4.3.3.2 Çakışma (Collusion) Çözümleme

- → çakışma: aynı slota iki elemanın talip olması
- → ikincisini tabloya nasıl koyacağız?
- → bunun için sunacağımız sistematik yönteme çakışma çözümleme deriz
- → çırpı mükemmelse, çakışma yoktur
- → bu ise nadir bir durumdur, genelde çözümlemeye ihtiyaç duyarız

açık adresleme

→ çırpı tablosundaki boş slotları kullanmak

Yöntem

1. çırpı işlevinden geçir
2. çakışma varsa **doğrusal** aramayla ilk boş yere koy

   → *tablodan sorgularken bunları nasıl bulacağız?*

3. gerekirse başa sar

açık adresleme

→ açık adresleme

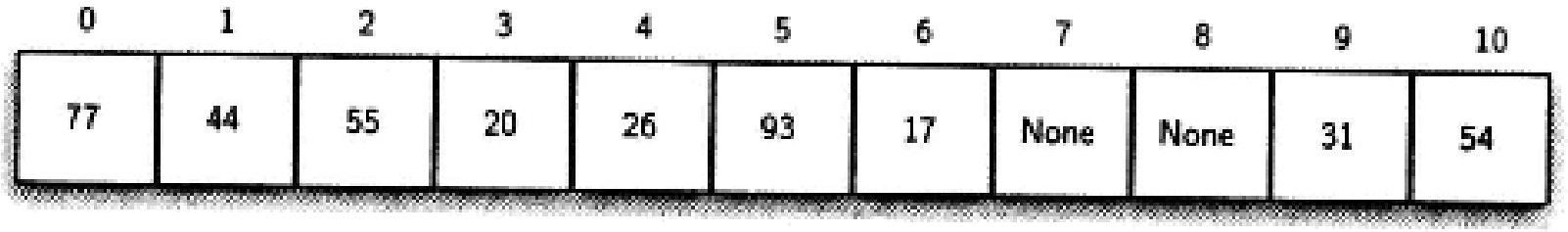

Figure 4.9: Collision Resolution with Linear Probing

senaryolar

→ 93 nerede? 93 % 11 = 5, slot5: 93 $\Longrightarrow$ True

→ 20 nerede? 20 % 11 = 9, slot9: 31, doğrusal arama, slot3: 20 $\Longrightarrow$ True

kümelenme

→ açık adresleme kümelenmeye sebep olur

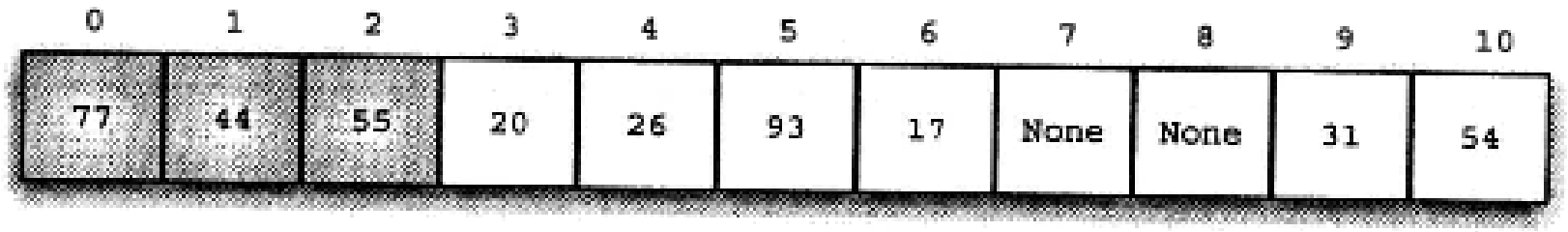

Figure 4.10: A Cluster of Items for Slot 0

atlatma tekniği

→ sonraki ilk boşluğa yerleştirme yerine,

→ n (örneğin 3) atla,

→ çakışanı yerleştir

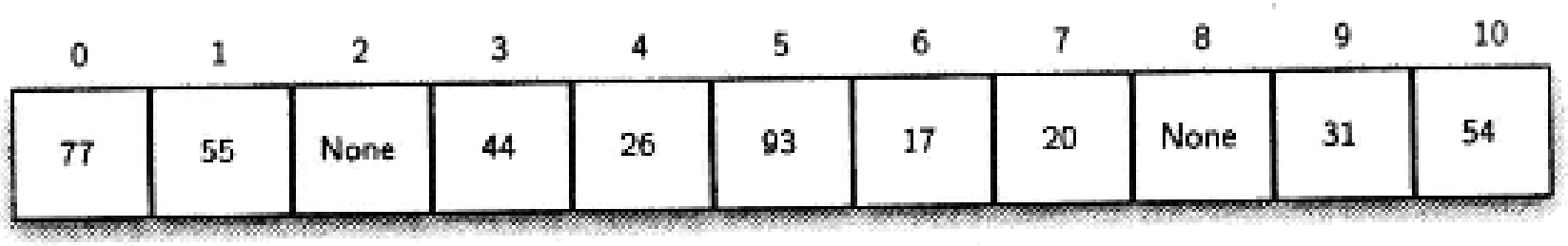

Figure 4.11: Collision Resolution Using "Plus 3"

yeniden çırp (rehashing)

→ yeni çırpı değeri = yeniden_çırp(eski çırpı değeri)

```
rehash(pos) = (pos + 1) % sizeoftable
```

→ atlatarak

```
rehash(pos) = (pos + 3) % sizeoftable
```

→ genelleştirilmiş

```
rehash(pos) = (pos + skip) % sizeoftable
```

quadratic problama

→ skip değerini sabit tutmak yerine 1,3,5,7,... artımlı yapmak

→ ilki h ise, sonraki h+1, h+4, ...

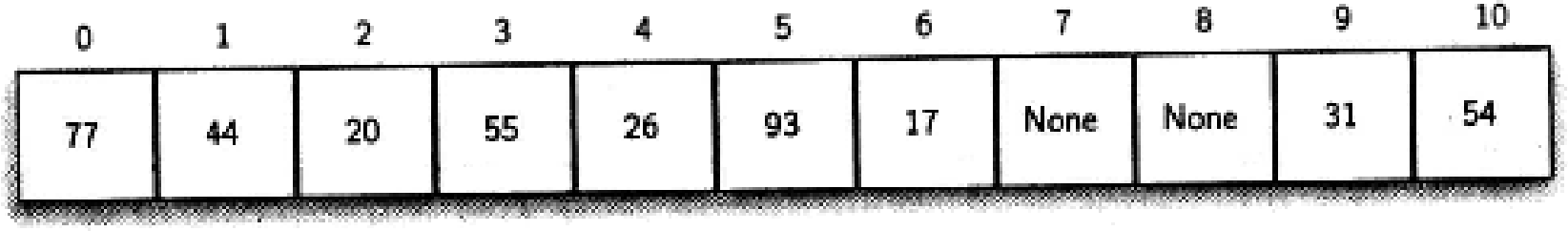

Figure 4.12: Collision Resolution with Quadratic Probing

zincirleme

Yöntem

1. Çırp
2. Slot doluysa, zincirin sonuna ekle

zincirleme

Yöntem

1. Çırp
2. Slot doluysa, zincirin sonuna ekle

Ararken

1. çırp
2. slottaki zinciri tara

zincirleme

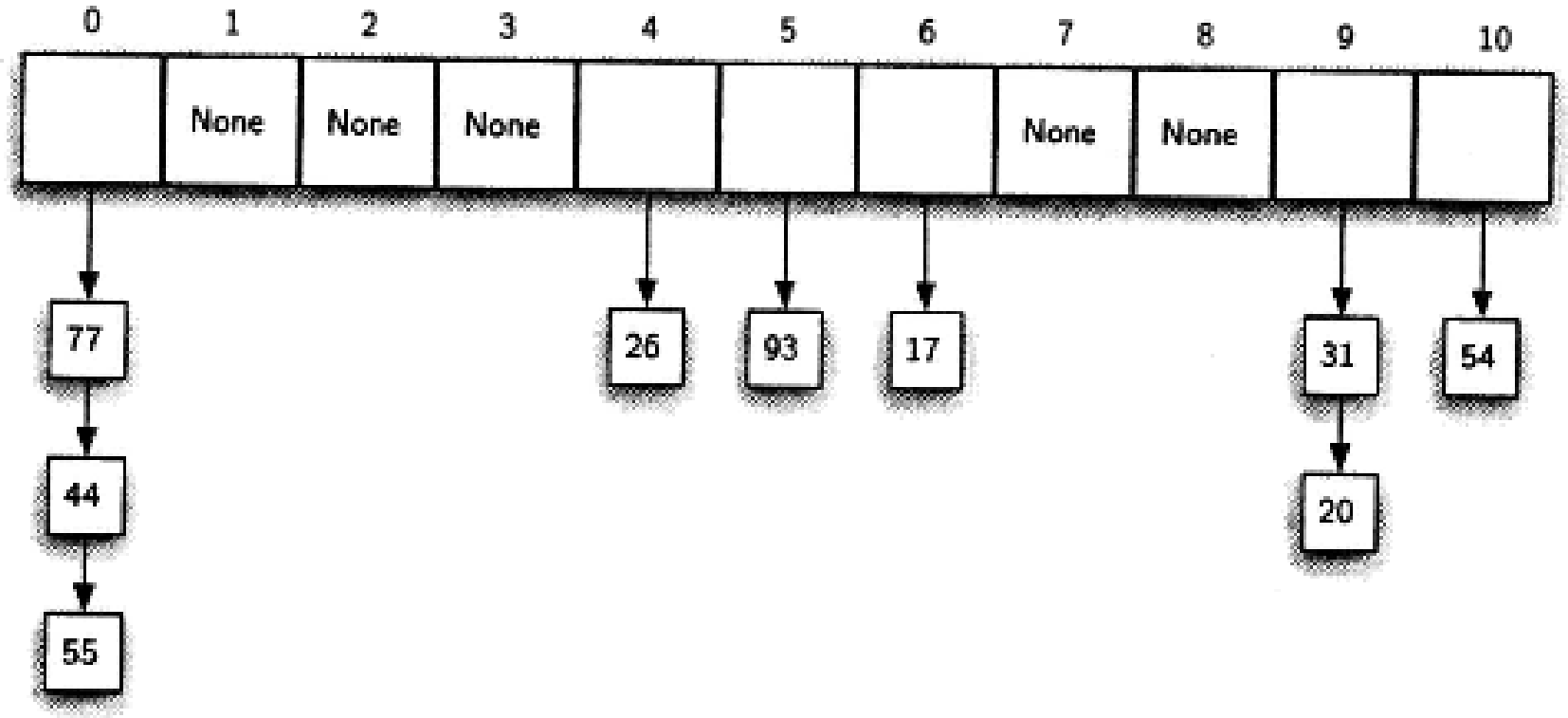

Figure 4.13: Collision Resolution with Chaining

→ bağlı listelerle nasıl temsil ederiz?