: [25, **12**, **64**, 22, 11, 36, 48]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [25, 12, **64**, **22**, 11, 36, 48]
- Adım 3:
- Üçüncü ve dördüncü elemanları karşılaştırın (64 ve 22).
- 22, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi üçüncü adımdan sonra: [25, 12, **22**, **64**, 11, 36, 48]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [25, 12, 22, **64**, **11**, 36, 48]
- Adım 4:
 - Dördüncü ve beşinci elemanları karşılaştırın (64 ve 11).
 - 11, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
 - Dizi dördüncü adımdan sonra: [25, 12, 22, **11**, **64**, 36, 48]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [25, 12, 22, 11, **64, 36**, 48]
- Adım 5:
 - Beşinci ve altıncı elemanları karşılaştırın (64 ve 36).
 - 36, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
 - Dizi beşinci adımdan sonra: [25, 12, 22, 11, **36, 64**, 48]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [25, 12, 22, 11, 36, **64**, **48**]
- Adım 6:
 - Altıncı ve yedinci elemanları karşılaştırın (64 ve 48).
 - 48, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
 - Dizi altıncı adımdan sonra: [25, 12, 22, 11, 36, **48**, **64**]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [25, 12, 22, 11, 36, 48, 64]
- Adım 7:
- Dizinin ilk iki elemanını karşılaştırın (25 ve 12).
- 12, 25'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi 7. adımdan sonra: [12, 25, 22, 11, 36, 48, 64]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [12, **25**, **22**, 11, 36, 48, 64]
- Adım 8:
 - Dizinin ikinci ve üçüncü elemanını karşılaştırın (25 ve 22).
 - 22, 25'ten küçük olduğu için bu iki elemanın yerini değiştirin.
 - Dizi 8. adımdan sonra: [12, **22**, **25**, 11, 36, 48, 64]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [12, 22, **25**, **11**, 36, 48, 64]
- Adım 9:
 - Dizinin üçüncü ve dördüncü elemanını karşılaştırın (25 ve 11).
 - 11, 25'ten küçük olduğu için bu iki elemanın yerini değiştirin.
 - Dizi 9. adımdan sonra: [12, 22, **11**, **25**, 36, 48, 64]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [12, 22, 11, **25, 36**, 48, 64]
- Adım 10:
 - Dizinin dördüncü ve beşinci elemanını karşılaştırın (25 ve 36).
 - 36, 25'ten büyük olduğu için değişiklik yok.
 - Dizi 10. adımdan sonra: [12, 22, 11, **25, 36**, 48, 64]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [12, 22, 11, 25, **36, 48**, 64]
- Adım 11:
 - Dizinin beşinci ve altıncı elemanını karşılaştırsın (36 ve 48).
 - 48, 36'dan büyük olduğu için değişiklik yok.
 - Dizi 11. adımdan sonra: [12, 22, 11, 25, **36, 48**, 64]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [12, **22**, **11**, 25, 36, 48, 64]
- Adım 13:
 - Dizinin ikinci ve üçüncü elemanını karşılaştırın (22 ve 11).
 - 11, 22'den küçük olduğu için iki elemanın yerini değiştirin.
 - Dizi 13. adımdan sonra: [12, **11**, **22**, 25, 36, 48, 64]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [12, 11, 22, 25, 36, 48, 64]
- Adım 16:
 - Dizinin birinci ve ikinci elemanını karşılaştırın (12 ve 11).
 - 11, 12'den küçük olduğu için iki elemanın yerini değiştirin.
 - Dizi 16. adımdan sonra: [11, 12, 22, 25, 36, 48, 64]



Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [11, 12, 22, 25, 36, 48, 64]
- Adım 21:
- Dizinin birinci ve ikinci elemanını karşılaştırsın (11 ve 12).
- 12, 11'den büyük olduğu için değişiklik yok.
- Dizi 21. adımdan sonra: [11, 12, 22, 25, 36, 48, 64]



SON