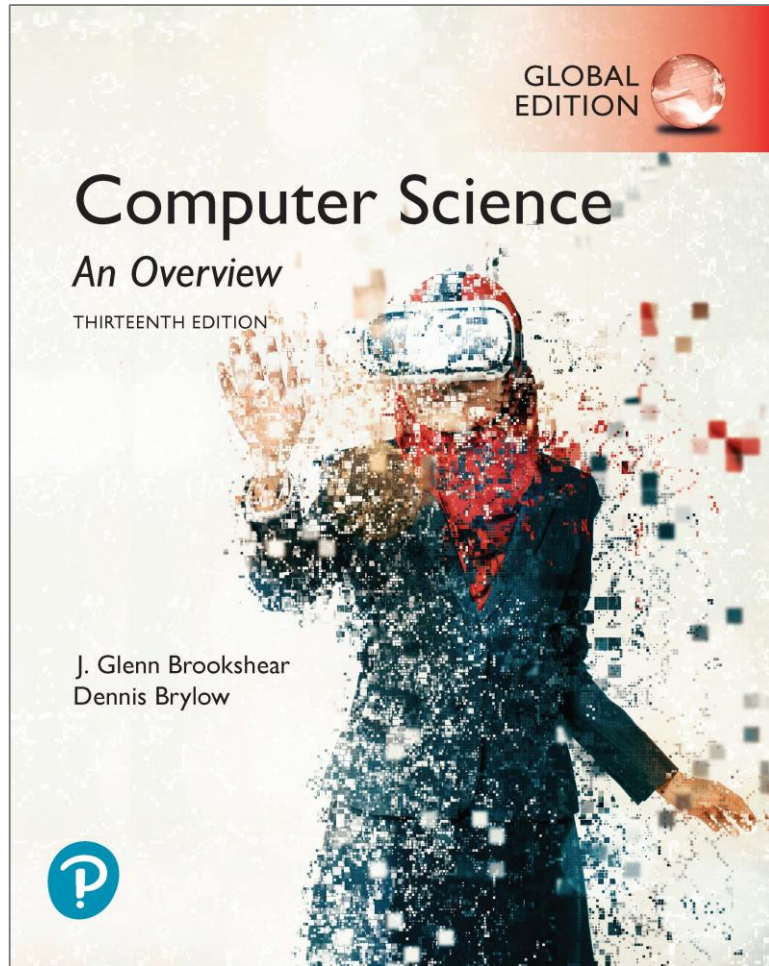


Bilgisayar Bilimine Giriş

13. Baskı, Global Edition



Bölüm 5

Algoritmalar

5.1 Bir algoritmanın fikri

- Bir çok araştırmacı insan beyninin her etkinliğinin bir algoritmanın sonucu olduğuna inanıyor

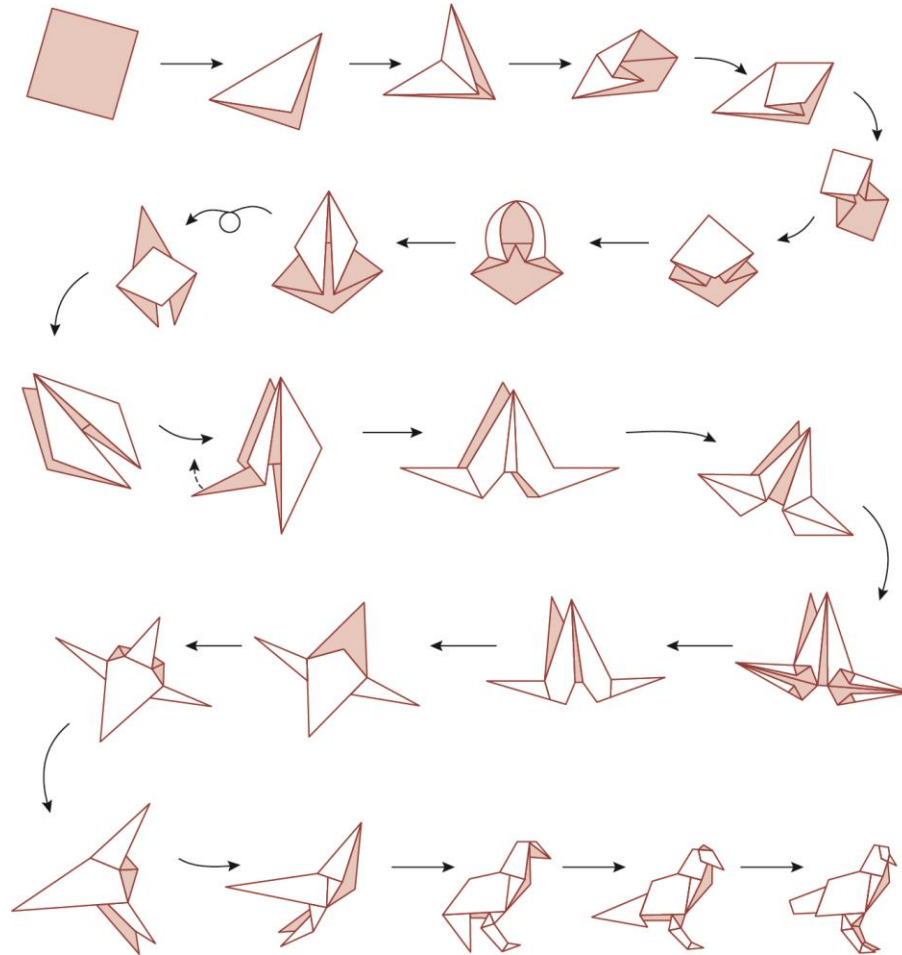
Algoritmanın Resmi Tanımı

- Bir algoritma işlemi tanımlayan kesin, çalıştırılabilir ve mantıklı adımların bir araya gelmesidir
- Algoritmanın adımları farklı şekillerde sıralanabilir
 - Doğrusal (1, 2, 3, ...)
 - Paralel (çoklu işlemci)

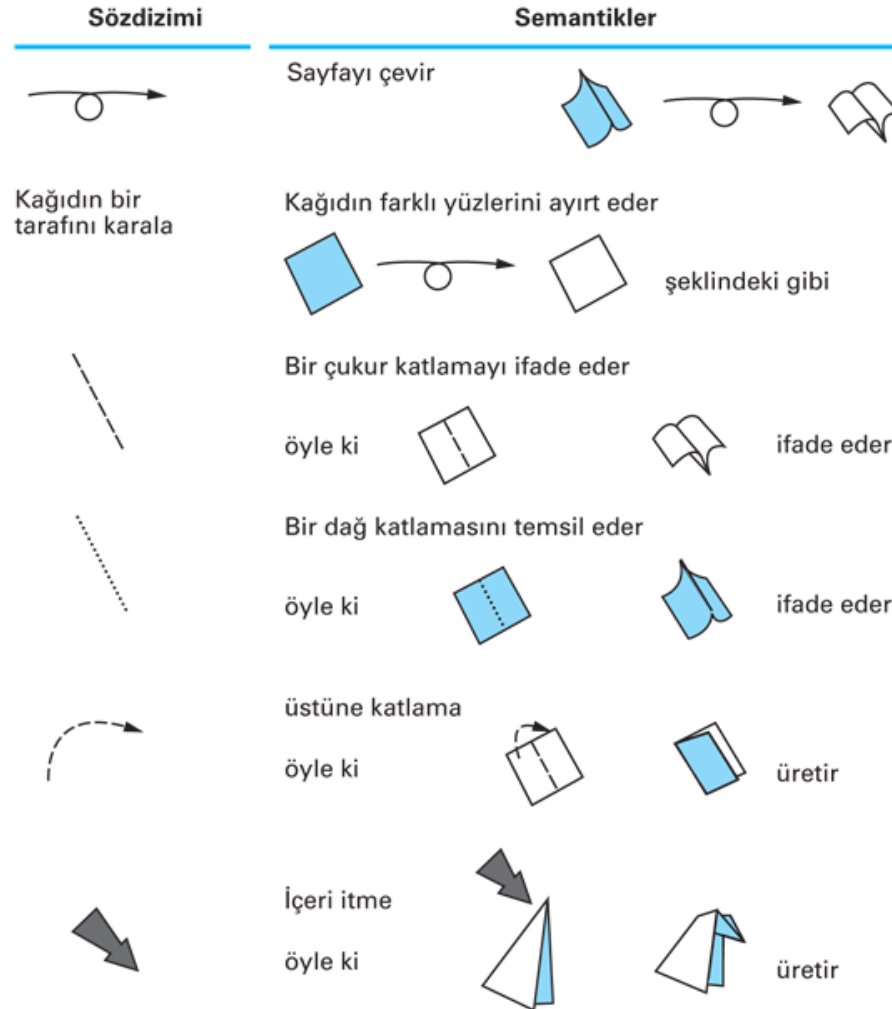
5.2 Algoritmanın Gösterimi

- İyi tanımlanmış Primitiflerle resmi bir şekilde yapılabilir
 - Bir Primitif topluluğu programlama dilini oluşturur
- Pseudocode(Sözde kod)ile resmi olmayan bir şekilde yapılabilir
 - Pseudocode konuşma diliyle kodlama dili arasında bir dildir.

Şekil 5.2 Kare şeklinde bir kağıdı katlayarak kuş yapma



Şekil 5.3 Origami primitifleri



Örnek: Primitiflerle algoritma oluşturma için bir uygulama

- Web (Flash Gerektirir)

- <https://lightbot.com/flash.html>

- Mobile

- AppStore

<https://apps.apple.com/tr/app/lightbot-code-hour/id873943739?l=tr>

- Google Play

<https://play.google.com/store/apps/details?id=com.lightbot.lightbothoc>

Bir Pseudocode Tasarlama (Sözde Kod)

- Sözde kod, bilgisayar bilimleri alanında algoritmalar ve programlar oluşturulurken ve aktarılırken kullanılan, günlük konuşma diline benzer ve belli bir programlama dilinin detaylarından uzak anlatımlardır.
- Programın yapısının ve çalışma mantığının yüksek seviyeli bir biçimde, gerektiği yerde doğrudan doğal dil cümleleriyle, ama yine de bir program yapısı ve akışı içinde anlatılmasıdır.
- Böylelikle sözde kodu okuyan ya da yazan birisi, programlama dillerinin sözdizim detaylarına dikkat etmek zorunda kalmadan, programın ve algoritmanın çalışma mantığını anlayabilir.

Designing a Pseudocode (Sözde Kod)

Düz Yazı

1. BAŞLA
2. Birinci sayıyı gir
3. İkinci sayıyı gir
4. İki sayıyı topla
5. Sayıların toplam değerini yaz
6. BİTİR

Sözde Kod

Toplam için T, birinci sayı için X, ikinci sayı için Y seç

1. BAŞLA
2. X değerini OKU
3. Y değerini OKU
4. $T = X + Y$
5. T değerini YAZ
6. BİTİR

Pseudocode Primitifleri

- Atama

name = *expression*

- Örnek:

RemainingFunds = CheckingBalance + SavingsBalance

Pseudocode Primitifleri (devamı)

- Koşullu seçim

```
if (koşul):  
    faaliyet
```

- örnek

```
if (satışlar < 100):  
    fiyatı %5 düşür
```

Pseudocode Primitifleri (devamı)

- Koşullu seçim

```
if (koşul):  
    faaliyet  
else:  
    faaliyet
```

- örnek

```
if (yıl artık yıl mı):  
    günlük toplam = toplam / 366  
else:  
    günlük toplam = toplam / 365
```

Pseudocode Primitifleri (devamı)

- Tekrarlı Çalıştırma

```
while (koşul):  
    gövde
```

- örnek

```
while (satılacak bilet varsa):  
    bileti sat
```

Pseudocode Primitifleri (devamı)

- Satırın başındaki boşluk iç içe koşulları gösterir

```
if (yağmur yağmıyor):  
    if (sıcaklık == çok sıcak):  
        yüzmeye git  
    else:  
        golf oyna  
else:  
    televizyon izle
```

Pseudocode Primitifleri (devamı)

- Fonksiyon tanımlama

```
def isim():
```

- örnek

```
def Ahmetinisi():
```

- Fonksiyonu çağırma

```
Ahmetinisi()
```

Şekil 5.4 Selamlar fonksiyonunun sözde kodu

```
def Selamlar():  
    sayac = 3  
    while (sayac > 0):  
        print('Merhaba')  
        sayac = sayac - 1
```


5.3 Algoritmanın keşfi

- Program geliştirmenin ilk adımıdır
- Bir yetenekten ziyade sanattır

Polya'nın Sorun Çözme Algoritması

- 1. Problemi anla.
- 2. Problemi çözmek için bir plan yap.
- 3. Planı işleme koy.
- 4. Çözümün benzer diğer problemleri çözmek için potansiyel ve verimlilik açısından yeterli olup olmadığını kontrol et.

* matematikçi «G. Polya» 1945'te tanımlamıştır.

Program Geliştirmede Polya'nın Algoritması

- 1. Problemi anla.
- 2. Problemi nasıl bir algoritmanın çözebileceği hakkında bir fikir edin.
- 3. Algoritmayı bul ve program haline getir.
- 4. Çözümün benzer diğer problemleri çözmek için potansiyel ve verimlilik açısından yeterli olup olmadığını kontrol et.

Çocukların Yaşı

- A kişisi B'nin üç çocuğunun yaşlarını bulmakla görevlendirilmiştir.
 - B A'ya çocuklarının yaşının çarpımının 36 olduğunu söylüyor.
 - A başka bir bilgi daha istiyor.
 - B A'ya çocukların yaşının toplamını söylüyor.
 - A başka bir bilgi daha istiyor.
 - B A'ya en büyük çocuğun piyano çaldığını söylüyor.
 - A B'ye çocukların yaşlarını söylüyor.
- Üç çocuk kaç yaşında?

Şekil 5.5

a. Çarpımı 36 olan üçlüler

(1,1,36) (1,6,6)

(1,2,18) (2,2,9)

(1,3,12) (2,3,6)

(1,4,9) (3,3,4)

b. (a) kısmındaki üçlülerin toplamaları

$$1 + 1 + 36 = 38$$

$$1 + 6 + 6 = 13$$

$$1 + 2 + 18 = 21$$

$$2 + 2 + 9 = 13$$

$$1 + 3 + 12 = 16$$

$$2 + 3 + 6 = 11$$

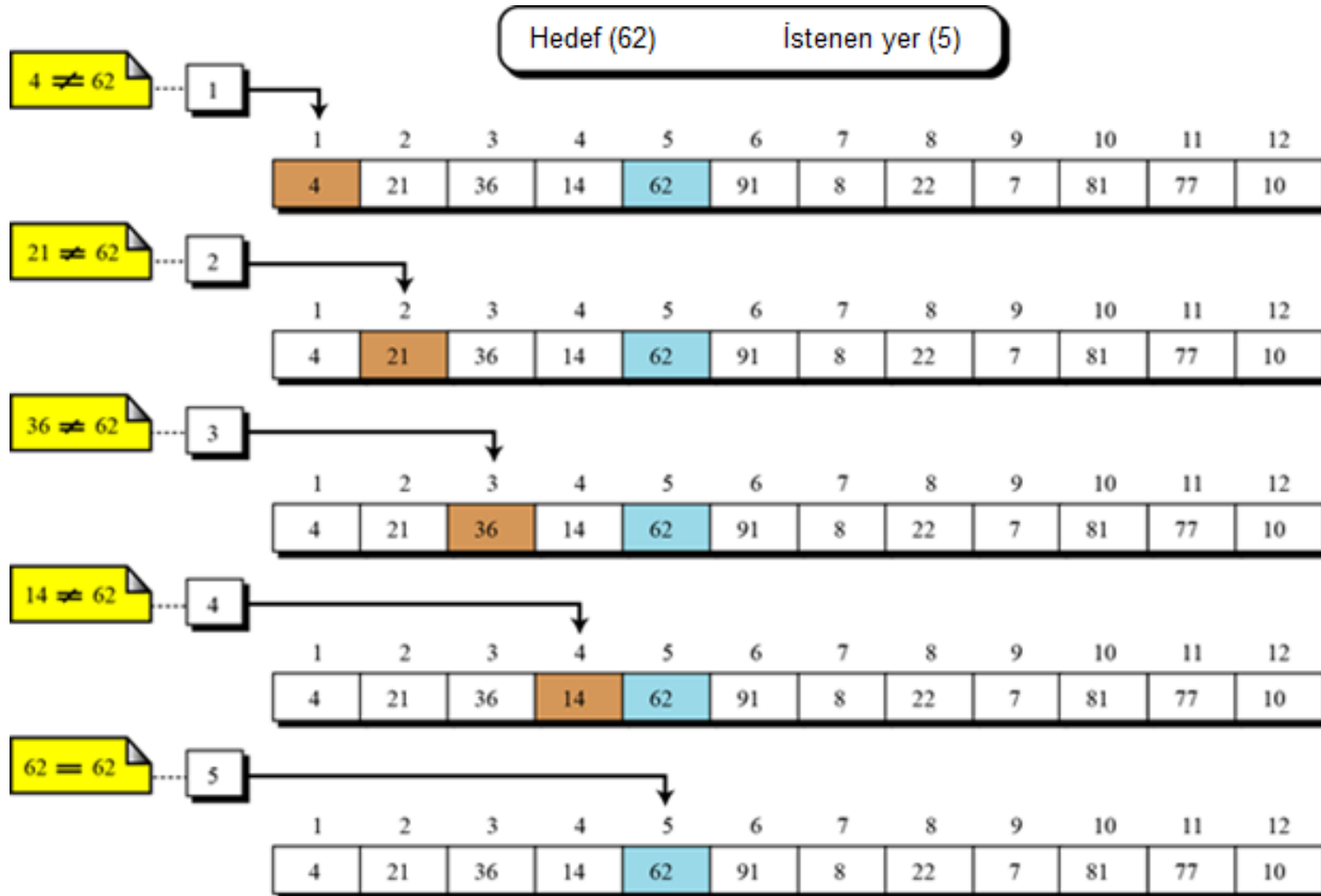
$$1 + 4 + 9 = 14$$

$$3 + 3 + 4 = 10$$

5.4 İteratif Yapılar

- Döngüsel bir yolda tekrar eden yönergeler topluluğu
- Örnek olarak:
 - Ardışık arama algoritması
 - Eklemeli sıralama algoritması

Ardışık Arama Algoritması



İteratif Yapılar

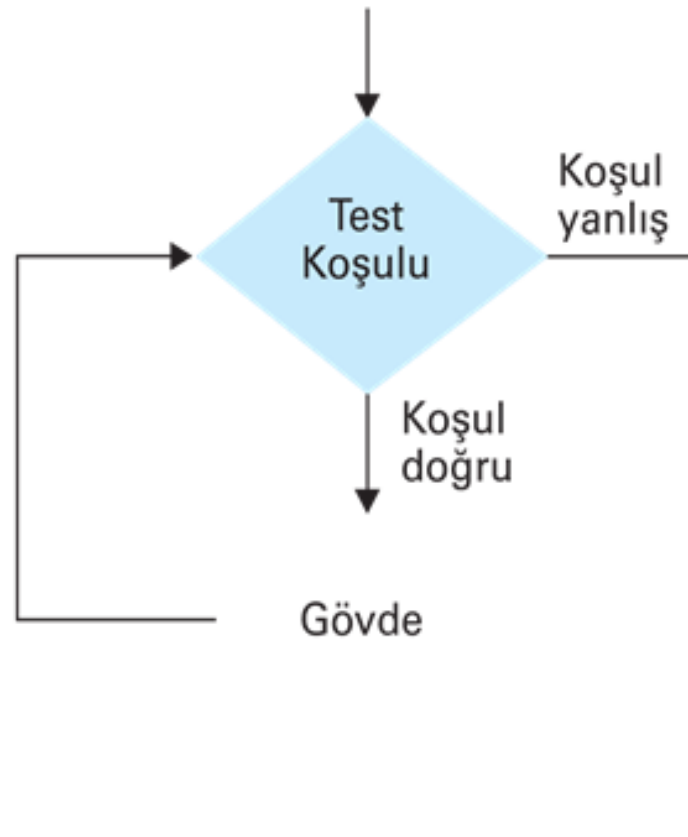
- Döngü öncesi kontrol:

```
while (koşul):  
    gövde
```

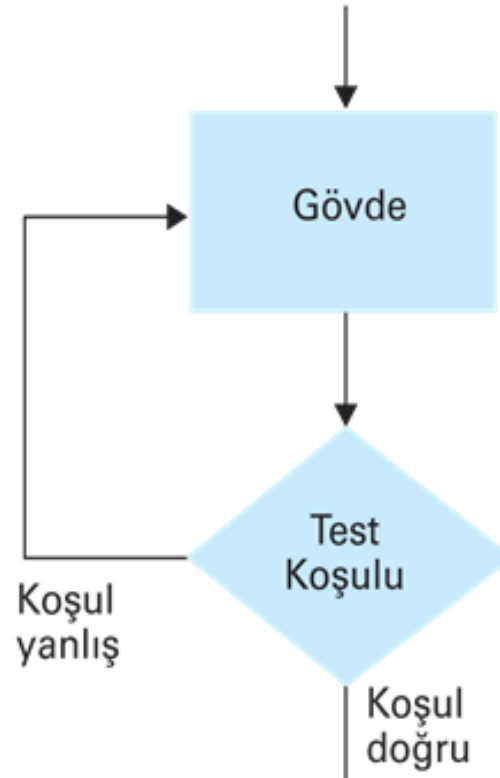
- Döngü sonrası kontrol:

```
repeat:  
    gövde  
until(koşul)
```


Şekil 5.8 While döngü yapısı



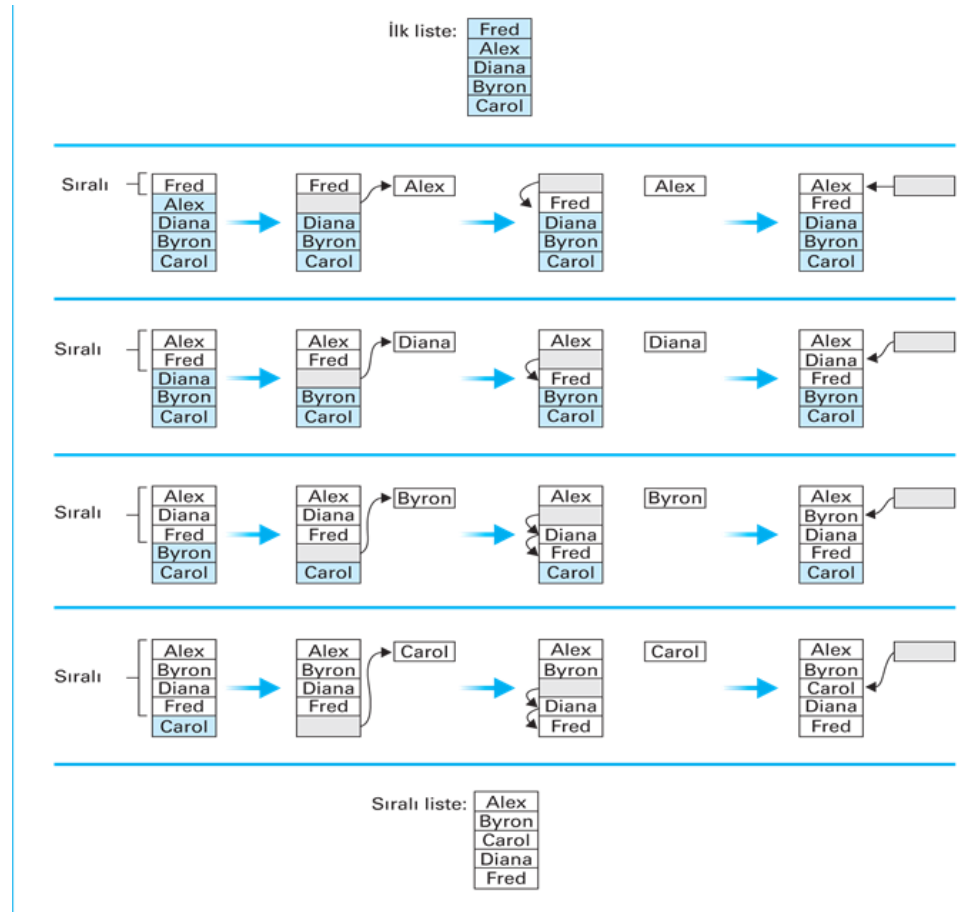
Şekil 5.9 Repeat döngü yapısı



Eklemeli Sıralama (Insertion Sort)

<https://visualgo.net/bn/sorting>

Şekil 5.10 Fred, Alex, Diana, Byron ve Carol isimlerini alfabetik olarak sıralayınız

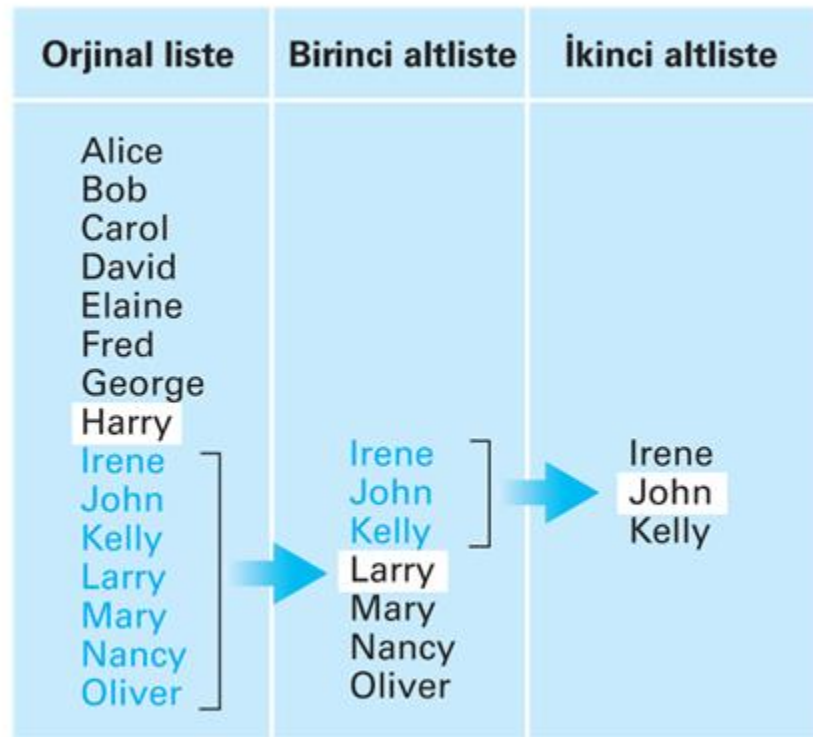


5.5 Özyinelemeli Fonksiyonlar

- Özyinelemeli yapılar tekrarlı faaliyetlerin gerçekleştirilmesi için döngü paradigmasına bir alternatif sunar. Bir döngü tekrarlanan bir dizi talimat içerirken, özyineleme tekrarlanan bir dizi talimatı kendisinin bir alt görevi olarak içerir.
- Örnek: İkilik sayı arama algoritması

Şekil 5.12 Bir listede John elemanını aramak için stratejinin uygulanması

Liste Sıralı olmak zorundadır!



Recursive Example:Towers of Hanoi

Find the Algorithm of Towers of Hanoi

<https://www.mathsisfun.com/games/towerofhanoi.html>

Solution in C Language

<https://www.geeksforgeeks.org/c-program-for-tower-of-hanoi-2/>

Solution in Python

<https://www.geeksforgeeks.org/python-program-for-tower-of-hanoi/>

5.6 Efficiency and Correctness

- Verimli (efficient) ve verimsiz (inefficient) algoritmalar arasındaki seçim, pratik bir çözüm ile pratik olmayan bir çözüm arasındaki farkı yaratabilir.
- Bir algoritmanın doğruluğu, algoritmanın uygulanmasını test ederek değil, resmi olarak algoritma hakkında akıl yürütme yoluyla belirlenir.

Efficiency

- Measured as number of instructions executed
- Uses big theta notation:
 - Example: Insertion sort is in $\Theta(n^2)$