
COMPUTER SCIENCE
An Overview
12th Edition
J. Glenn Brookshear
Dennis Brylow
Çeviri Editörü
Yrd. Doç. Dr. Birim Balcı Demirci
BİLGİSAYAR BİLİMİNE
Giriş
12. Basımdan Çeviri
PEARSON
nobel

İÇİNDEKİLER

ALGORİTMALAR

BİR ALGORİTMA KAVRAMI

ALGORİTMA GÖSTERİMİ

ALGORİTMA KEŞFİ

İTERATİF YAPILAR

ÖZYİNELEMELİ YAPILAR

VERİMLİLİK VE DOĞRULUK

BÖLÜM 5
ALGORİTMALAR

BİR ALGORİTMA KAVRAMI

Şekil 5.1 Algoritma tanımı

> Algoritma sonlanan bir işlemi tanımlayan kesin, çalıştırılabilir adımların sıralı bir kümesidir.

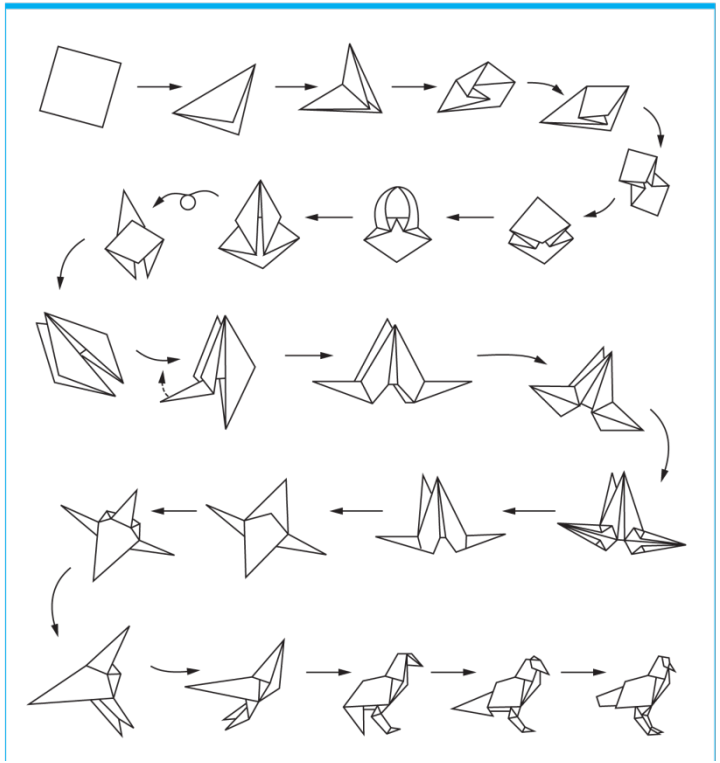
Bir algoritma soyuttur ve gösteriminden ayrıdır. Tek bir algoritma çok çeşitli şekillerde gösterilebilir.

ALGORİTMA GÖSTERİMİ

Temel Öğeler (Primitifler)

Bilgisayar bilimi bu problemlere iyi tanımlanmış bir dizi yapı taşı oluşturarak yaklaşır. Böyle bir yapı taşına **temel öğe** (primitif, ilkel) denir.

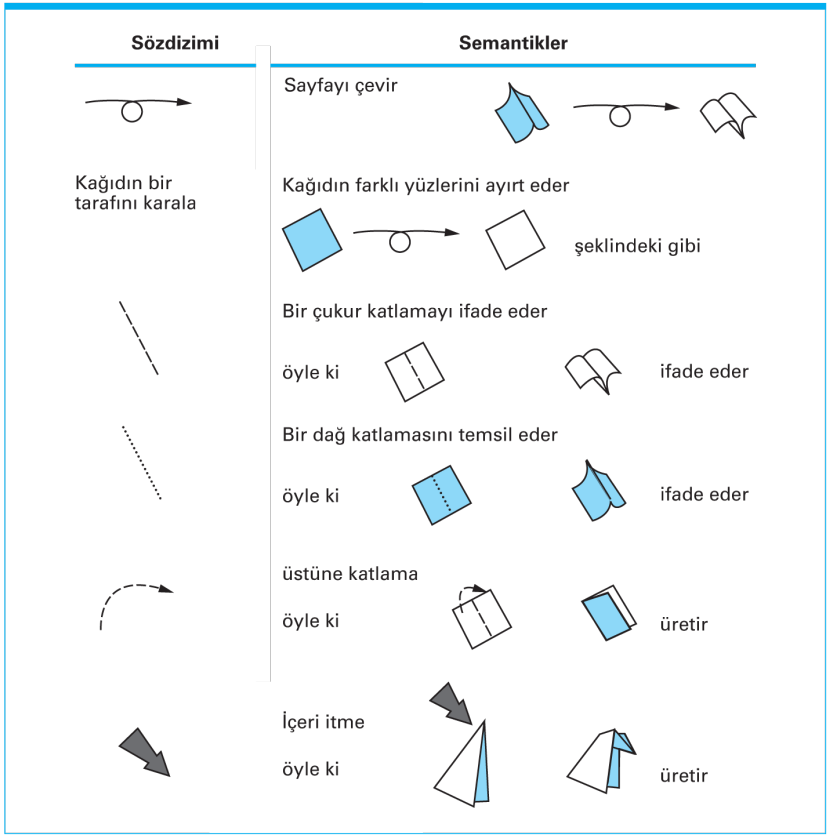
Şekil 5.2 Kare bir kağıttan bir kuş katlamak



Sözde Kod

Genel olarak, bir **sözde kod** düşüncelerin algoritma geliştirme sürecinde gayri resmi olarak ifade edilebildiği bir işaret ve rakamlarla gösterme sistemidir.

Şekil 5.3 Origaminin temel öğeleri



ALGORİTMA KEŞFİ

Bir programın geliştirilmesi iki faaliyetten oluşur altta yatan algoritmanın keşfi ve bir program olarak bu algoritmayı gösterme.

Problem Çözme Becerisi

Algoritma keşfi ve genel problem çözme süreci arasındaki yakın ilişki bilgisayar bilimcilerinin daha iyi problem çözme teknikleri arayışında diğer disiplinlere katılmalarına neden olmuştur.

1945 'te matematikçi G.Polya tarafından sunulan problem çözme aşamaları,
Aşama 1. Problemi anlamak.
Aşama 2. Problemi çözmek için bir plan kurmak.
Aşama 3. Planı uygulamak.
Aşama 4. Çözümün doğruluğunu ve diğer problemleri çözmede bir araç olarak potansiyelini değerlendirmek.

Şekil 5.5 Olasılıkların analizi

a. Çarpımı 36 olan üçlüler		**b.** (a) kısmındaki üçlülerin toplamları	
(1,1,36)	(1,6,6)	1 + 1 + 36 = 38	1 + 6 + 6 = 13
(1,2,18)	(2,2,9)	1 + 2 + 18 = 21	2 + 2 + 9 = 13
(1,3,12)	(2,3,6)	1 + 3 + 12 = 16	2 + 3 + 6 = 11
(1,4,9)	(3,3,4)	1 + 4 + 9 = 14	3 + 3 + 4 = 10

İlk Adımı Atma

Basitçe ifade ile "ilk adımı atma". Örneğin, aşağıdaki basit probemi ele alalım:

A,B,C ve D bir yarış koşusundan önce aşağıdaki tahminleri yaptılar:
A, B'nin kazanacağını tahmin etti.
B, D'nin sonuncu olacağını tahmin etti.
C, A'nın üçüncü olacağını tahmin etti.
D, A'nın tahmininin doğru olacağını tahmin etti.
Bu tahminlerden sadece biri doğruydu ve bu kazanan tarafından yapılan tahmindi. A,B,C ve D hangi sırada yarışı bitirmişlerdir?

İTERATİF YAPILAR

Şimdi amacımız algoritmik süreçlerin açıklanmasında kullanılan bazı tekrarlı yapıları çalışmaktır.

Ardışık Arama Algoritması

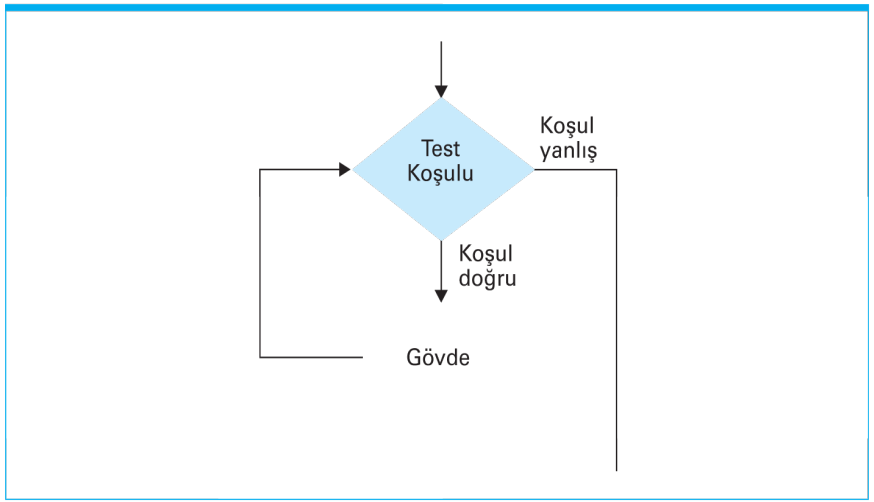
Şekil 5.6 Ardışık arama algoritmasının sözde kodu

```
def Arama(Liste, HedefDeger):
  if (Liste boş):
    Başarısız bir arama bildir.
  else:
    Listedeki ilk elemanı TestEleman olarak seç.
    while (HedefDeğer> TestEleman ve ele alınacak elemanlar var):
      Listede sonraki elemanı TestEleman olarak seç.
    if (HedefDeger == TestEleman):
      Başarılı bir arama bildir.
    else:
      Başarısız bir arama bildir.
```

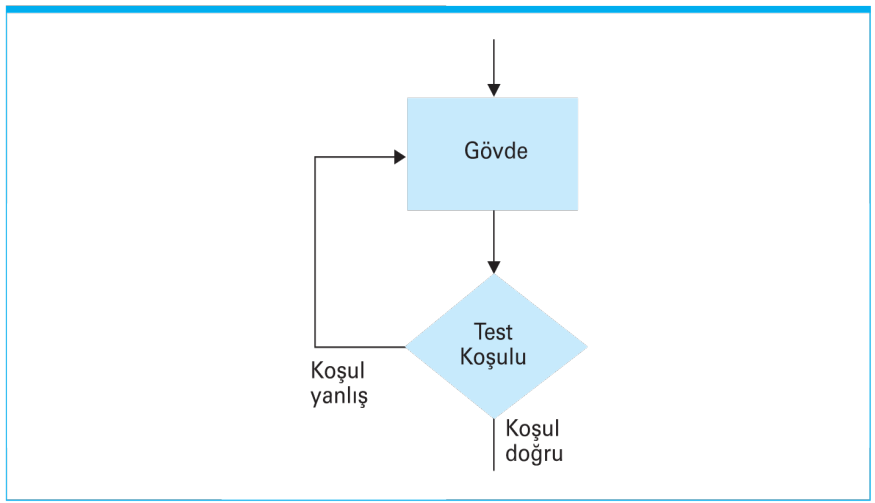
Döngü Kontrolü

Bir talimatın veya bir talimatlar dizisinin tekrarlı kullanımı önemli bir algoritmik kavramdır. Böyle yinelemeyi gerçekleştirmenin bir yolu **döngü** olarak bilinen tekrarlı yapıdır. While Döngüsü , Repeat Döngüsü.

Şekil 5.8 while döngü yapısı



Şekil 5.9 repeat döngü yapısı

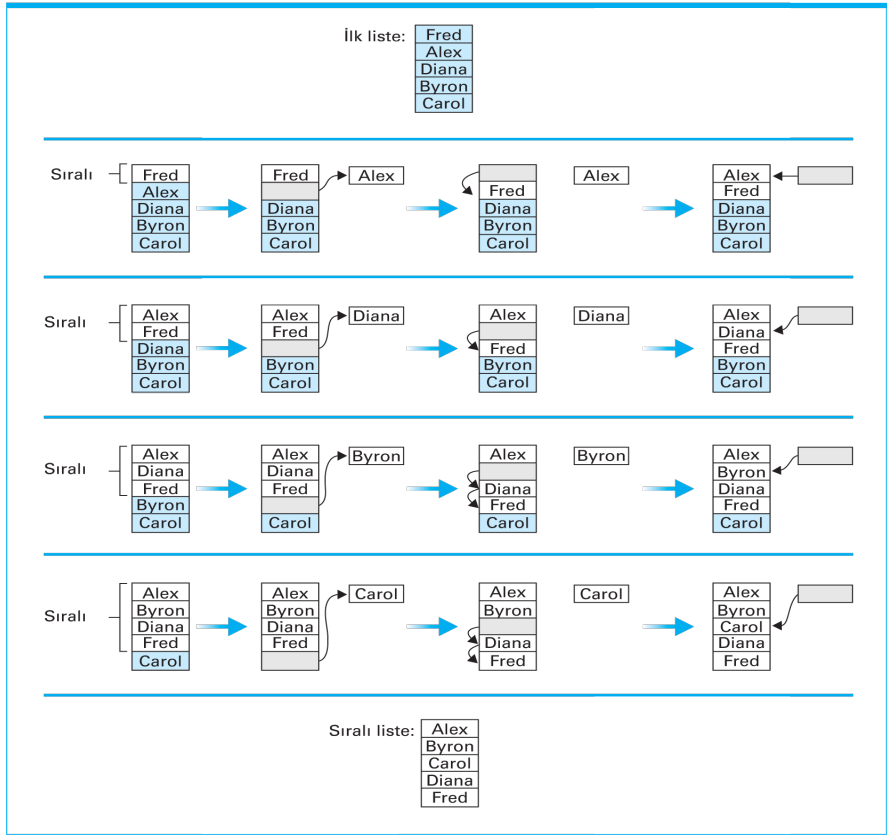


Eklemeli Sıralama Algoritması

İteratif yapıların kullanımına ek bir örnek olarak, bir isimler listesini alfabetik sırada sıralama problemini ele alalım.

Fakat ilerlemeden önce çalışacağımız kısıtları belirlemeliyiz. Basit ifadeyle, amacımız listeyi "kendi içinde" sıralamaktır.

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



Şekil 5.11 Eklemeli sıralama algoritmasının sözde kodu

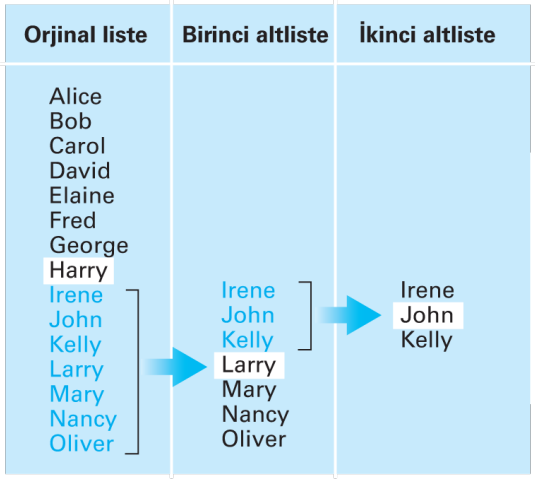
```
def Sırala (Liste):
  N = 2
  while (N in değeri listenin uzunluğunu aşmıyor):
    Listede N. elemanı pivot eleman olarak seç.
    Pivot elemanı Listede bir boşluk bırakarak geçici
      bir konuma taşı .
    while (boşluğun üstünde bir isim var ve
        bu isim pivottan büyük):
      boşluğun üstündeki ismi boşluğa taşı.
    Pivot elemanı Listedeki boşluğa taşı.
    N = N + 1
```

ÖZYİNELEMELİ YAPILAR

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.

İkili Arama Algoritması

Şekil 5.12 Bir listede John elemanını aramak için stratejimizin uygulanması



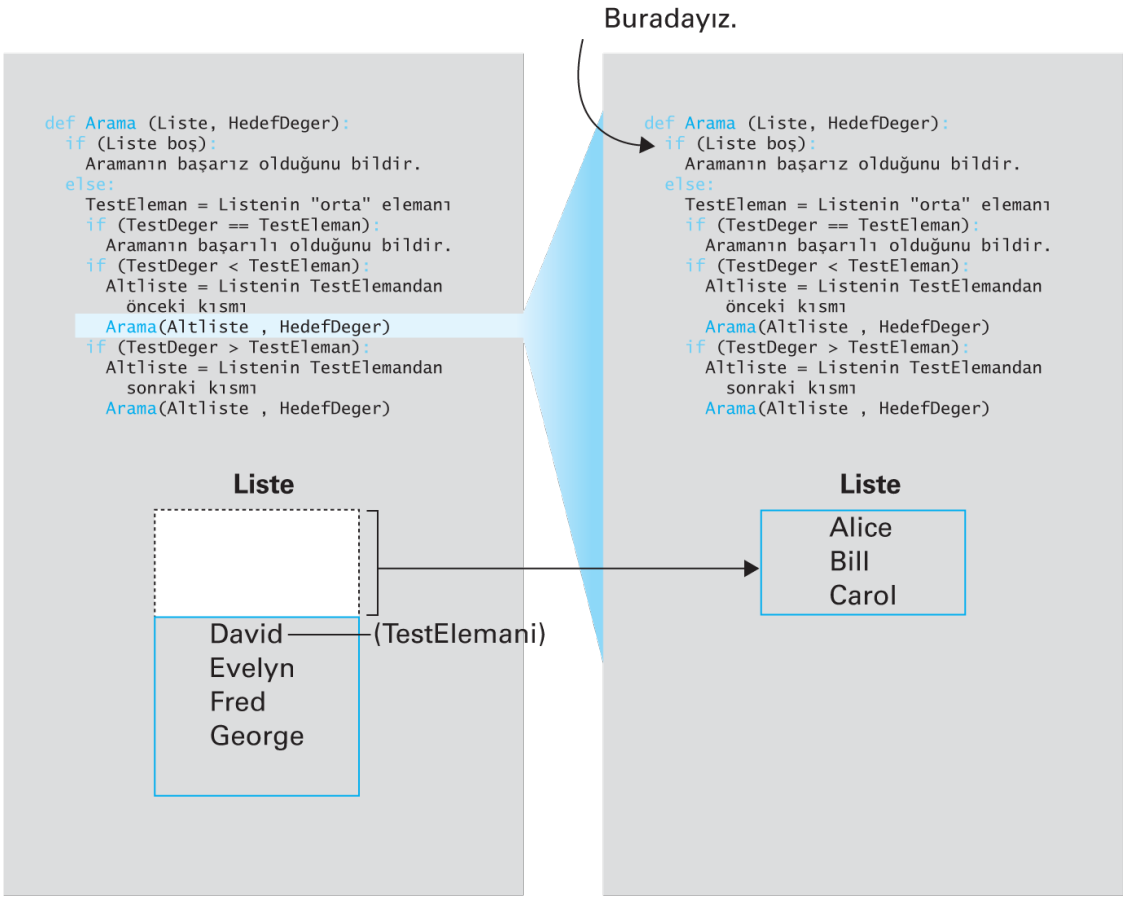
Şekil 5.13 İkili arama tekniğinin ilk taslağı

```
if (Liste boş):
  Aramanın başarız olduğunu bildir.
else:
  TestEleman = Listenin "orta" elemanı
  if (TestDeger == TestEleman):
    Aramanın başarılı olduğunu bildir.
  if (TestDeger < TestEleman):
    HedefDeger için Listenin TestElemandan  önceki kısmına
      Arama() ve bu aramanın sonucunu bildir.
  if (TestDeger > TestEleman):
    HedefDeger için Listenin TestElemandan  sonraki kısmına
      Arama() ve bu aramanın sonucunu bildir.
```

Şekil 5.14 İkili arama algoritmasının sözde kodu

```
def Arama (Liste, HedefDeger):
  if (Liste boş):
    Aramanın başarız olduğunu bildir.
  else:
    TestEleman = Listenin "orta" elemanı
    if (TestDeger == TestEleman):
      Aramanın başarılı olduğunu bildir.
    if (TestDeger < TestEleman):
      Altliste = Listenin TestElemandan
        önceki kısmı
      Arama(Altliste, HedefDeger)
    if (TestDeger > TestEleman):
      Altliste = Listenin TestElemandan
        sonraki kısmı
      Arama(Altliste , HedefDeger)
```

Şekil 5.15 Özyinelemeli Arama

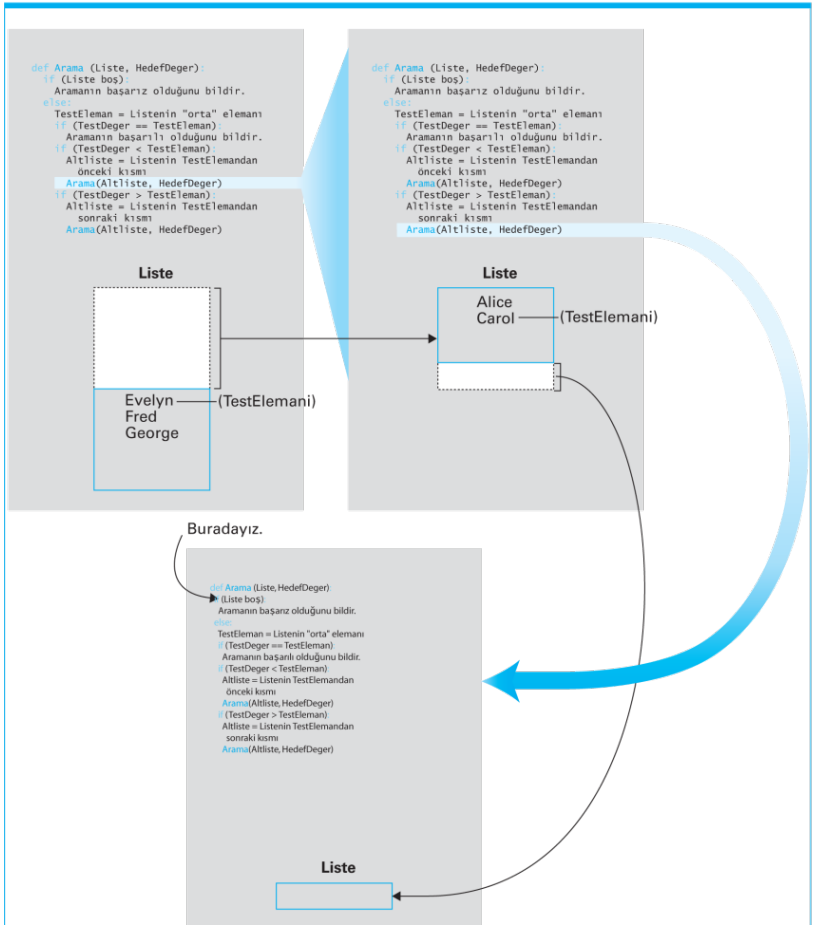


Özyineleme Kontrolü

- Ardışık arama tekrarın dairesel bir formunu içerirken ikili arama tekrarın her aşamasını önceki aşamanın bir alt görevi olarak yürütür.

- Bu teknik **özyineleme** olarak bilinir.

Şekil 5.17 İkinci Özyinelemeli Arama, İkinci anlık görüntü

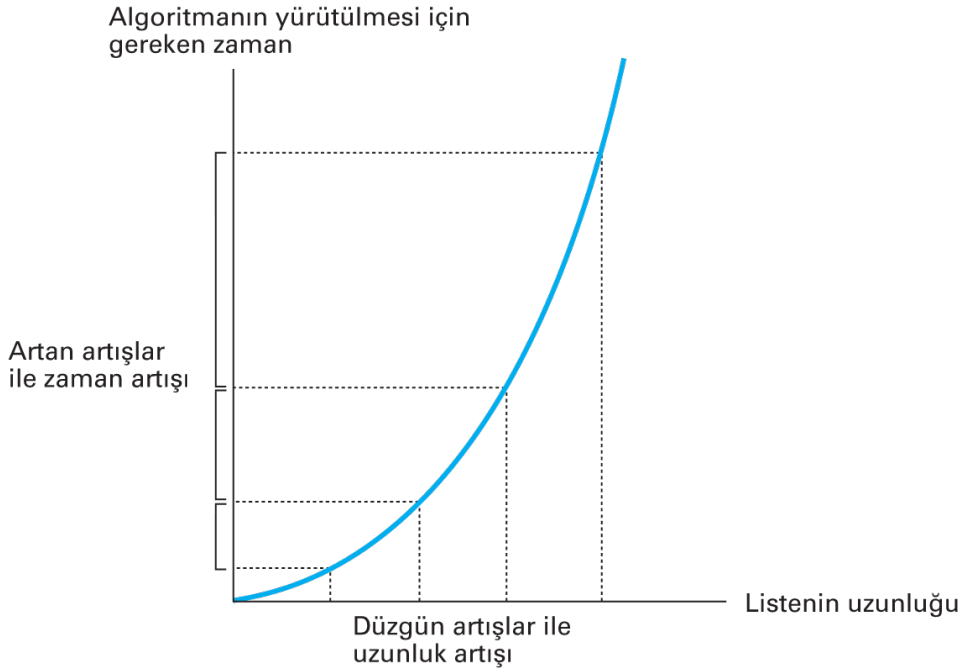


VERİMLİLİK VE DOĞRULUK

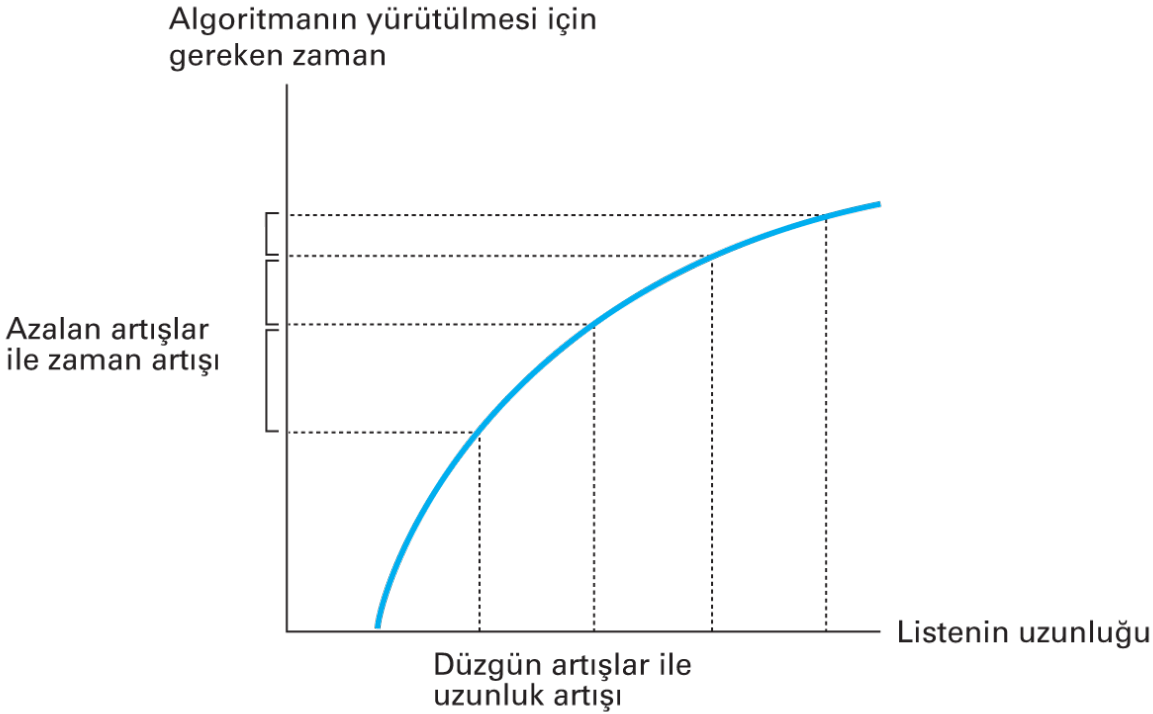
Algoritma Verimliliği

Günümüz makineleri her saniye milyonlarca veya milyarlarca komut çalıştırma kabiliyetine sahip olmasına rağmen, verimlilik algoritma tasarımında önemli meselelerden biri olmaya devam etmektedir. Genellikle verimli ve verimsiz algoritmalar arasındaki seçim bir probleme elverişli bir çözüm ve elverişsiz bir çözüm arasındaki farkı yaratabilir.

Şekil 5.19 Eklemeli Sıralama Algoritmasının en kötü-durum analizi grafiği



Şekil 5.20 İkili arama algoritmasının en kötü-durum analizi grafiği

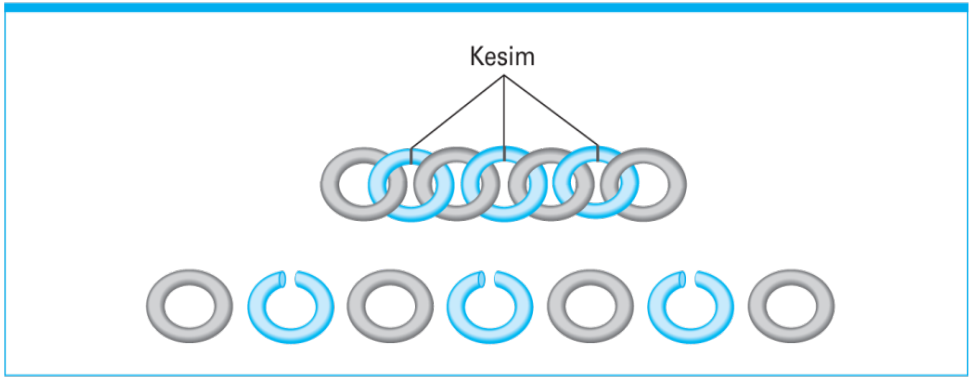


Yazılım Doğrulama

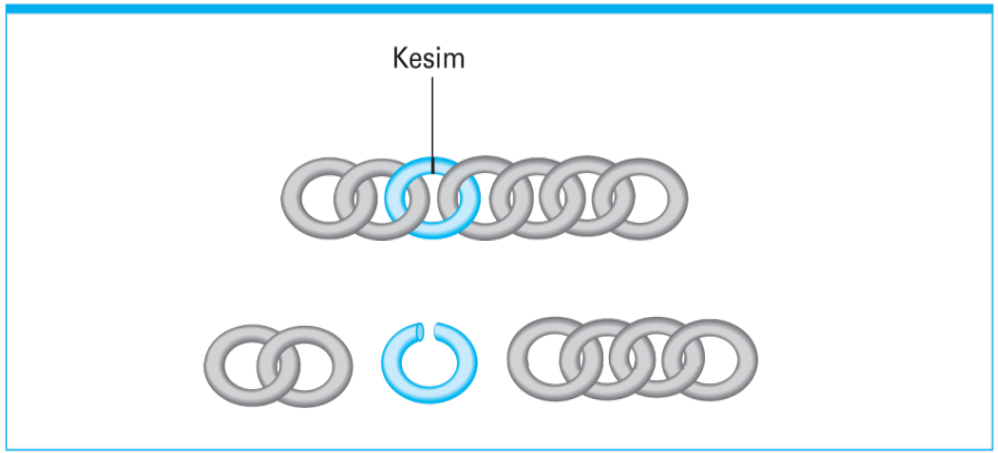
Yedi halkalı bir altın zinciri olan bir gezgin yedi gece izole edilmiş bir otelde kalmalıdır. Her gecenin kirası zincirden bir halkadır.

Konaklamaya peşin ödeme yapmadan gezginin her sabah bir zincir halkasını otele ödeyebilsin diye kesilmesi gereken en az halka sayısı kaçtır?

Şekil 5.21 Sadece üç kesim kullanarak zinciri ayırma



Şekil 5.22 Sadece bir kesim ile problemi çözme



Şekil 5.23 Tipik bir while yapısıyla ilişkili iddialar