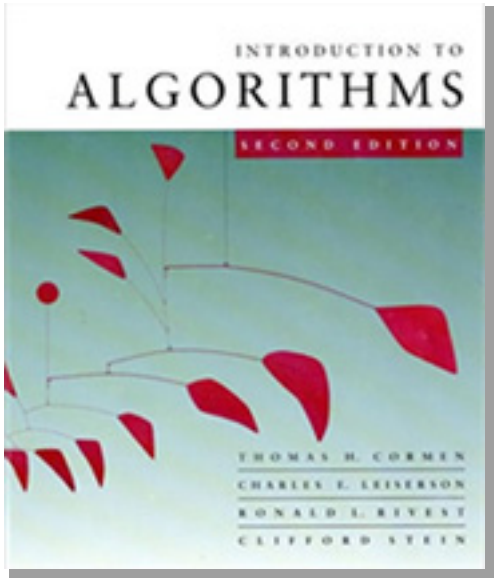
Algoritmalara Giriş

6.046J/18.401J

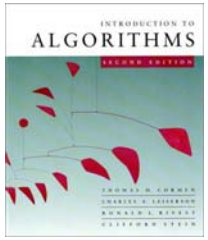


DERS 10

Dengeli Arama Ağaçları

- Kırmızı-siyah ağaçlar
- Kırmızı-siyah ağacın yüksekliği
- Rotation / Dönme
- Insertion / araya yerleştirme

Prof. Erik Demaine

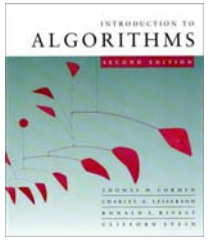


Dengeli arama ağaçları

Dengeli arama ağacı: ***n*** elemanlı bir değişken kümede işlem yaparken $O(\lg n)$ yüksekliğinin garanti edildiği bir arama ağacı veri yapısı.

Örnekler:

- AVL ağaçları
- 2-3 ağaçları
- 2-3-4 ağaçları
- B-ağaçları
- Kırmızı-siyah ağaçlar

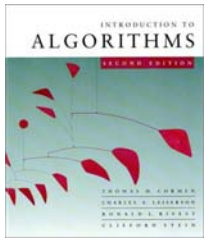


Kırmızı-siyah ağaçlar

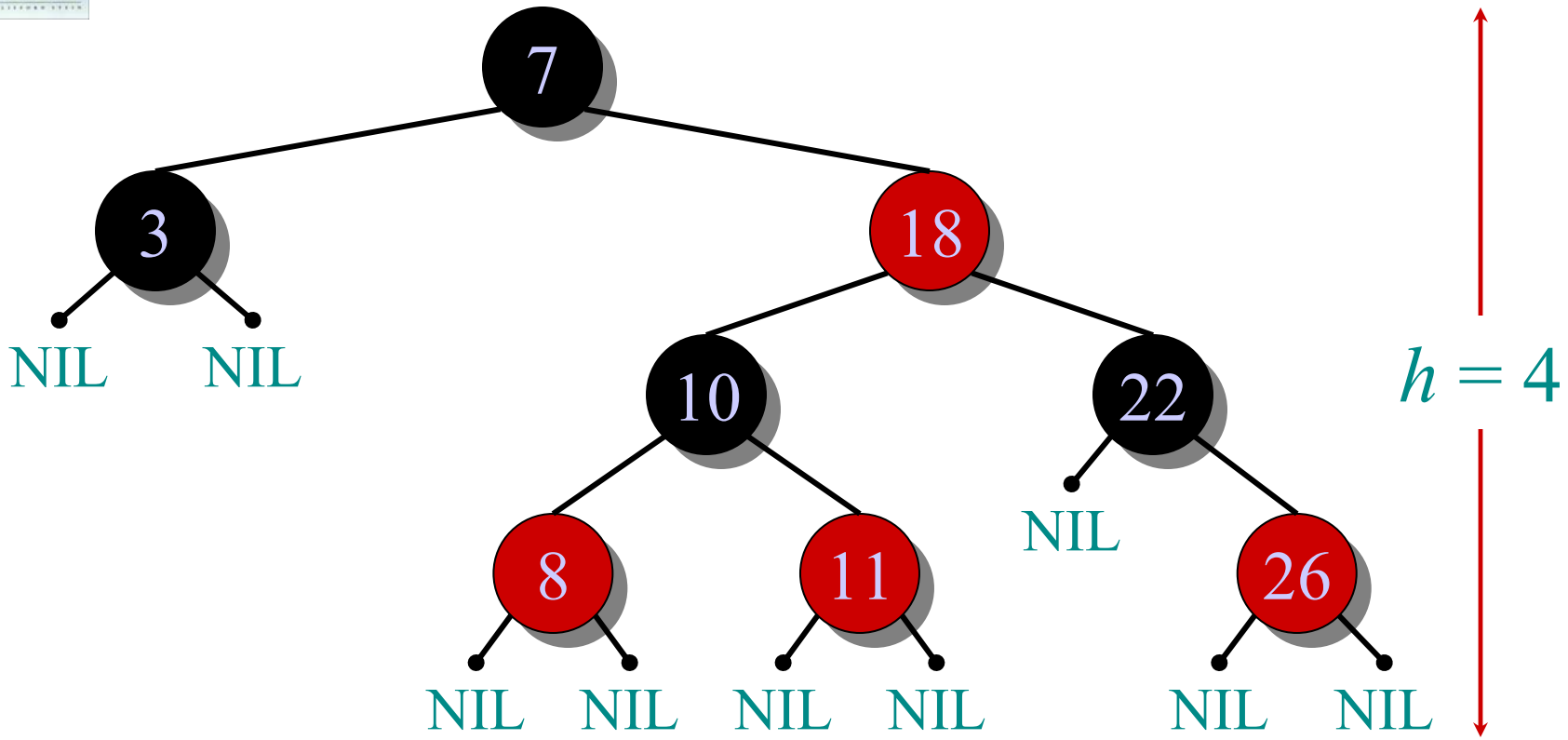
Bu veri yapısının her düğümünde bir-bitlik renk alanına ihtiyaç vardır.

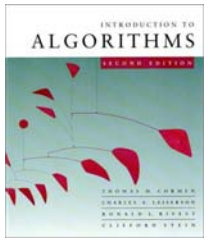
Kırmızı-siyah özellikler:

1. Her düğüm ya kırmızı ya da siyahtır.
2. Kök ve yapraklar (NIL'ler yani sıfır'lar) siyahtır.
3. Eğer bir düğüm kırmızı ise, atası siyahtır.
4. Herhangi bir x düğümünden ardıl yaprağa giden basit yollarda aynı sayıda siyah düğüm vardır = black-height(x) yani siyah-yükseklik(x).

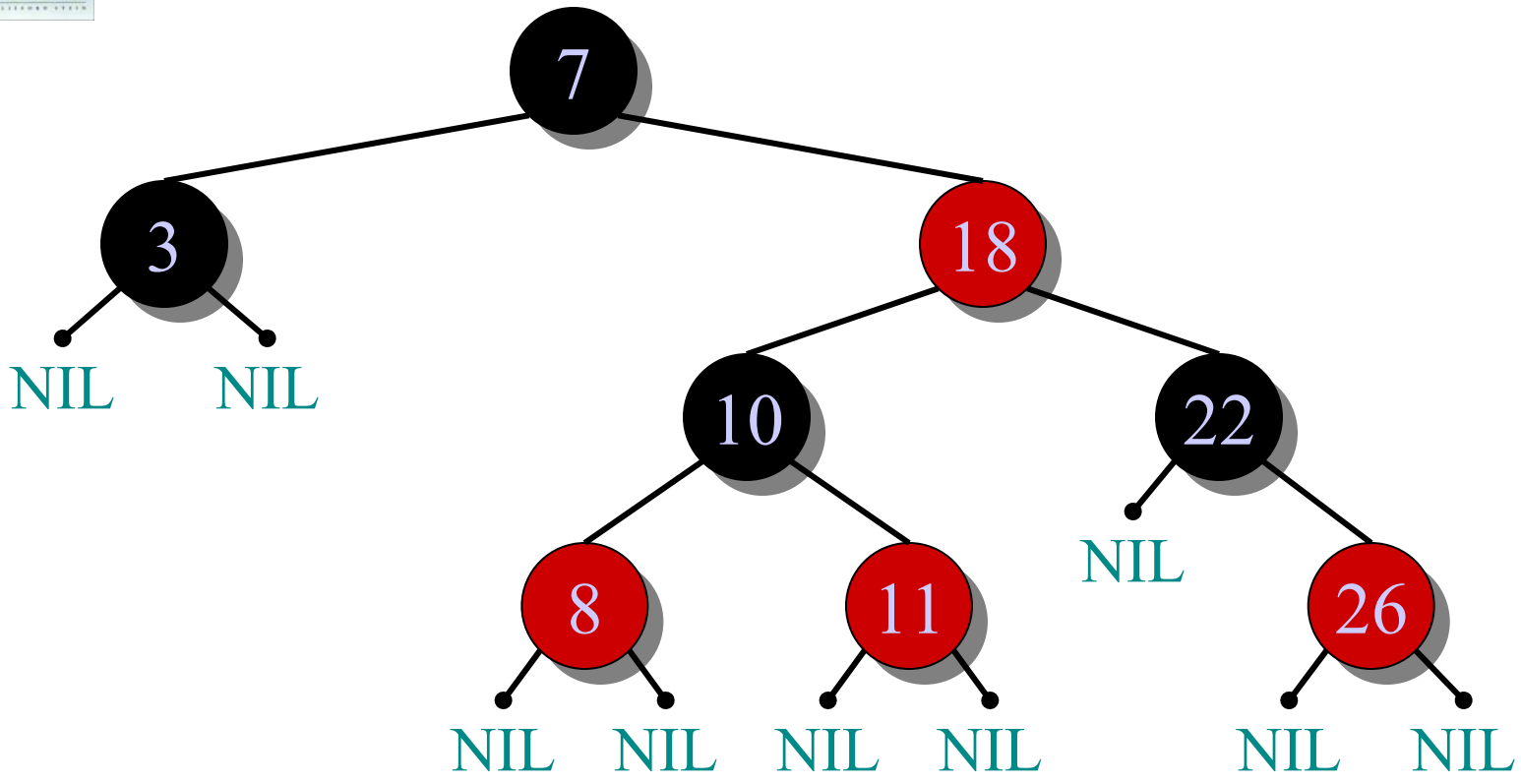


Bir kırmızı-siyah ağaç örneği

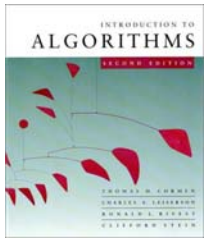




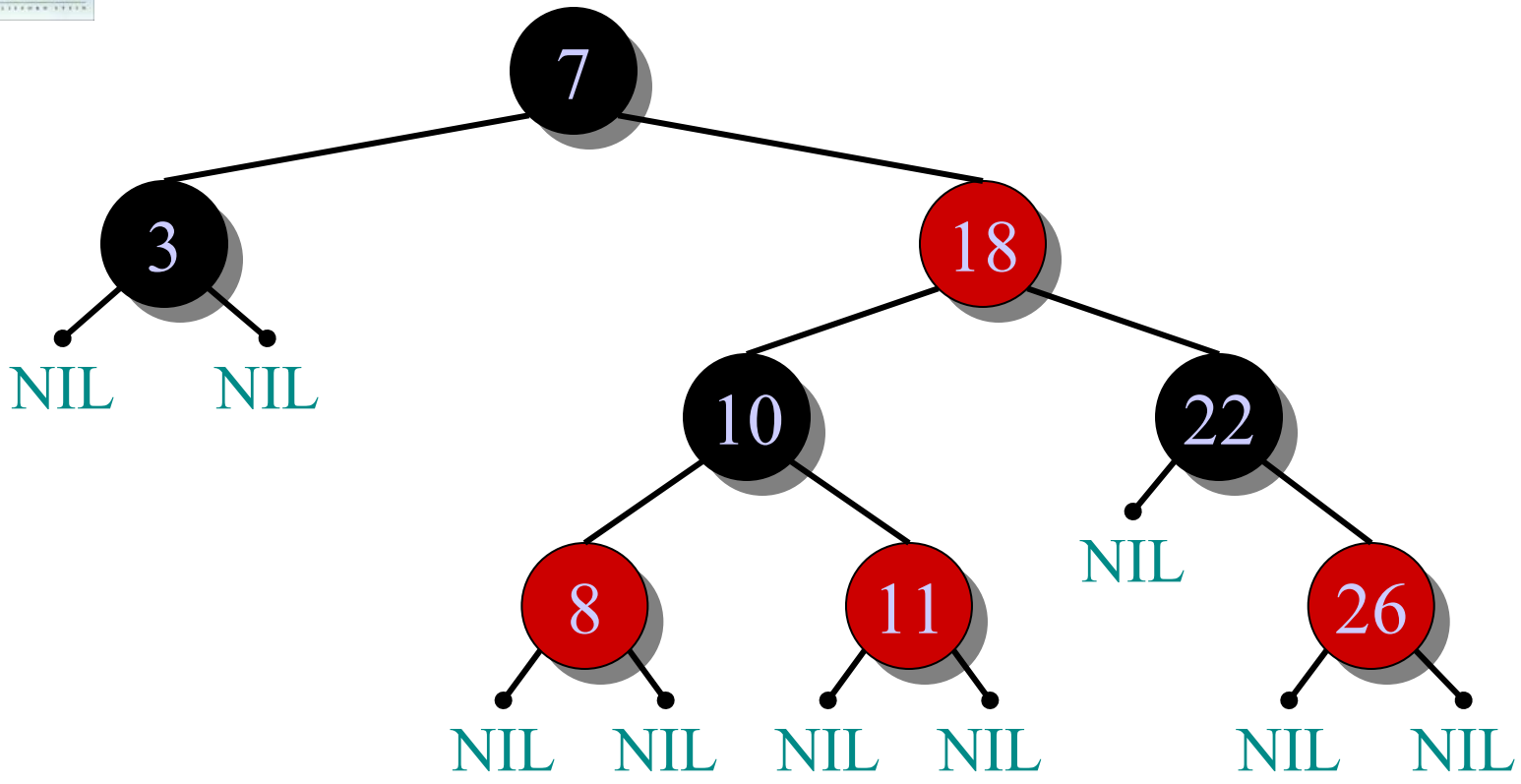
Bir kırmızı-siyah ağaç örneği



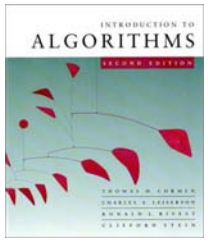
1. Her düğüm ya kırmızı ya siyahtır.



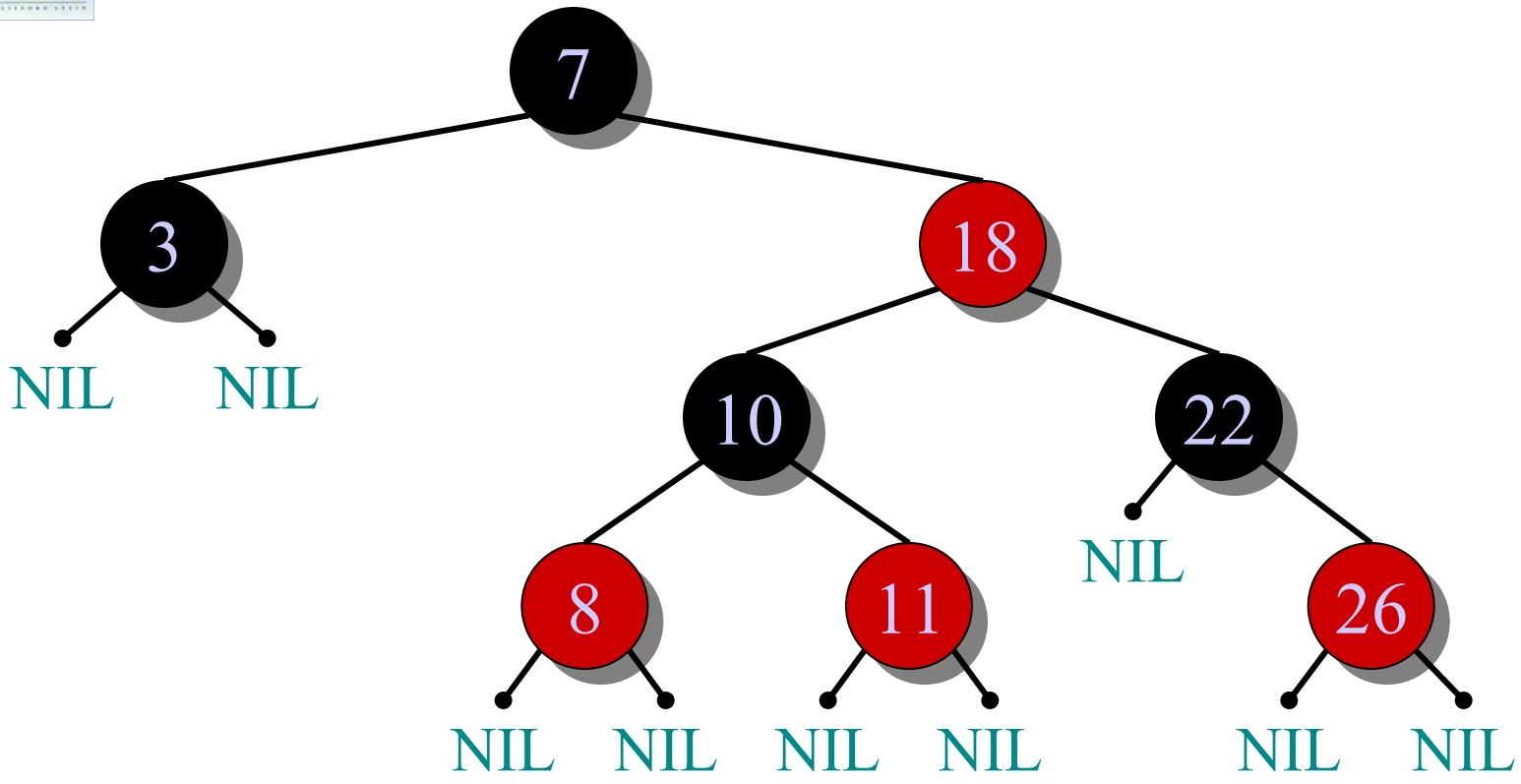
Bir kırmızı-siyah ağaç örneği



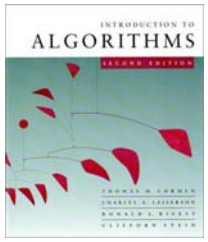
2. Kök ve yapraklar (NIL'ler) siyahtır.



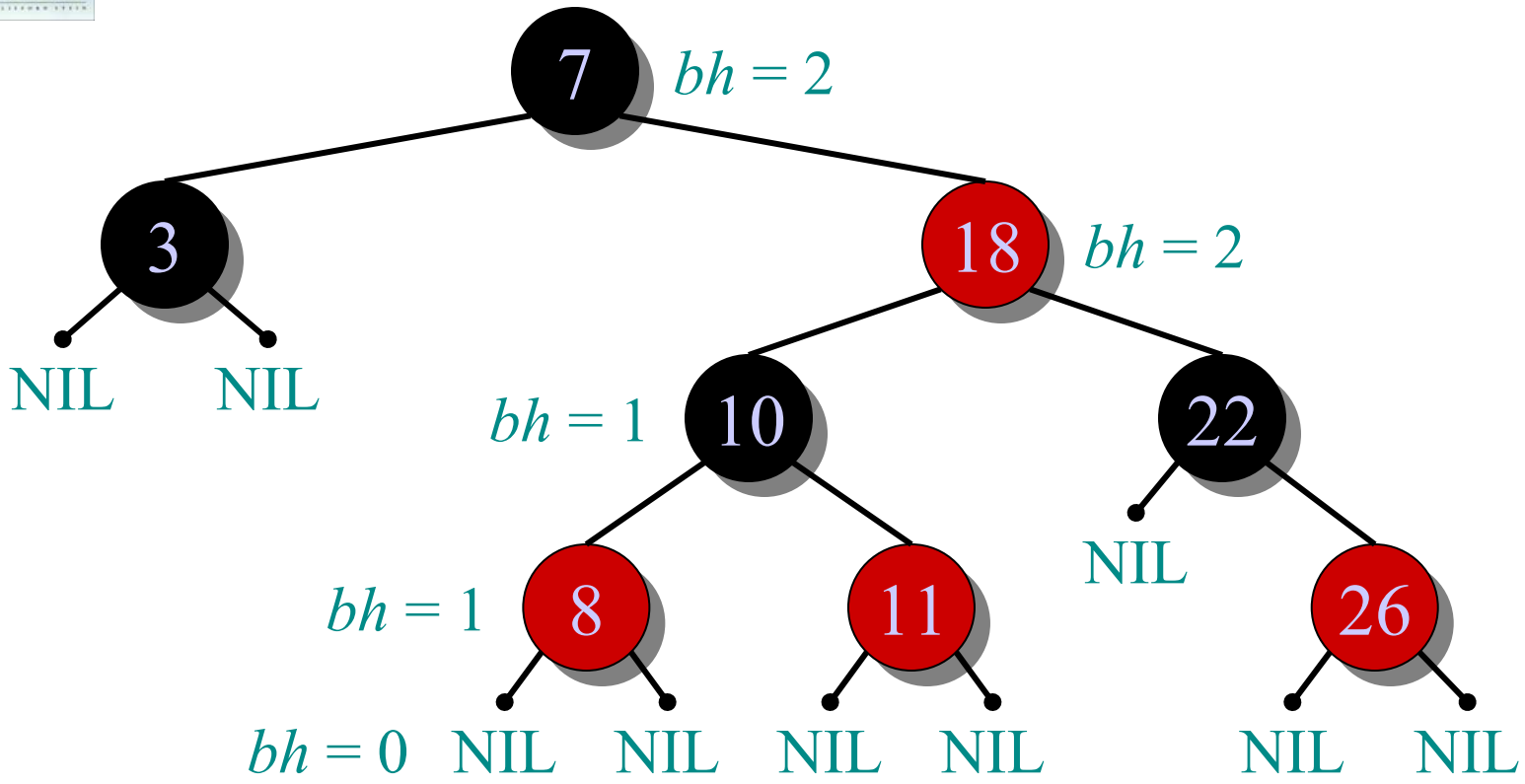
Bir kırmızı-siyah ağaç örneği



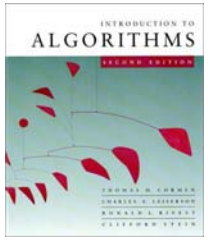
3. Eğer bir düğüm kırmızı ise, atası siyahtır.



Bir kırmızı-siyah ağaç örneği



4. Herhangi bir x düğümünden ardıl yaprağa giden basit yollarda aynı sayıda siyah düğüm vardır = *siyah-yükseklik*(x).



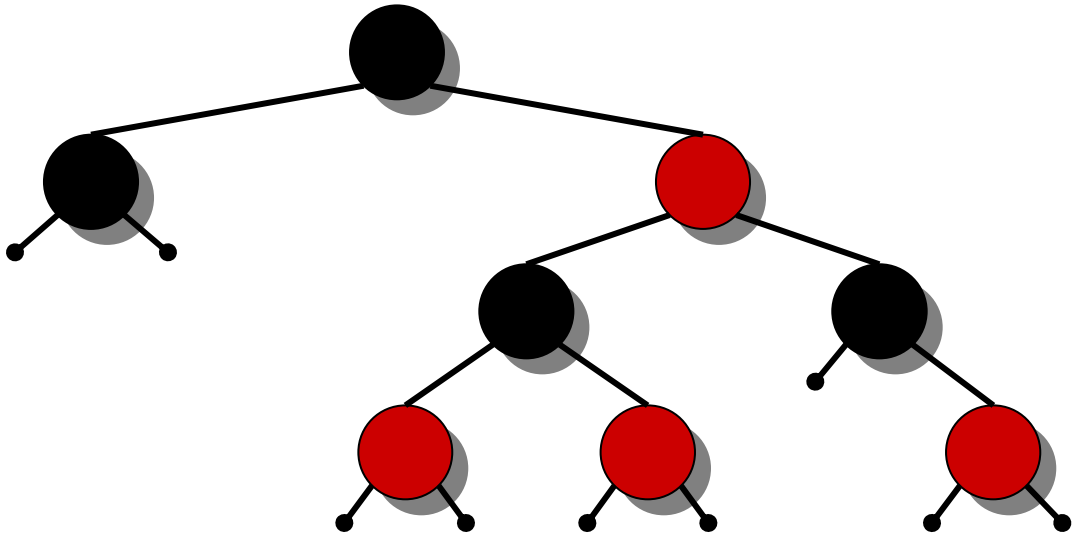
Kırmızı-siyah ağacın yüksekliği

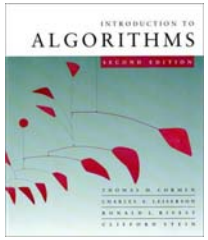
Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği $h \leq 2 \lg(n + 1)$ dir.

Kanıt. (Kitap tümevarımı kullanıyor. Dikkatle okuyun.)

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarına yaklaştırın.





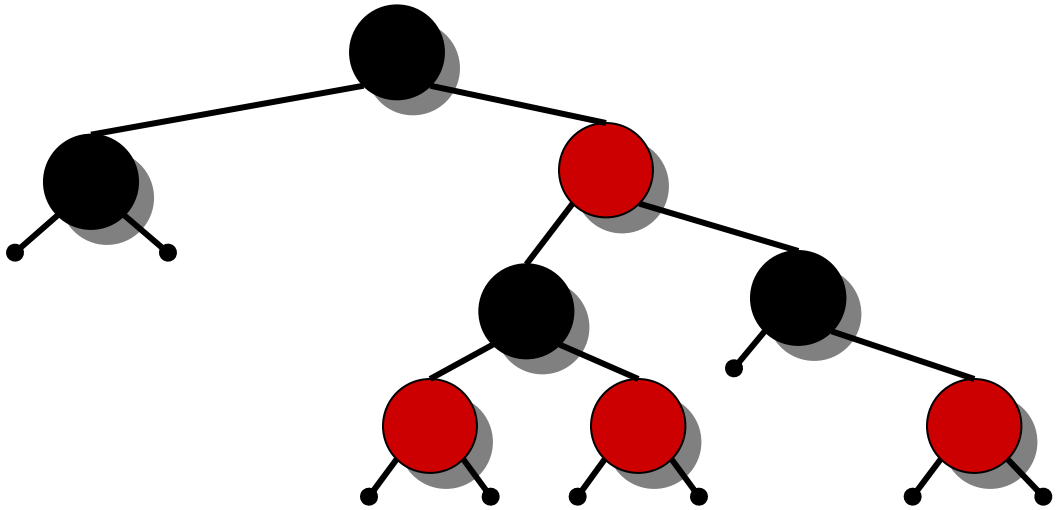
Kırmızı-siyah ağacın yüksekliği

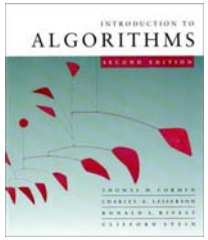
Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği $h \leq 2 \lg(n + 1)$ dir.

Kanıt. (Kitap tümevarımı kullanıyor. Dikkatle okuyun.)

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarına yaklaştırın.





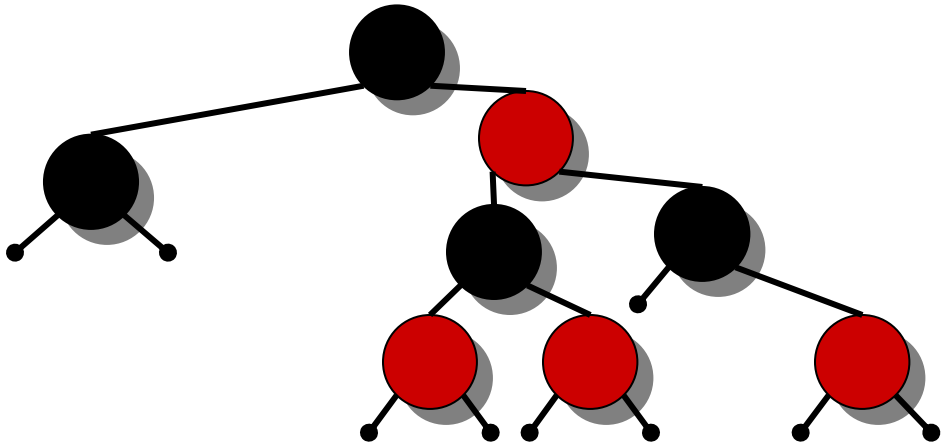
Kırmızı-siyah ağacın yüksekliği

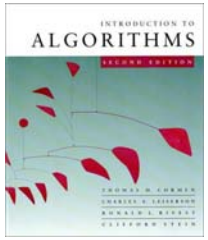
Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği $h \leq 2 \lg(n + 1)$ dir.

Kanıt. (Kitap tümevarımı kullanıyor. Dikkatle okuyun.)

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarına yaklaştırın.





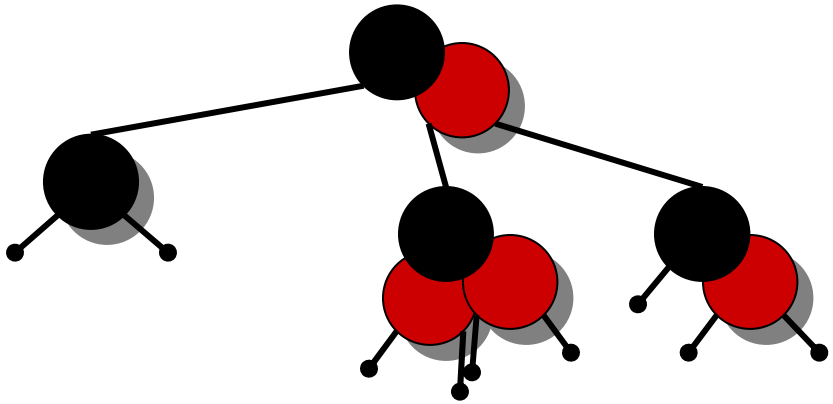
Kırmızı-siyah ağacın yüksekliği

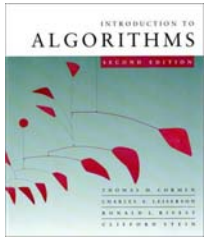
Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği $h \leq 2 \lg(n + 1)$ dir.

Kanıt. (Kitap tümevarımı kullanıyor. Dikkatle okuyun.)

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarıyla birleştirin.





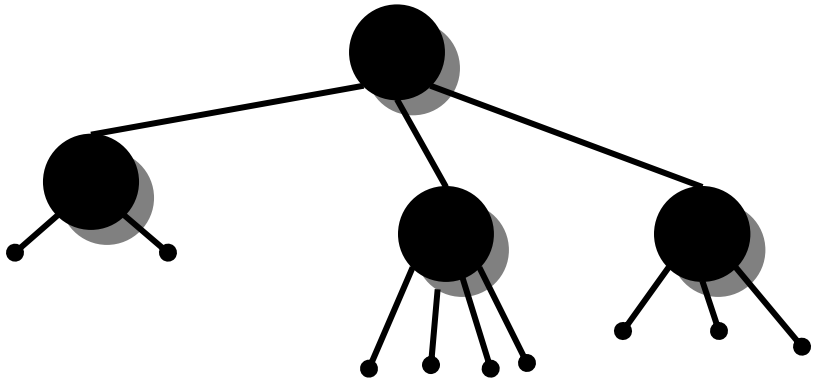
Kırmızı-siyah ağacın yüksekliği

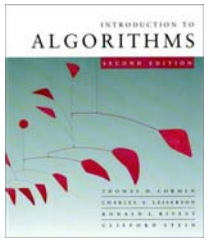
Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği $h \leq 2 \lg(n + 1)$ dir.

Kanıt. (Kitap tümevarımı kullanıyor. Dikkatle okuyun.)

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarıyla bütünleştirin.





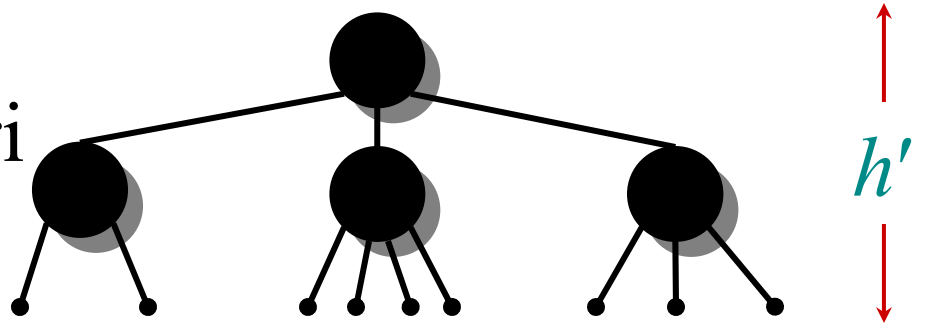
Kırmızı-siyah ağacın yüksekliği

Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği $h \leq 2 \lg(n + 1)$ dir.

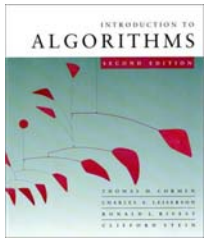
Kanıt. (Kitap tümevarımı kullanıyor. Dikkatle okuyun.)

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarıyla bütünleştirin.



- Bu işlem sonucunda oluşan ağacın her düğümünün 2, 3, ya da 4 ardılı olur.
- 2-3-4 ağacının yapraklarının derinliği h' tekbiçimlidir.



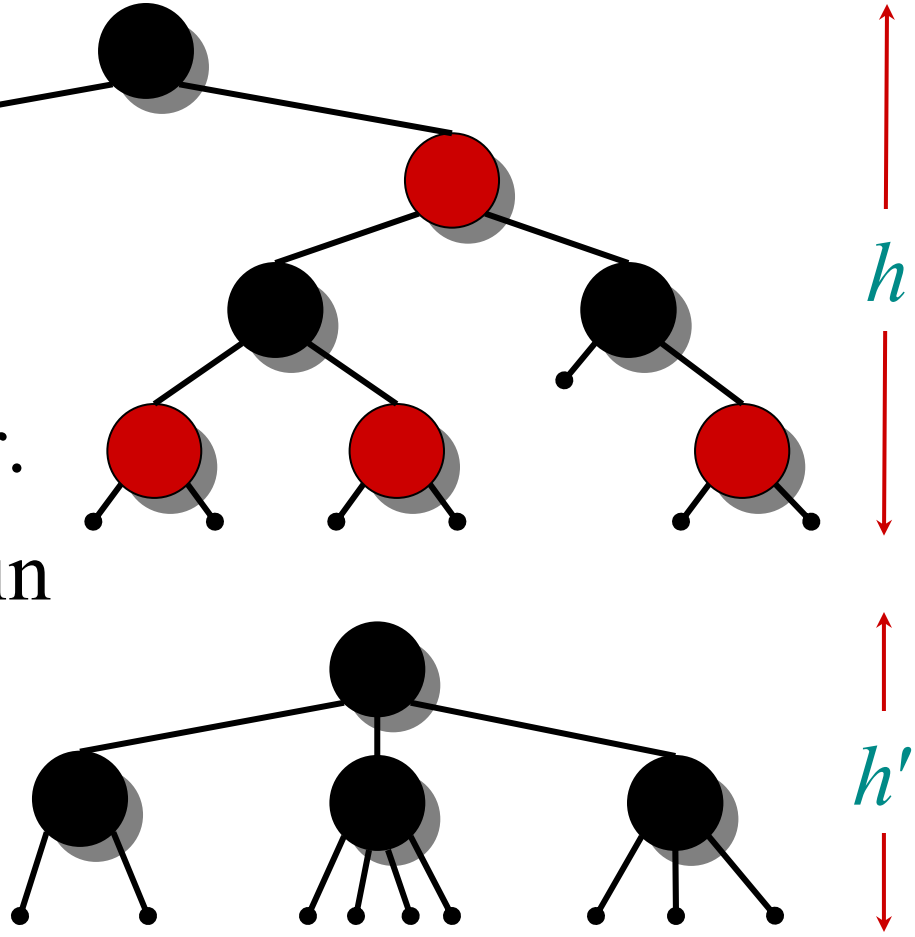
Kanıtlama (devamı)

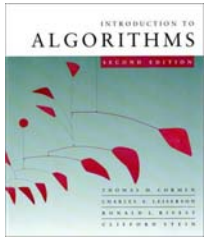
- Elimizde $h' \geq h/2$ olur, çünkü her yoldaki yaprakların en çok yarısı kırmızıdır.
- Her ağaçtaki yaprakların sayısı: $n + 1$

$\Rightarrow n + 1 \geq 2^{h'}$

$\Rightarrow \lg(n + 1) \geq h' \geq h/2$

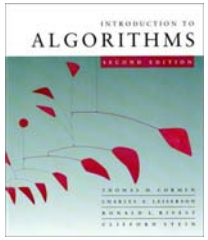
$\Rightarrow h \leq 2 \lg(n + 1)$. □





Sorgulama işlemleri

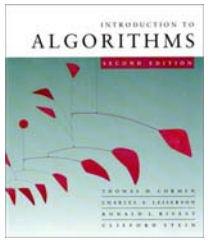
Corollary (Doğal sonuç). n düğümlü bir kırmızı-siyah ağaçta SEARCH (ARAMA), MIN, MAX, SUCCESSOR (ARDIL) ve PREDECESSOR (ATA) sorgulamalarının hepsi $O(\lg n)$ süresinde çalışırlar.



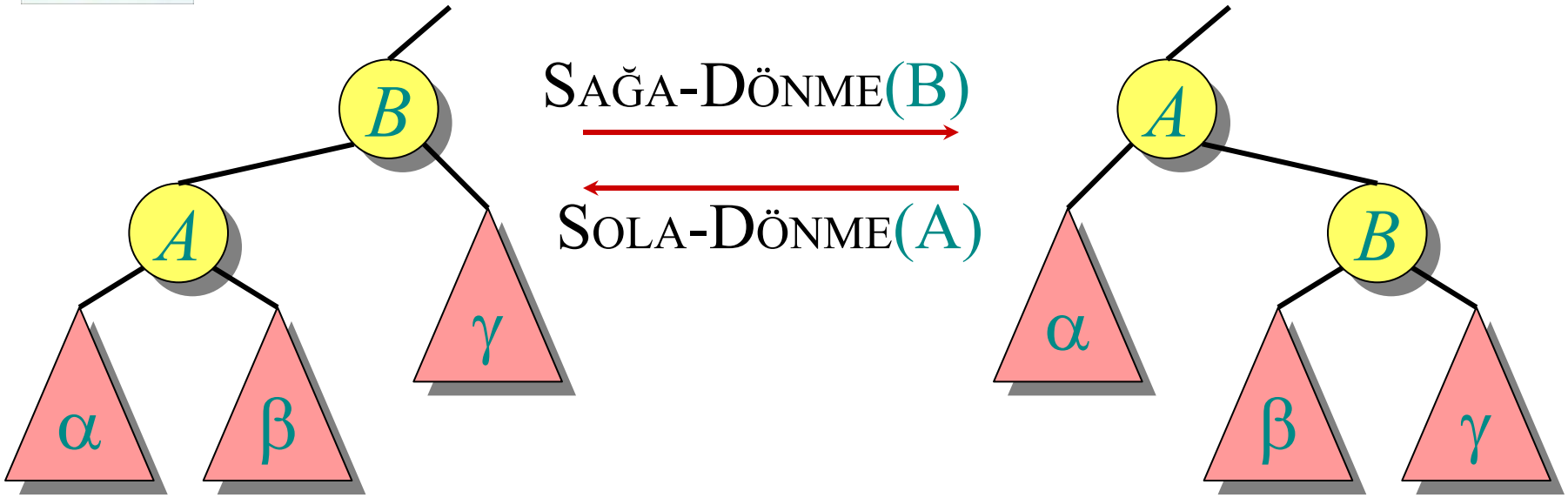
Değiştirme işlemleri

INSERT (ARAYA YERLEŞTİRME) ve DELETE (SİLME) işlemleri kırmızı-siyah ağaçta değişime neden olur:

- işlemin kendi yapısı,
- renk değişimleri,
- ağacın bağlantılarının ***"rotations/rotasyonlar"*** yordamıyla yeniden yapılanması.



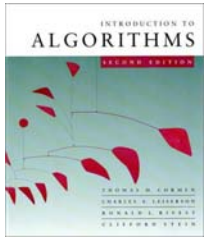
Rotasyonlar / Dönmeler



Rotasyonlar anahtarların sıralı düzenini korurlar:

- $a \in \alpha, b \in \beta, c \in \gamma \Rightarrow a \le A \le b \le B \le c$.

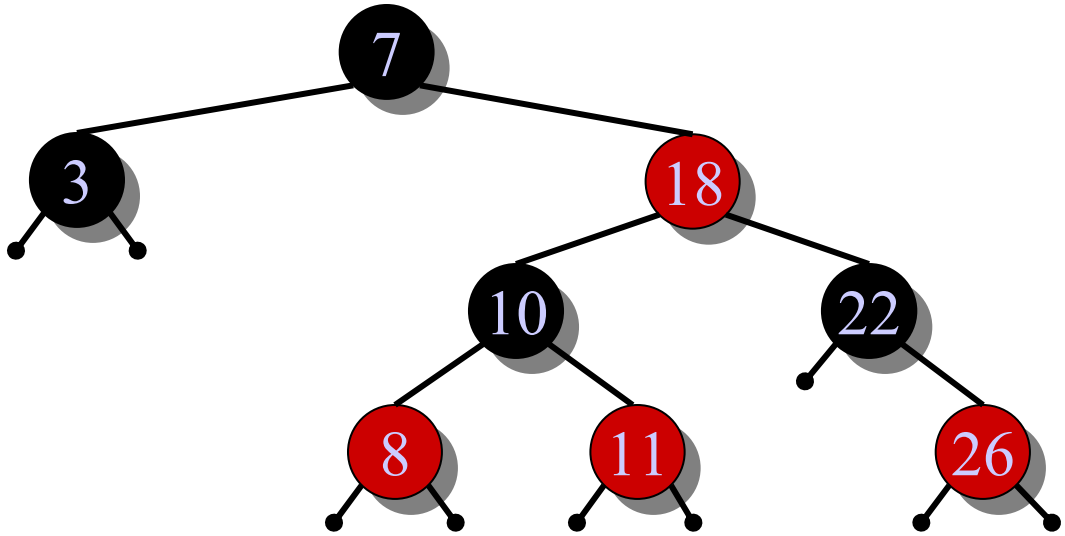
Bir rotasyon $O(1)$ sürede yapılabilir.

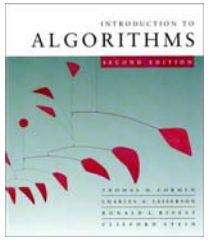


Kırmızı-siyah ağaçta araya yerleştirme

FİKİR: Ağaçta x' i araya yerleştirin. x' i kırmızı yapın. Sadece kırmızı-siyah özellik 3 ihlal edilebilir. İhlali ağaç boyunca yukarı doğru, rotasyonlar ve yeniden renklendirmeyle düzelene kadar taşıyın.

Örnek:



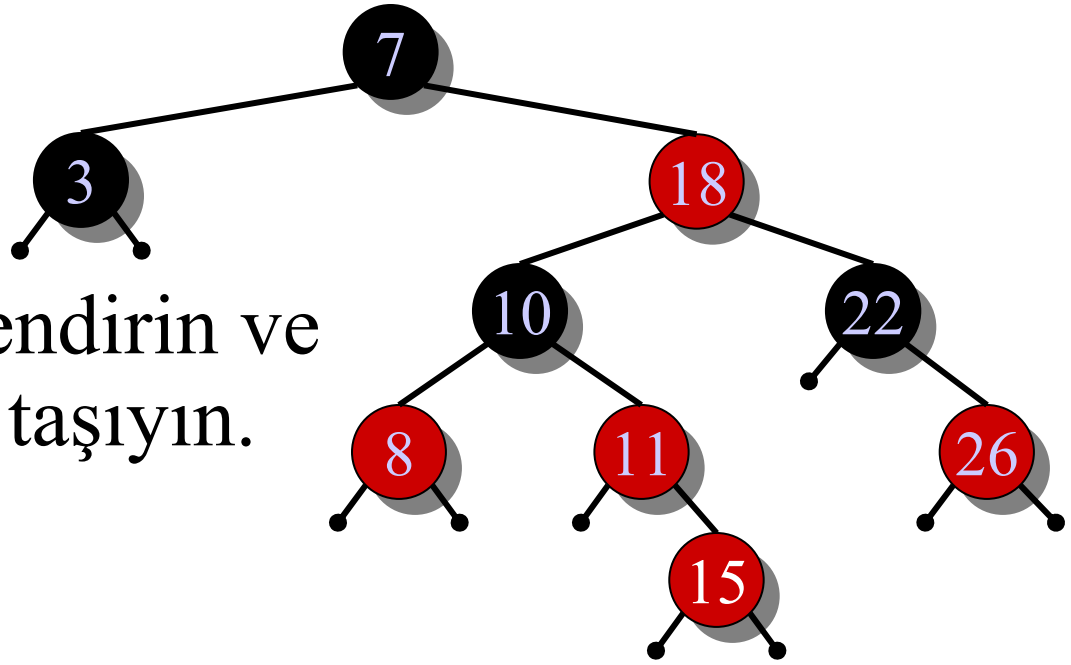


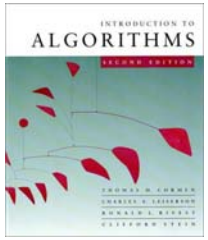
Kırmızı-siyah ağaçta araya yerleştirme

FİKİR: Ağaçta x' i araya yerleştirin. x' i kırmızı yapın. Sadece kırmızı-siyah özellik 3 ihlal edilebilir. İhlali ağaç boyunca yukarı doğru, rotasyonlar ve yeniden renklendirmeyle düzelene kadar taşıyın.

Örnek:

- Ar.Yer. $x = 15$.
- Yeniden renklendirin ve ihlali yukarıya taşıyın.



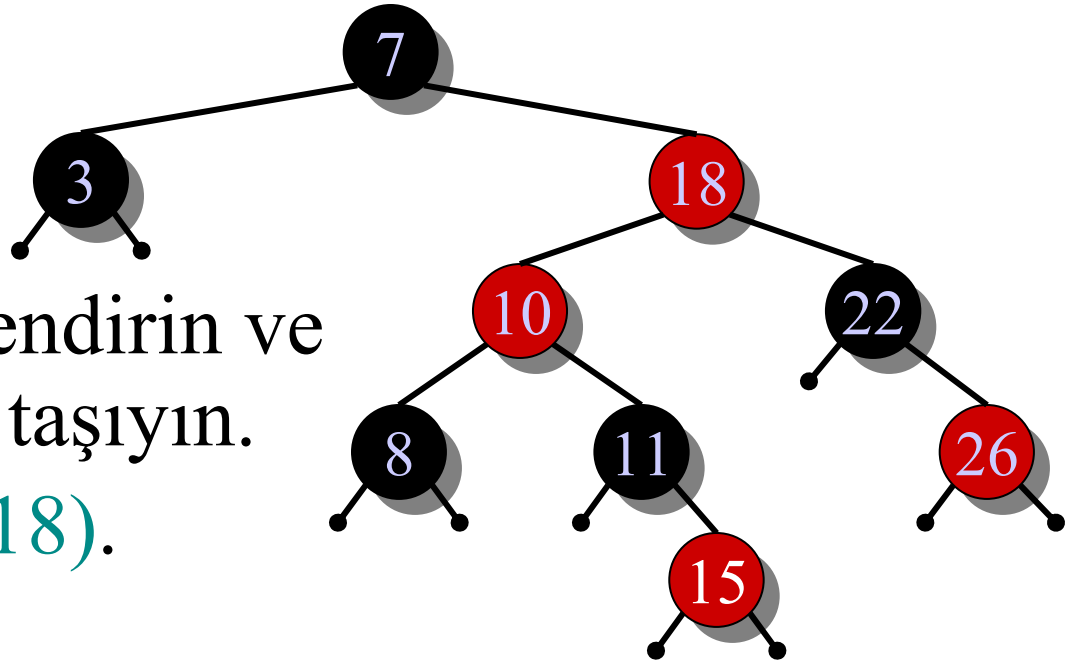


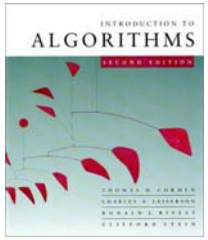
Kırmızı-siyah ağaçta araya yerleştirme

FİKİR: Ağaçta x' i araya yerleştirin. x' i kırmızı yapın. Sadece kırmızı-siyah özellik 3 ihlal edilebilir. İhlali ağaç boyunca yukarı doğru, rotasyonlar ve yeniden renklendirmeyle düzelene kadar taşıyın.

Örnek:

- Ar.Yer. $x = 15$.
- Yeniden renklendirin ve ihlali yukarıya taşıyın.
- SAĞA-DÖNME(18).



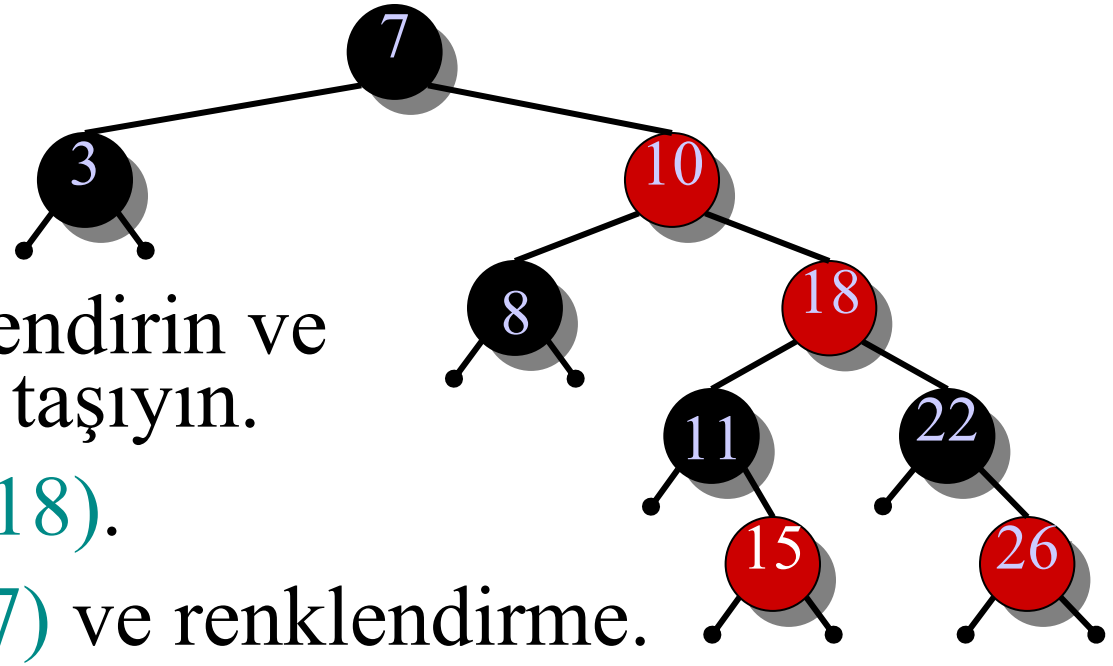


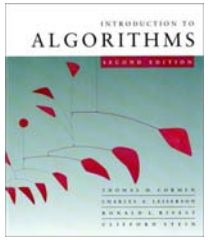
Kırmızı-siyah ağaçta araya yerleştirme

FİKİR: Ağaçta x' i araya yerleştirin. x' i kırmızı yapın. Sadece kırmızı-siyah özellik 3 ihlal edilebilir. İhlali ağaç boyunca yukarı doğru, rotasyonlar ve yeniden renklendirmeyle düzelene kadar götürün.

Örnek:

- Ar.Yer. $x = 15$.
- Yeniden renklendirin ve ihlali yukarıya taşıyın.
- SAĞA-DÖNME(18).
- SOLA-DÖNME(7) ve renklendirme.



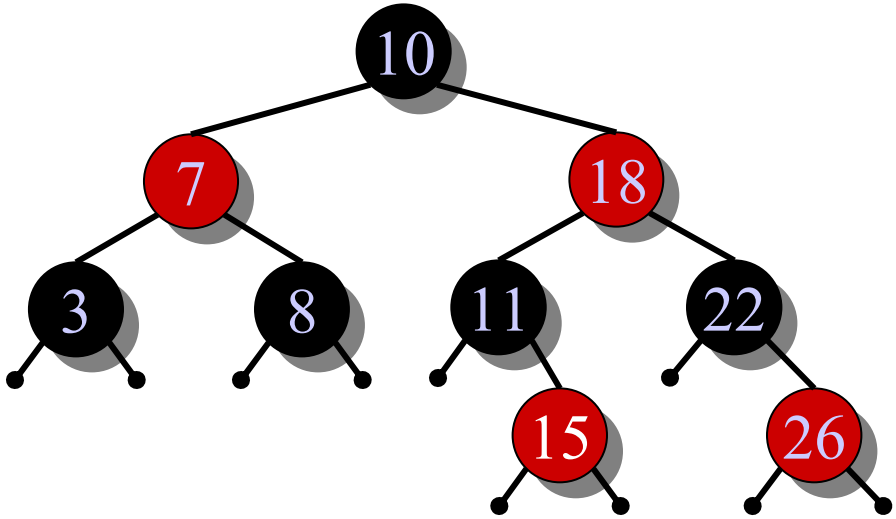


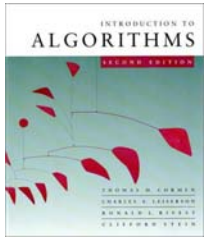
Kırmızı-siyah ağaçta araya yerleştirme

FİKİR: Ağaçta x' i araya yerleştirin. x' i kırmızı yapın. Sadece kırmızı-siyah özellik 3 ihlal edilebilir. İhlali ağaç boyunca yukarı doğru, rotasyonlar ve yeniden renklendirmeyle düzeltilene kadar götürün .

Örnek:

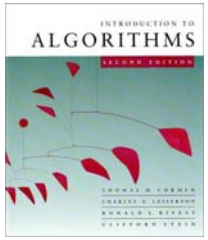
- Ar.Yer. $x = 15$.
- Yeniden renklendir, ihlali yukarıya taşı.
- SAĞA-DÖNME(18).
- SOLA-DÖNME(7) ve yeniden renklendirme.





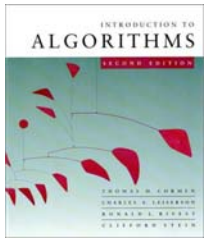
Pseudocode-Sözde kod (İngilizce)

RB-INSERT(T, x)
    TREE-INSERT(T, x)
    $color[x] \leftarrow$ RED ▷ only RB property 3 can be violated
    while $x \neq root[T]$ and $color[p[x]] =$ RED
        do if $p[x] = left[p[p[x]]$
            then $y \leftarrow right[p[p[x]]$ ▷ y = aunt/uncle of x
                if $color[y] =$ RED
                then ⟨**Case 1**⟩
                else if $x = right[p[x]]$
                    then ⟨**Case 2**⟩ ▷ Case 2 falls into Case 3
                ⟨**Case 3**⟩
            else ⟨**"then"** clause with "*left*" and "*right*" swapped⟩
    $color[root[T]] \leftarrow$ BLACK

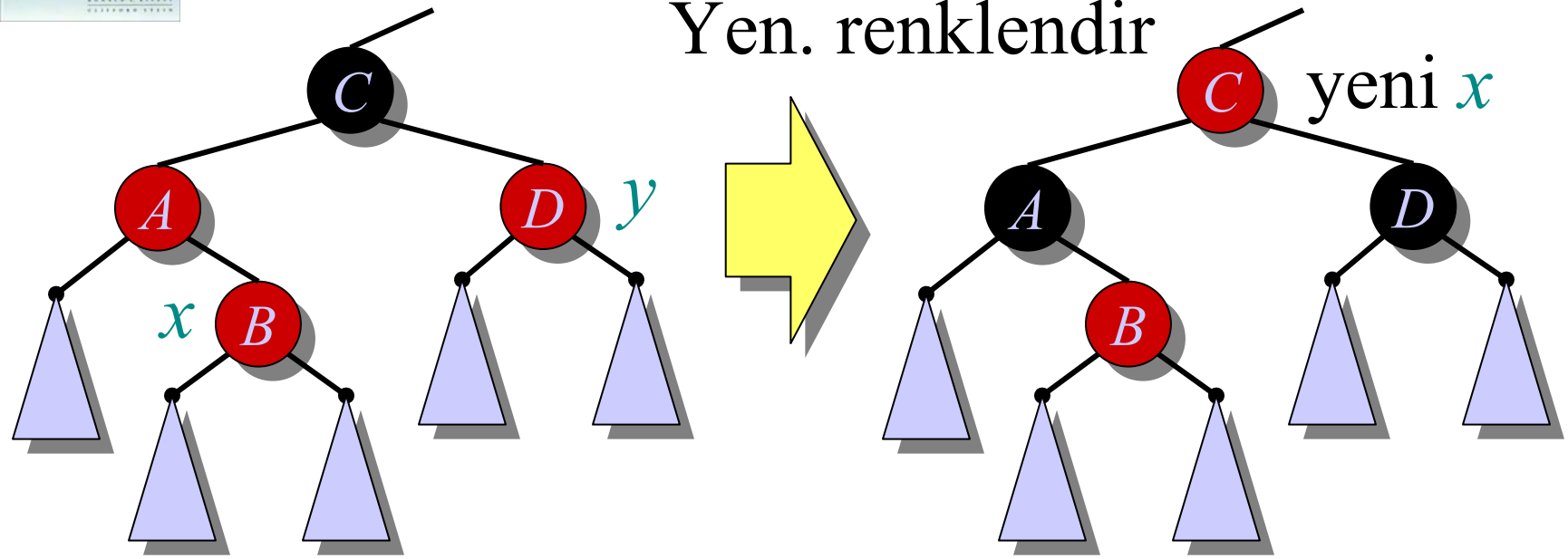


Grafik simgelem

- siyah kökü olan bir altağacı tanımlasın.
- ’ın tümünün siyah-yükseklikleri aynıdır.

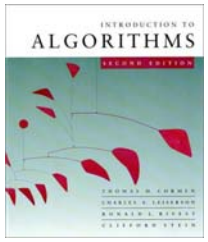


Durum 1

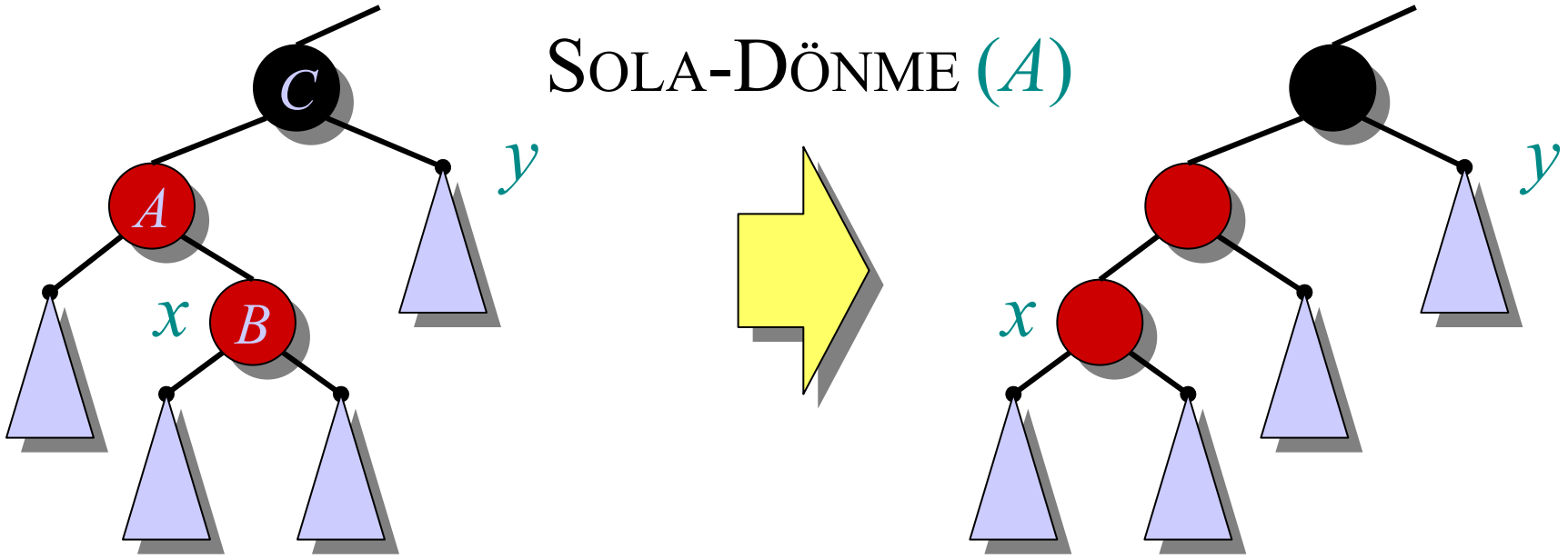


(veya, *A*'nın ardılları yer değiştirir.)

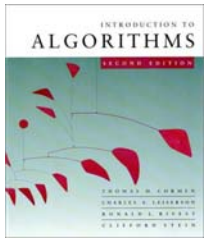
C 'nin siyahını *A* ve *D* 'ye doğru itin ve özyineleme yapın, çünkü *C*'nin atası kırmızı olabilir.



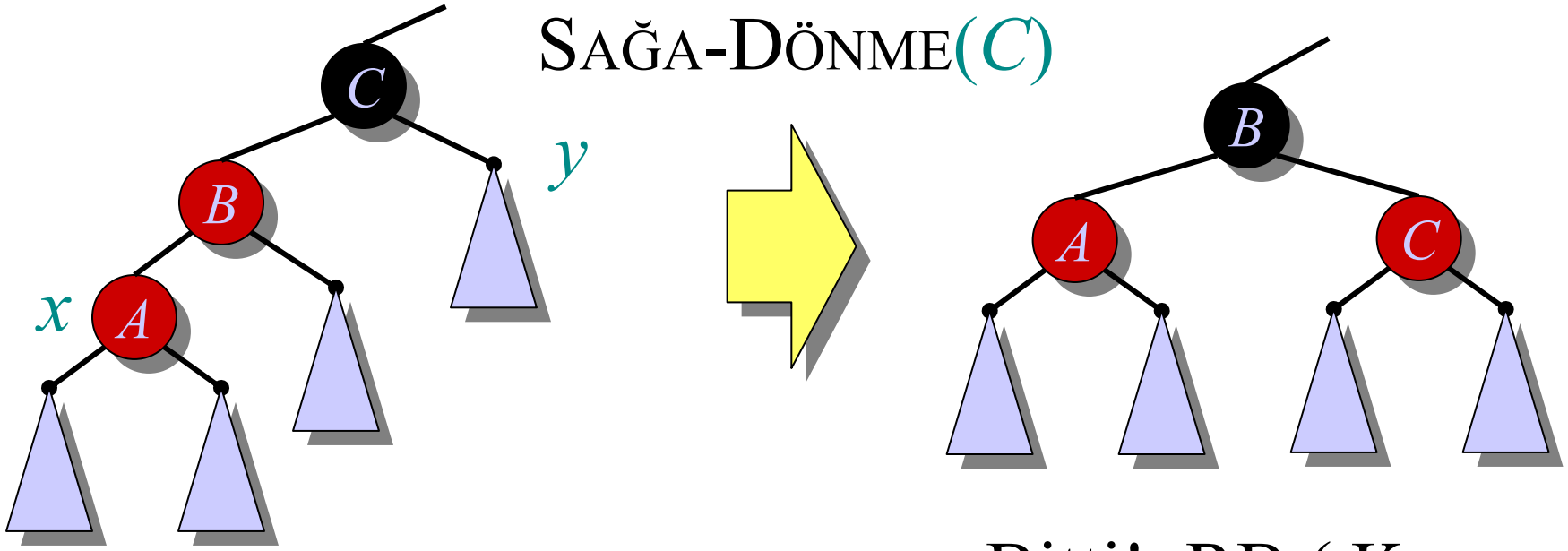
Durum 2



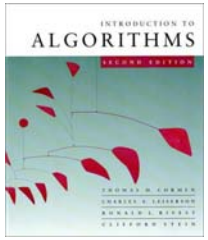
Durum 3'e dönüştürün.



Durum 3



Bitti! RB (Kırmızı-siyah) 3. özelliğin ihlali artık mümkün değil.



Çözümleme

- Ağaçta yukarıya giderken Durum 1' i uygulayın; bu durumda sadece düğümler yeniden renklendirilir.
- Eğer Durum 2 veya 3 ile karşılaşırsanız, 1 ya da 2 rotasyon yapın ve işlemi sonlandırın.

Yürütüm süresi: $O(\lg n)$ ve $O(1)$ rotasyon.

RB-DELETE (KIRMIZI_SİYAH SİLME) — asimptotik koşma süresi ve rotasyonların sayısı RB-INSERT (KIRMIZI-SİYAH ARAYA YERLEŞTİRME) ile aynıdır. (Kitaba bakınız.)