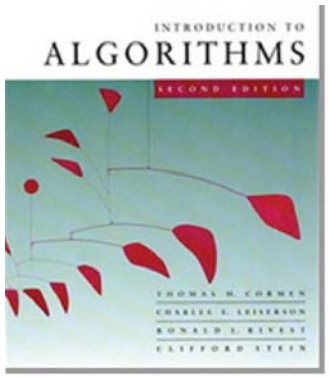
# *Algoritmalara Giriş*

## 6.046J/18.401J

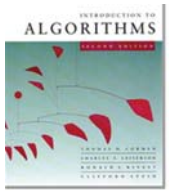



## DERS 11

**Veri Yapılarının Genişletilmesi**

- Dinamik Seviye İstatistikleri
- Metodoloji
- Aralık Ağaçları

**Prof. Charles E. Leiserson**



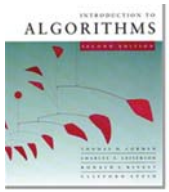

# Dinamik Seviye İstatistikleri

OS-SEÇ*(i,S)* : dinamik ***S*** kümesindeki
***i***' ninci elemanı döndürür.

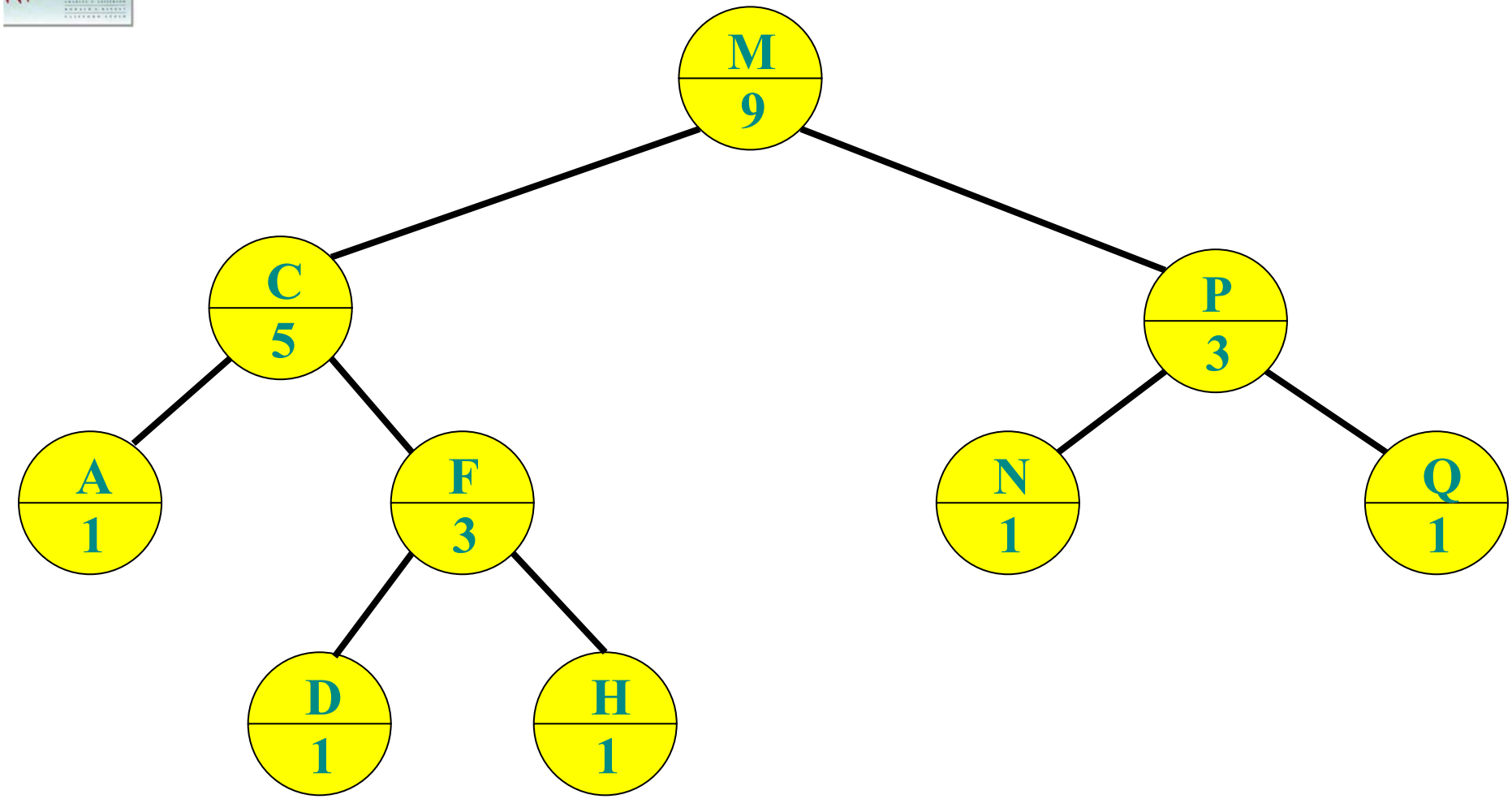
OS-RÜTBE *(i,S)* : **S** kümesinin sıralanmış elemanları arasında $x \in S$'in rankını döndürür.

**Fikir :** S kümesi için kırmızı – siyah ağacı kullanın, ama altağaç büyüklüklerini düğümlerde saklayın.

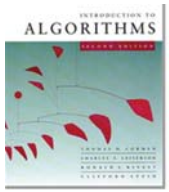
**Düğümlerin Simgelemi :**




# OS-Ağacı Örneği




*büyüklük[x] = büyüklük[sol[x]] + büyüklük[sağ[x]] + 1*




# Seçim

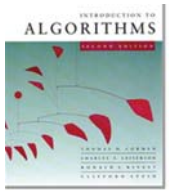
**Gerçekleştirme Hilesi :** BOŞLUK için *büyüklük*[BOŞLUK] = 0 gibi bir **gözcü** (kukla kayıt) kullanın.

OS-SEÇ***(x,i)*** ▷ ***x*** kökenli alt ağacın ***i***' ninci küçük elemanı.

***k*←*büyüklük*[*sol*[*x*]] + 1** ▷ **$k$ = rütbe($x$)**

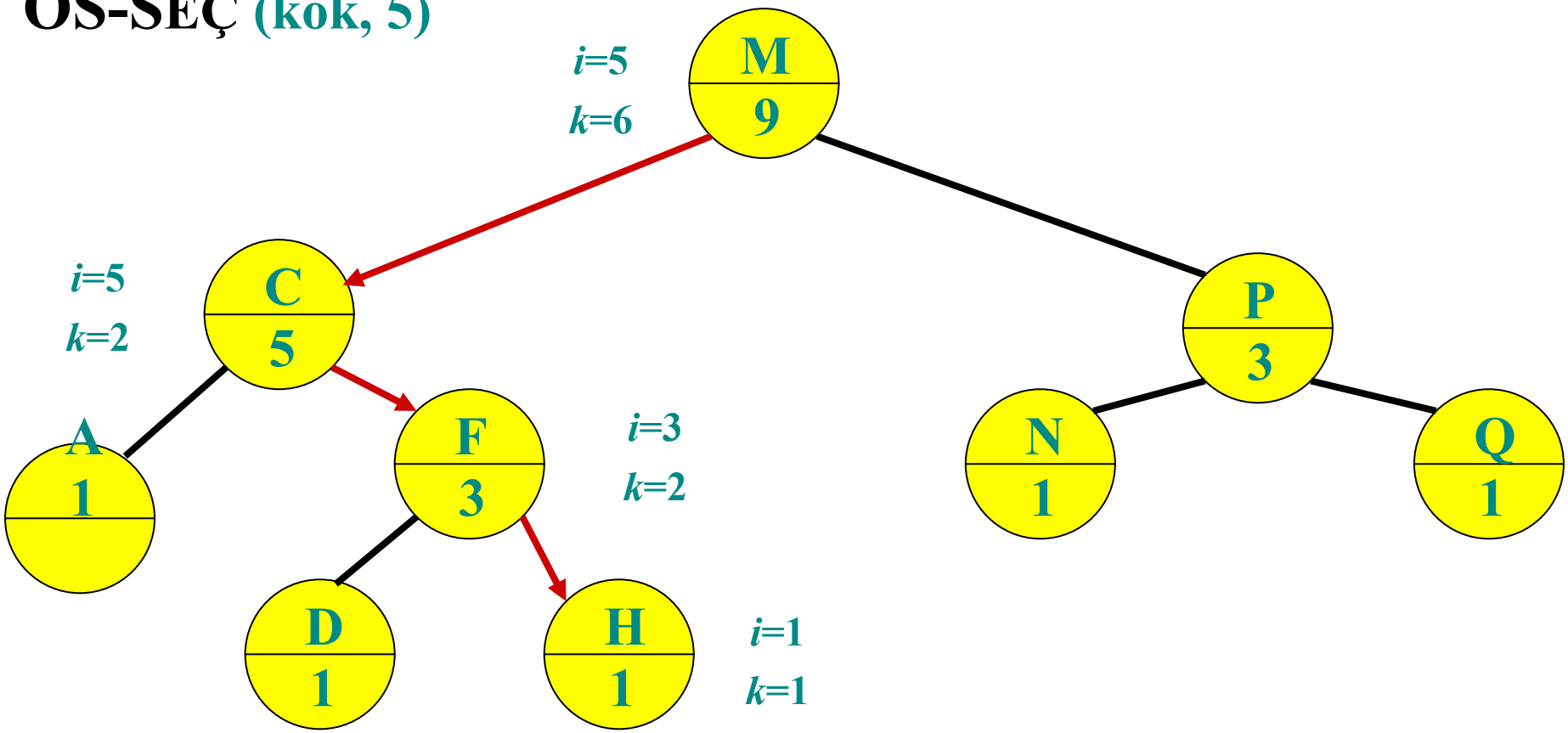
**eğer *i*= *k* ise *x*'**i **döndür.**
**eğer *i*< *k* ise** OS-SEÇ(*sol*[*x*], *i*)'i **döndür.**
**diğer durumlarda** OS-SEÇ(*sağ*[*x*], *i* –*k*)'i **döndür.**

**(Kitaptaki OS-Rütbe)**

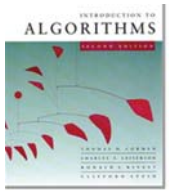



# Örnek

**OS-SEÇ (kök, 5)**




***Çalışma Zamanı*** = $O(h) = O(\lg n)$ (kırmızı – siyah ağaçlar için)



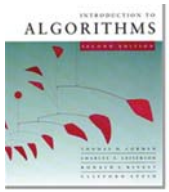

# Veri Yapısının Bakımı

**S.** Neden altağaç büyüklükleri yerine düğümlerdeki rankları (rütbeleri) saklamıyoruz?

**C.** Kırmızı – Siyah Ağaç değiştirildiğinde, rankları düzenlemek zor olur.

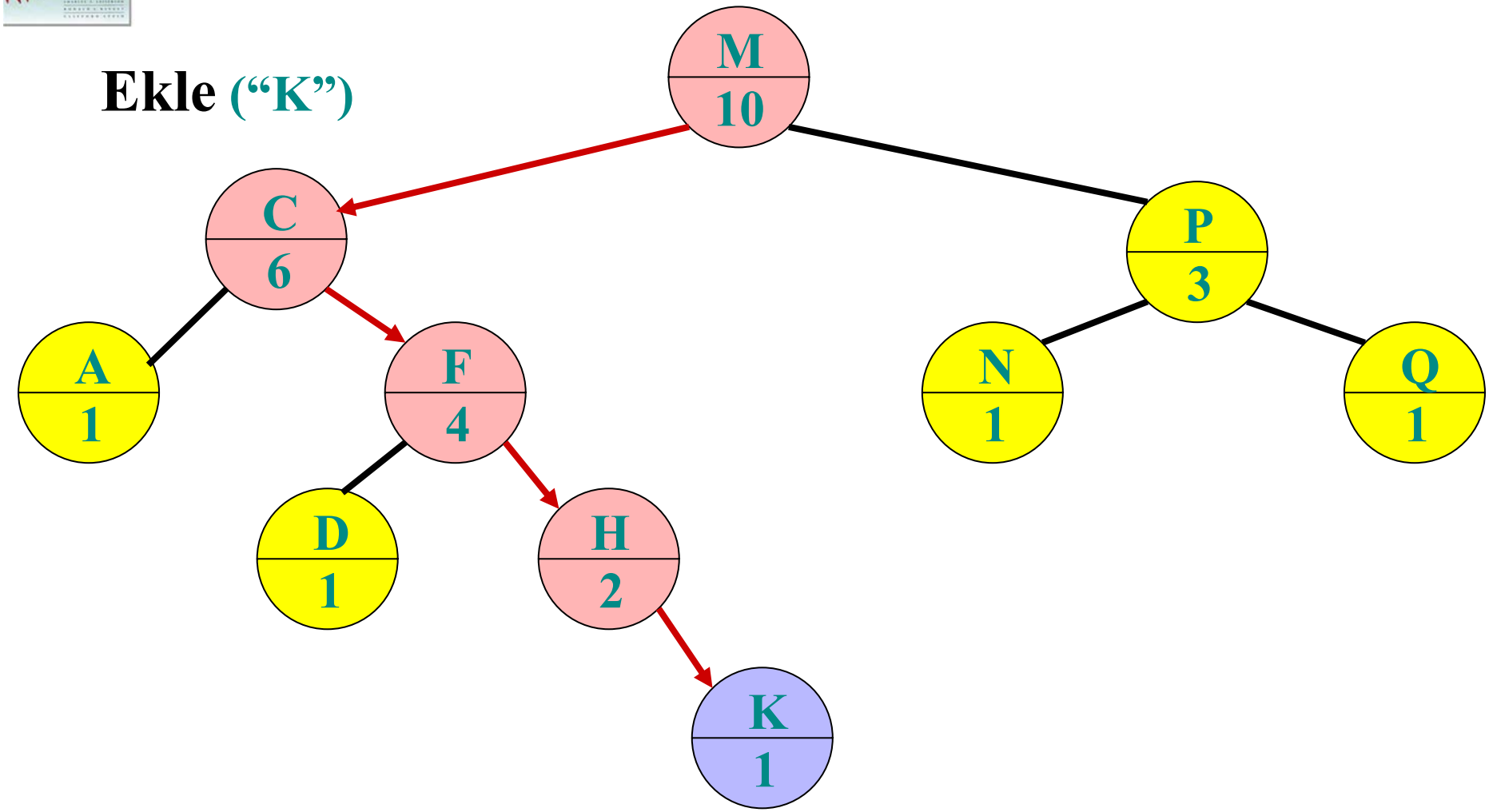
**Düzenleme İşlemleri :** **EKLE** ve **SİL**

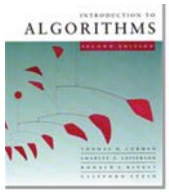
**Strateji :** Ekleme veya silme işlemi sırasında altağaç büyüklüklerini güncelleyin.




# Ekleme Örneği

**Ekle (“K”)**





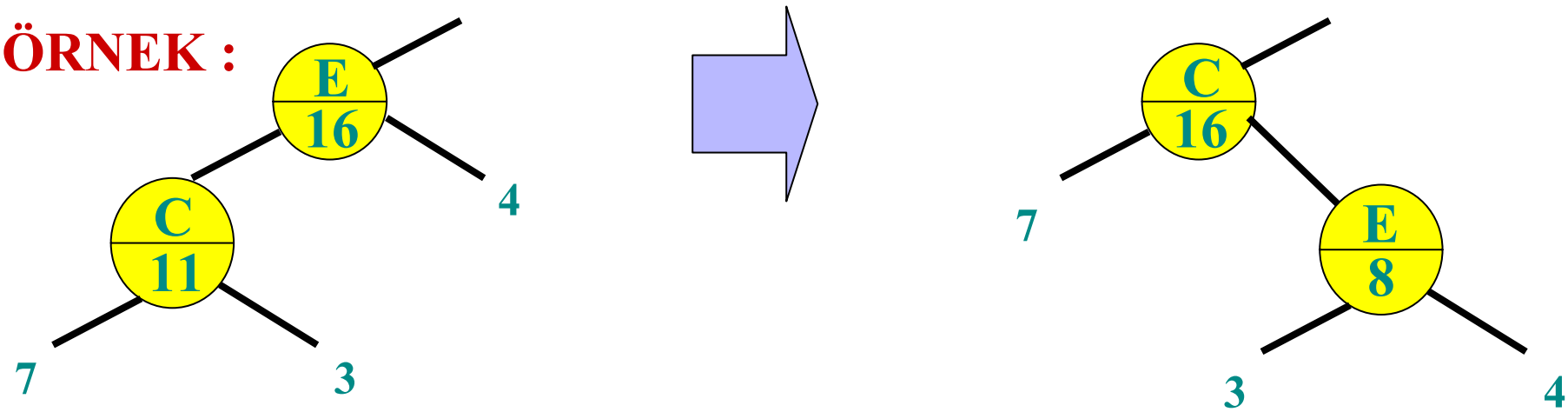

# Yeniden Dengelemeyi Ele Almak

YD-EKLE ve YD-SİL de dengenin sağlanması için kırmızı – siyah ağacı düzenlemeye ihtiyaç duyabilir.

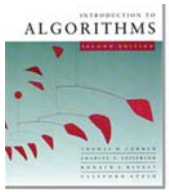
*Yeniden Renklendirmeler:* Altağaç büyüklüklerinde değişime neden olmaz.

*Döndürmeler:* Altağaç büyüklüklerini *O(1)* zamanda düzeltir.

**ÖRNEK :**




∴ YD-EKLE ve YD-SİL hala ***O(lg n)*** zamanda çalışıyor.




# Veri Yapısının Genişletilmesi

**Metodoloji:** (örn. Sı ralı istatistik ağaçları)

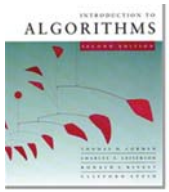
1. Bir veri yapısı seçin. (kırmızı – siyah ağaçları)

2. Veri yapısında saklanacak ek bilgileri belirleyin. (alt ağaç büyüklükleri)

3. Bu bilgilerin düzenleme işlemleri için sağlanabileceğini doğrulayın. (YD-EKLE, YD-SİL – döndürmeleri unutmayın… )

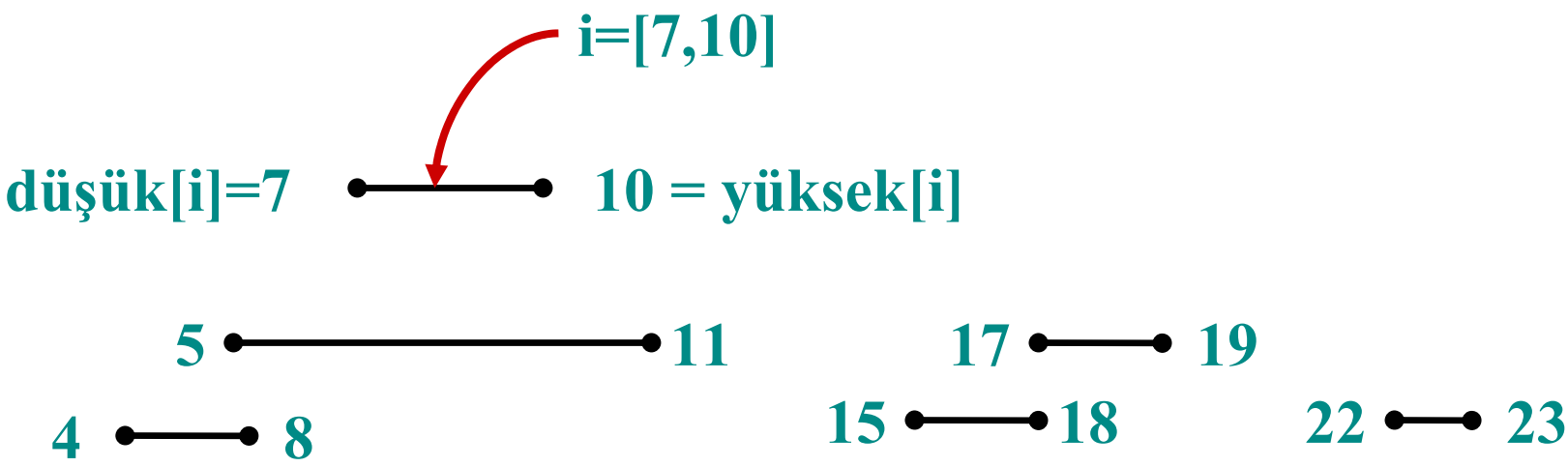
4. Bilgileri kullanan yeni dinamik küme işlemleri geliştirin. (OS-SEÇ, OS-RÜTBE)

Bu basamaklar yol göstericidir, mecburi değil.

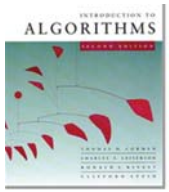



# Aralık Ağaçları

**Amaç:** Zaman aralıkları gibi dinamik aralık kümeleri oluşturmak.




**Sorgu:** Verilen bir *i* sorgu aralığı için, kümede, *i* ile örtüşen bir aralık bulun.



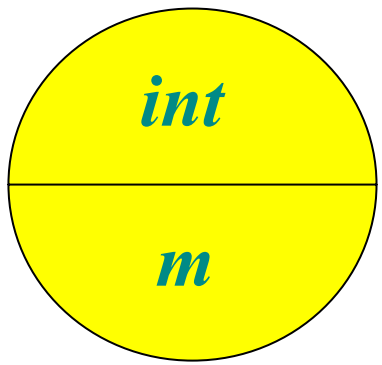

# Takip Eden Metodoloji

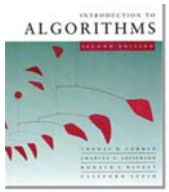
1. Altta yatan bir veri yapısı bulun.

   • Kırmızı-siyah ağaç düşük (sol) son noktasında anahtarlanır.

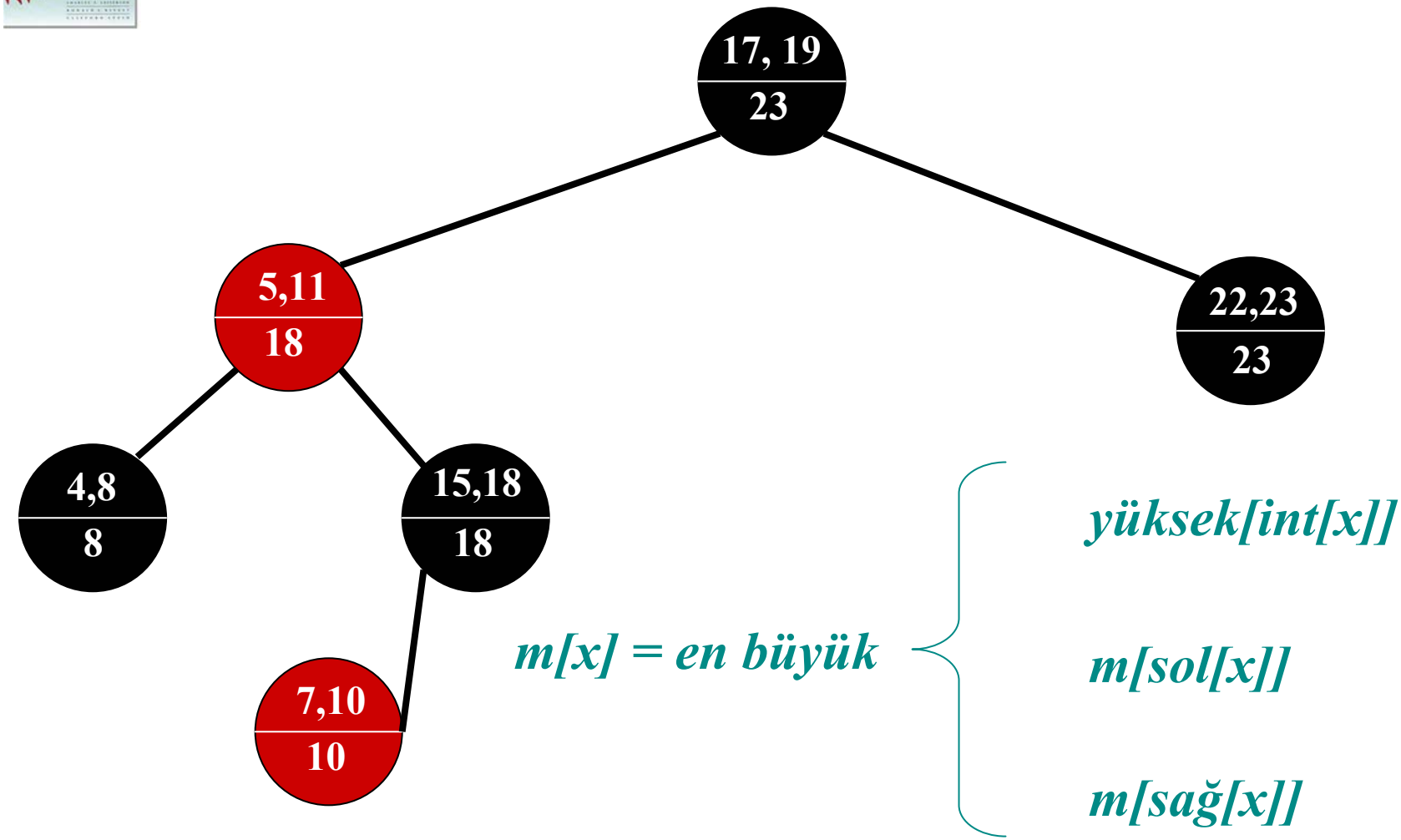
2. Veri yapısında tutulacak ek bilgiyi belirleyin.

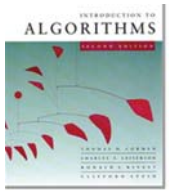
   • Her düğüm *x*'de, *x* köklü altağacın en büyük değeri *m(x)*'i ve anahtara karşılık gelen aralık *int(x)*'i saklayın.





# Aralık Ağacı Örneği

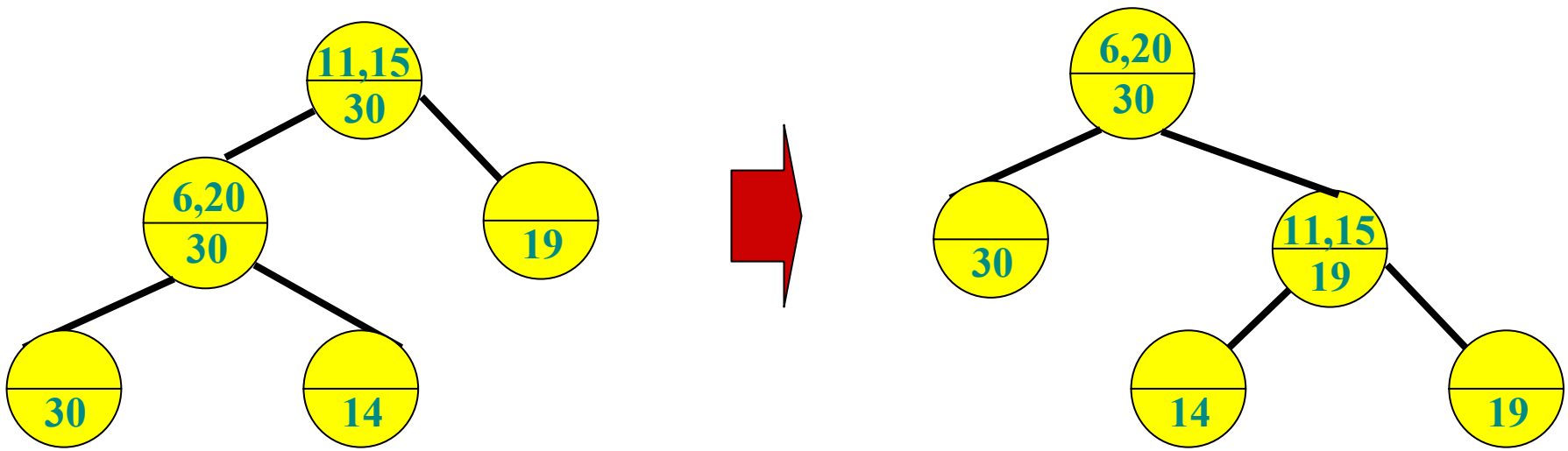




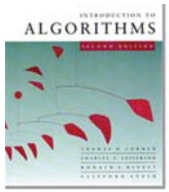

# Düzeltme İşlemleri

3. Bu bilginin düzenleme işlemleri sırasında sağlanabileceğini doğrulayın.

- EKLE : aşağıya doğru yolun üzerindeki *m*' leri onarın.
- Döndürmeler --- Düzeltme = *O(1)* her bir döndürme için:




Toplam EKLEME süresi = *O(lg n)*; SİLME de benzer.




# Yeni İşlemler

4. Bilgiyi kullanan yeni dinamik küme işlemleri geliştirin.

ARALIK-ARAMA*(i)*

$x \leftarrow$ *kök*

**her ne zaman** $x \neq$ BOŞLUK ve *(düşük[i]>yüksek[int[x]]*
veya *düşük[int[x]]>yüksek[i])*

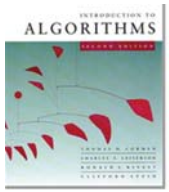
**yap** ▷ *i* ve *int[x]* çakışmasın

**eğer** *sol[x]* $\neq$ BOŞLUK ve *low[i]* $\leq$ *m[sol[x]]*

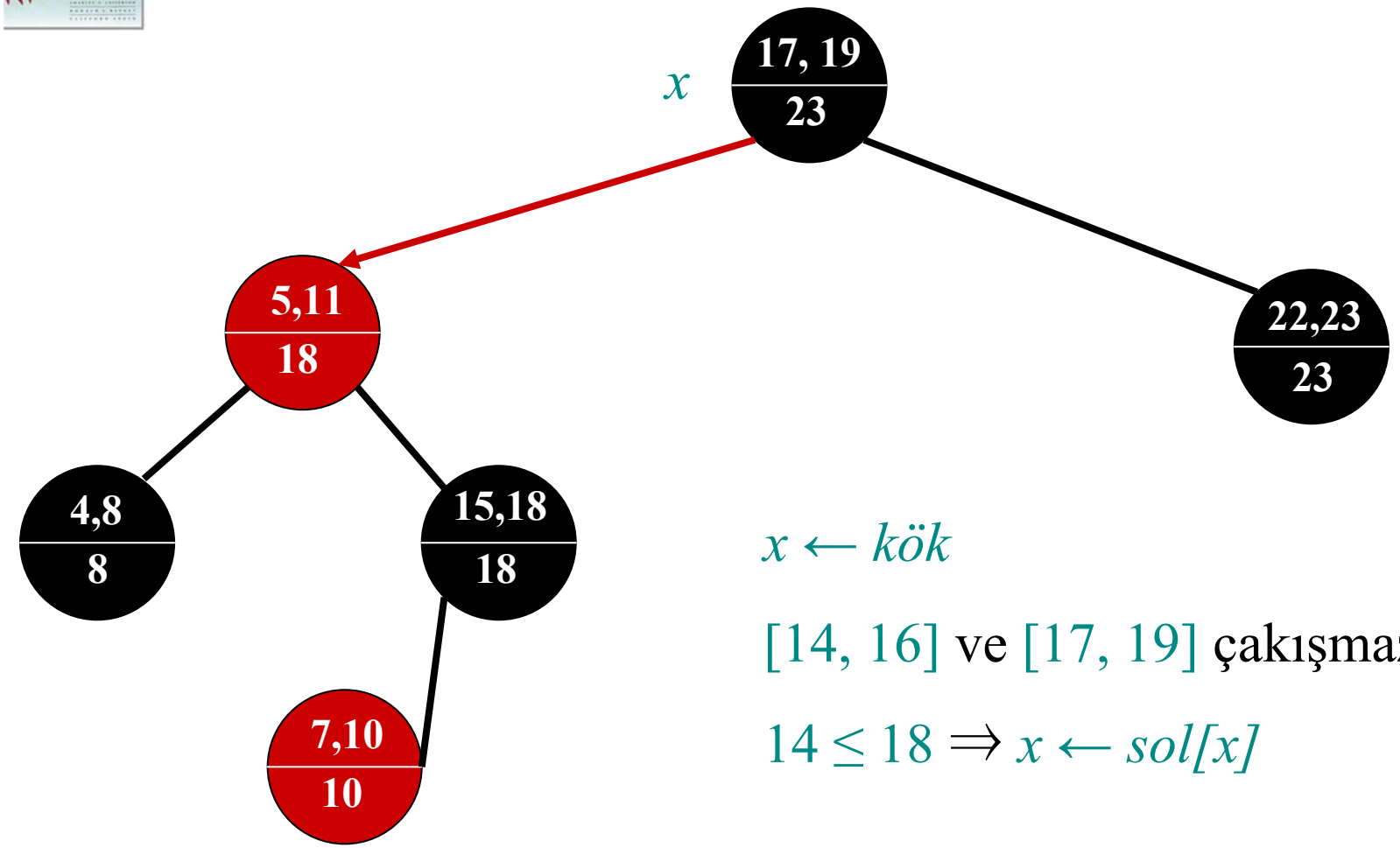
**ise** $x \leftarrow$ *sol[x]*

**değilse** $x \leftarrow$ *sağ[x]*

*x*'i **döndür**.



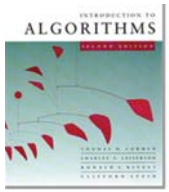

# Örnek 1 : Aralık Arama([14,16])



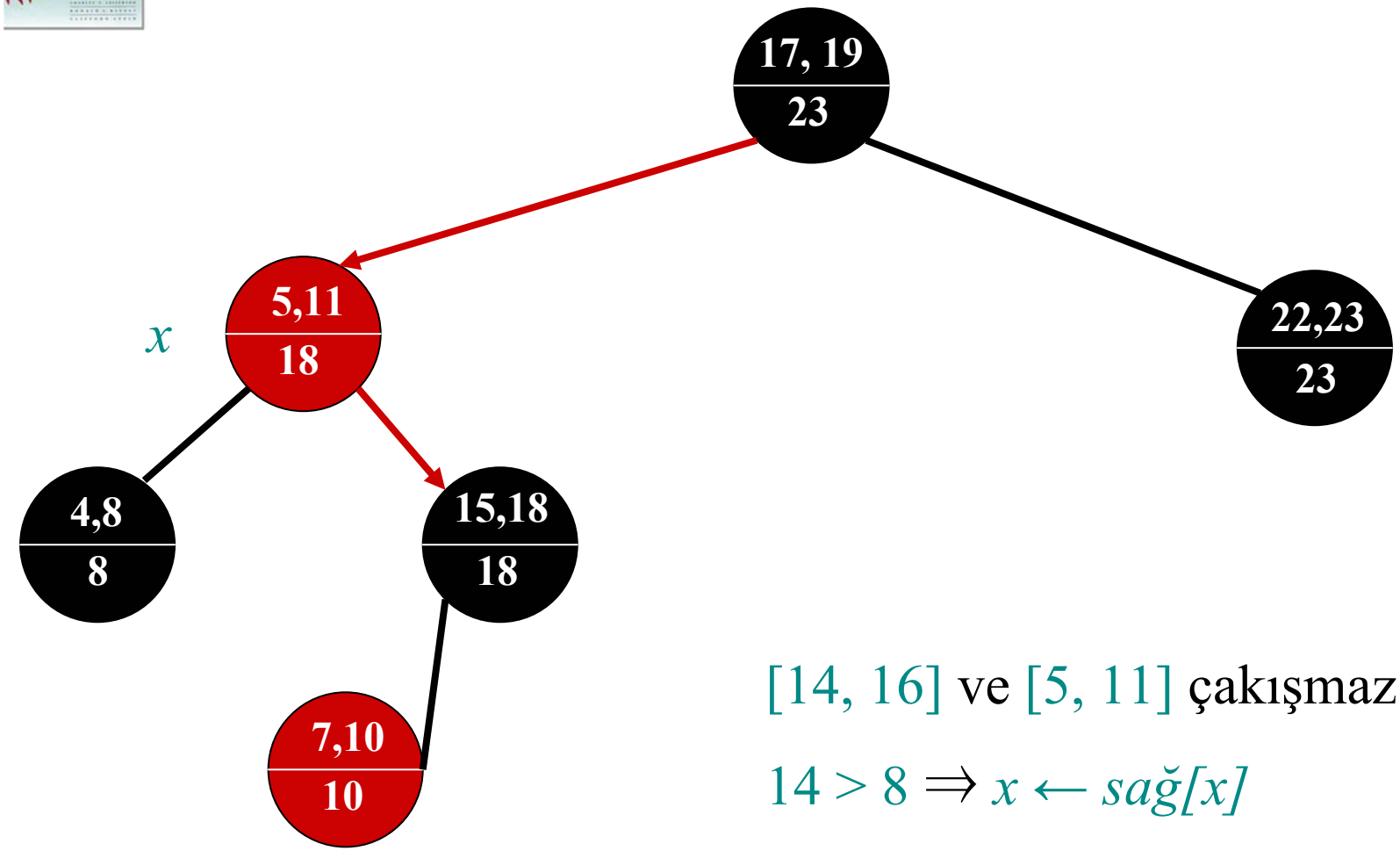

$x \leftarrow kök$

[14, 16] ve [17, 19] çakışmaz.

$14 \leq 18 \Rightarrow x \leftarrow sol[x]$

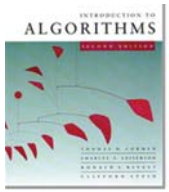



# Örnek 1 : Aralık Arama([14,16])

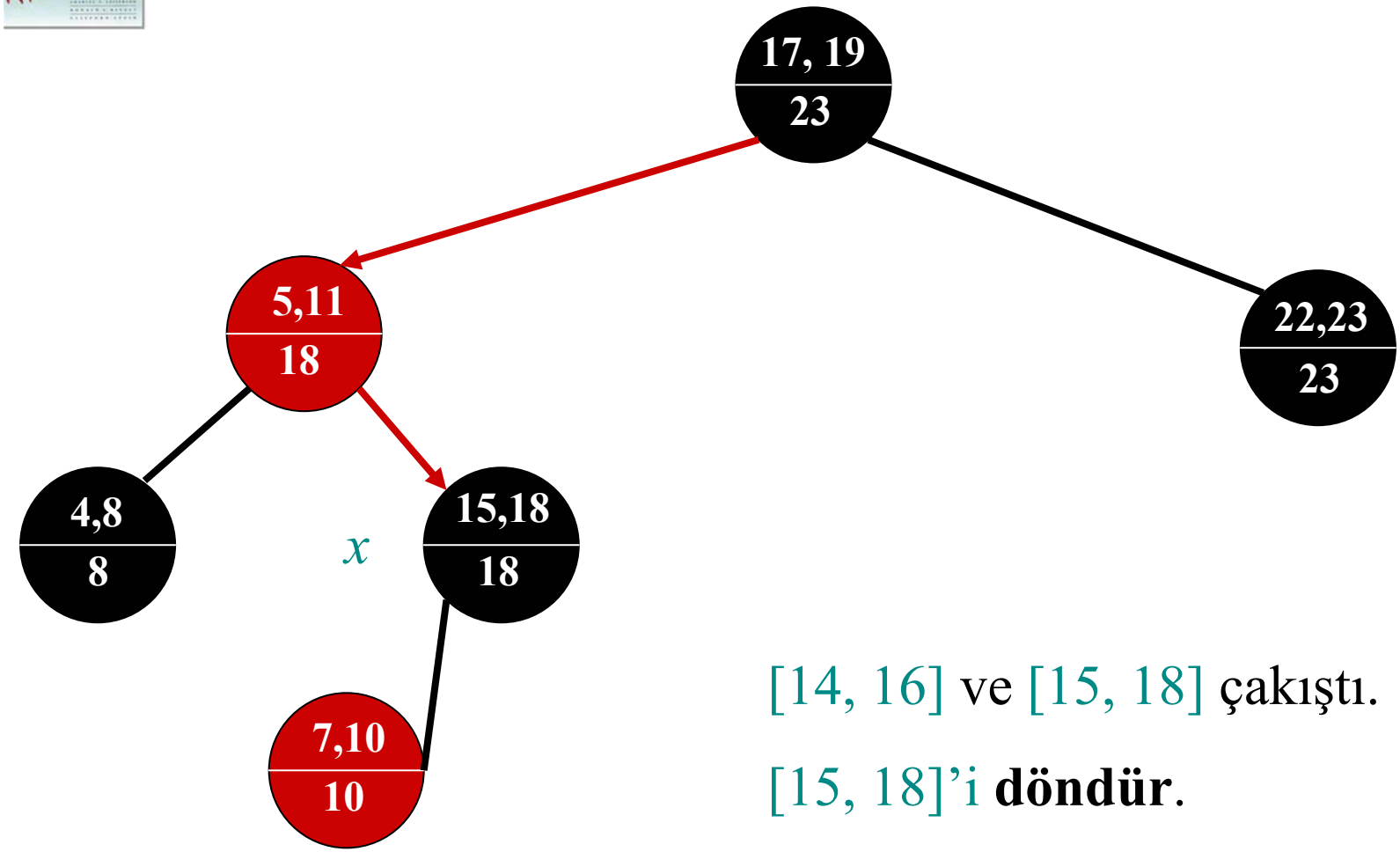



[14, 16] ve [5, 11] çakışmaz.

$14 > 8 \Rightarrow x \leftarrow sağ[x]$

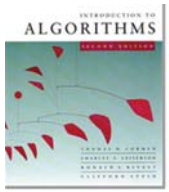



# Örnek 1 : Aralık Arama([14,16])

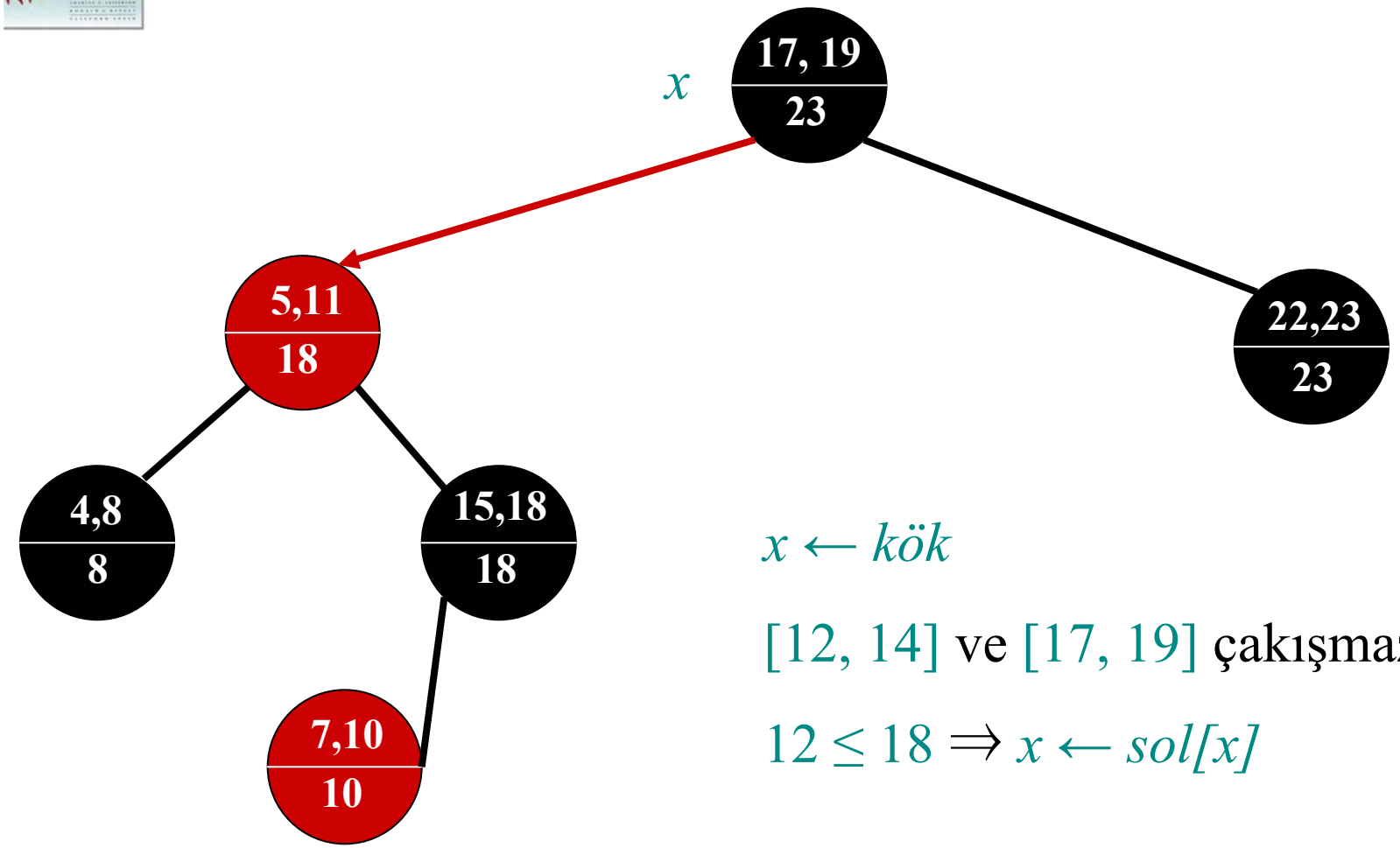



[14, 16] ve [15, 18] çakıştı.

[15, 18]'i **döndür**.



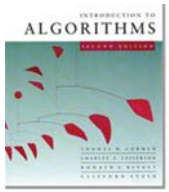

# Örnek 2 : Aralık Arama([12,14])



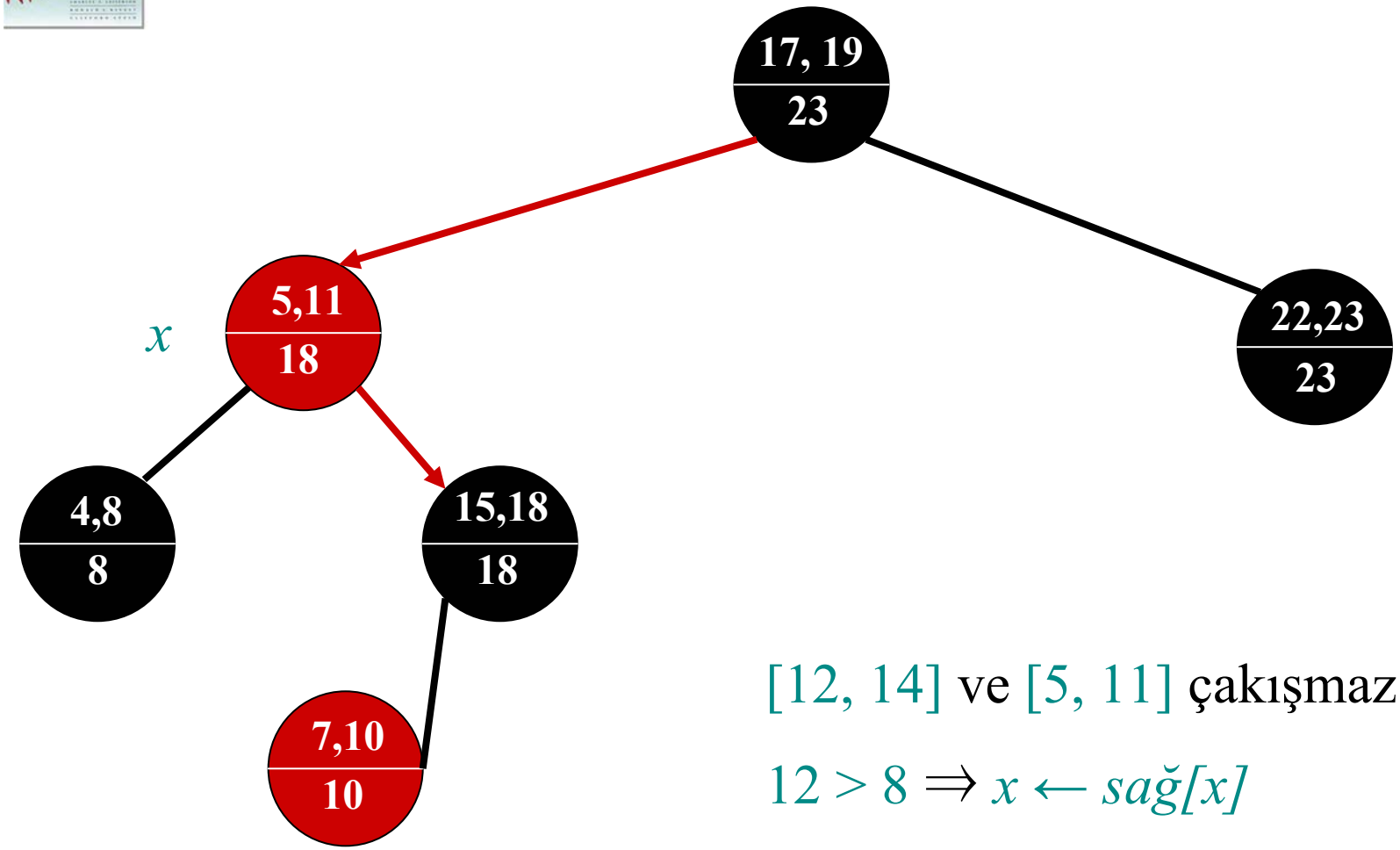

$x \leftarrow kök$

[12, 14] ve [17, 19] çakışmaz.

$12 \leq 18 \Rightarrow x \leftarrow sol[x]$

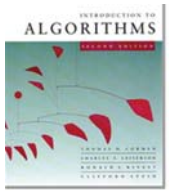



# Örnek 2 : Aralık Arama([12,14])

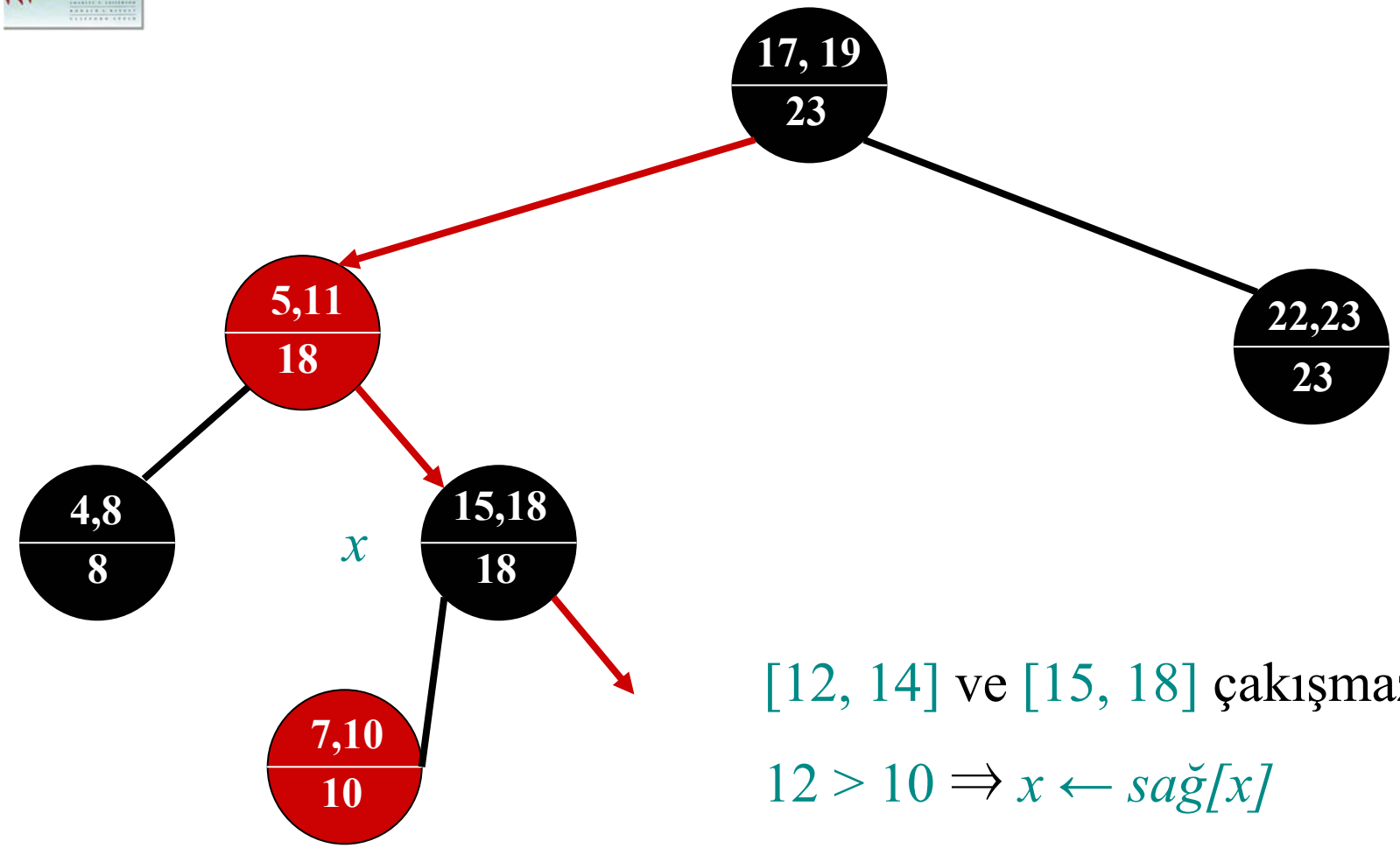



[12, 14] ve [5, 11] çakışmaz.

$12 > 8 \Rightarrow x \leftarrow sağ[x]$

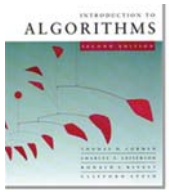


# Örnek 2 : Aralık Arama([12,14])

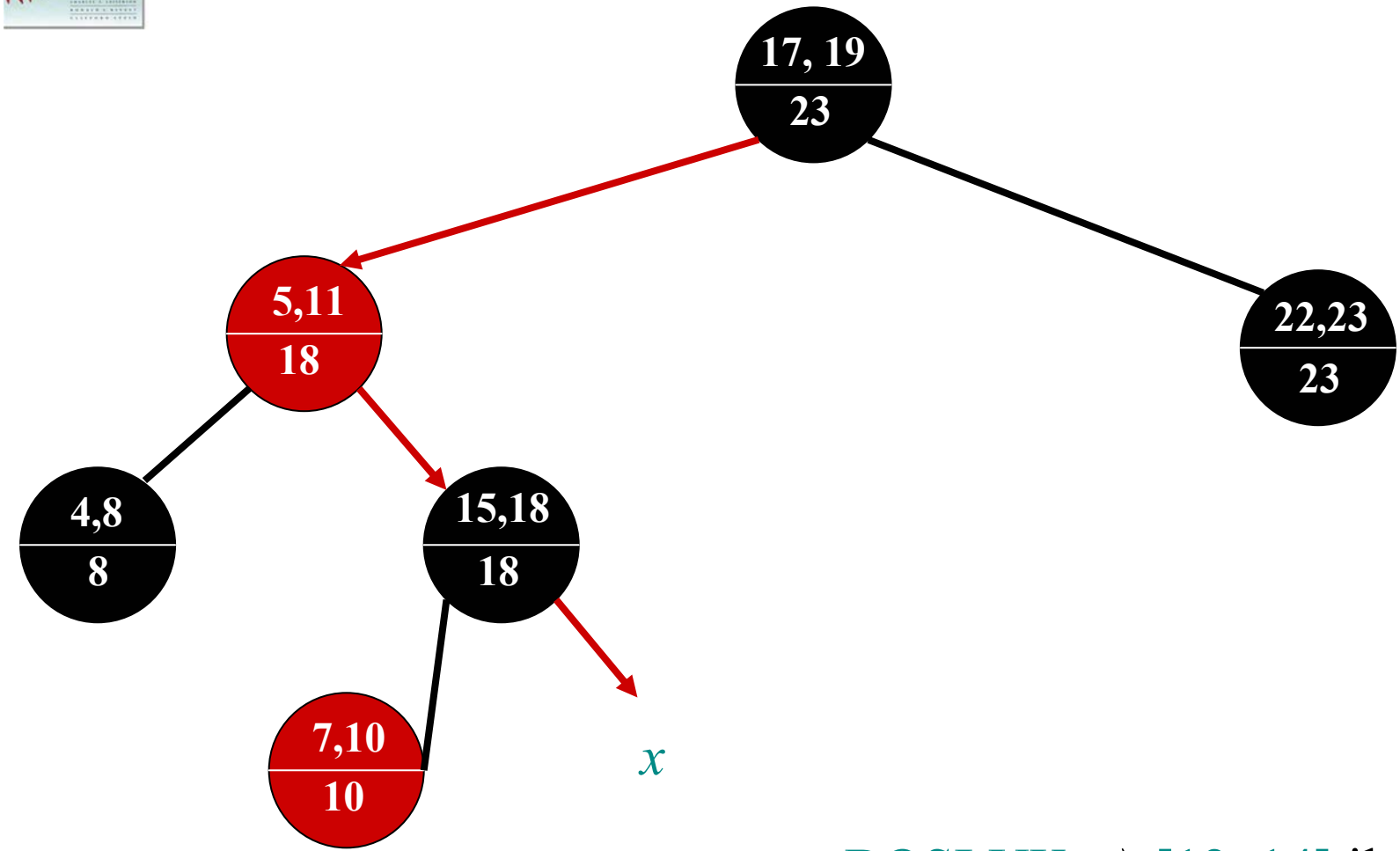



[12, 14] ve [15, 18] çakışmaz.

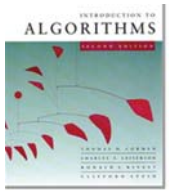
$12 > 10 \Rightarrow x \leftarrow sağ[x]$




# Örnek 2 : Aralık Arama([12,14])




$x$ = BOŞLUK $\Rightarrow$ [12, 14] ile çakışan herhangi bir aralık yok.




# Çözümleme

Zaman = $O(y) = O(\lg n)$ ; Aralık Arama, basit bir yol izlediğinden, her seviyede sabit iş yapmakta.

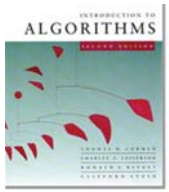
Çakışan bütün aralıkları listeleyin:

- Ara, listele, sil, tekrarla.
- Bunları en sonda tekrar ekleyin.

Zaman = $O(k \lg n)$; $k$ çakışan aralıkların toplam sayısı.

Bu çıktı - duyarlı sınırdır.

Şu ana kadarki en iyi algoritma: $O(k + \lg n)$




# Doğruluk

**Teorem.** $x$düğümünün sol altağacındaki aralıklar $L$, sağ alt ağacındaki aralıklar ise $R$ olsun.

- Eğer arama sağa gider ise,

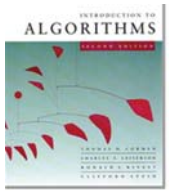
  $\{ i' \in L : i', i \text{ ile çakışır} \} = \varnothing$.

  - Eğer arama sola gider ise,

    $\{ i' \in L : i', i \text{ ile çakışır} \} = \varnothing$

    $\Rightarrow \{ i' \in R : i', i \text{ ile çakışır} \} = \varnothing$

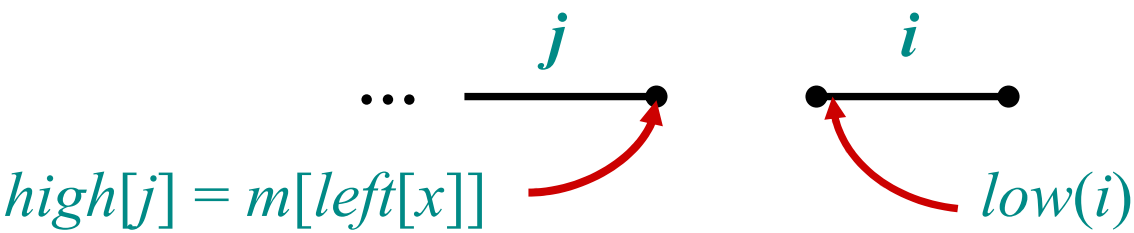
Başka bir deyişle, *2* çocuktan sadece *1*'ini almak her zaman güvenlidir: ya bir şeyler buluruz, ya da bulunacak bir şey yoktur.



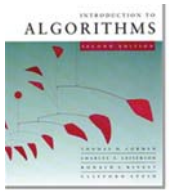

# Doğruluğun İspatı

**İspat.** Aramanın önce sağa gittiğini varsayın.

- Eğer sol[x] = BOŞLUK ise, işimiz tamam, $L = \varnothing$.
- Aksi halde, kod, *düşük[i]>m[sol[x]]* olması gerektiğini dikte ediyor. *m[sol[x]]*'in değeri, $j \in L$ olan bir aralığın son noktasına denk gelmektedir ve $L$'deki herhangi başka bir aralık *yüksek[j]*'den daha büyük bir yüksek son noktaya sahip değildir.




- Ayrıca, $\{ i' \in L : i'$, $i$ ile çakışır$\} = \varnothing$.




# Doğruluğun İspatı (Devamı)

Aramanın önce sola gittiğini varsayın ve

$\{ i' \in L : i', i$ ile çakışır$\} = \varnothing$ olduğunu varsayın.

- Bu durumda, kod, $j \in L$ için $düşük[i] \leq m[sol[x]] = yüksek[j]$ söylüyor.
- $j \in L$ olduğu sürece, $i$ ile çakışmaz ve $yüksek[i] < düşük[j]$ olur.
- Ancak, ikili arama ağacı kuralı, bütün $i' \in R$ *için,* $düşük[j] \leq düşük[i]$ olduğunu söylüyor.
- Öyleyse, $\{i' \in R : i', i$ ile çakışır$\} = \varnothing$.