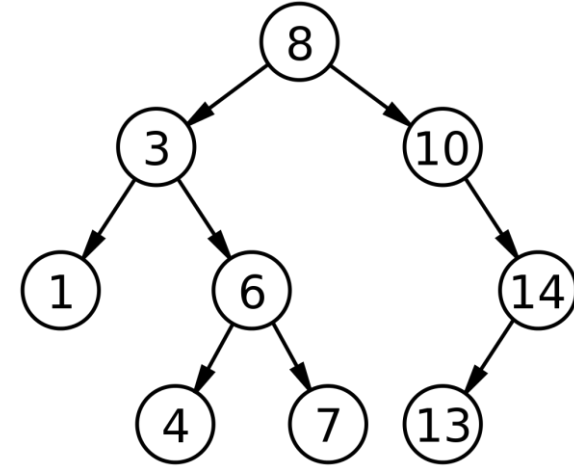


ARAMA (SEARCHING) (3)

Rastgele yapılanmış ikili
arama ağaçları

İKİLİ ARAMA AĞAÇLARI

- ❖ İkili arama ağacı, verileri organize etmek için kullanılan bir çeşit ikili ağaçtır.
- ❖ İkili ağaçtan temel farkı, verilerin sıralanmış bir şekilde tutulmasıdır.
- ❖ Bu sayede ikili arama algoritmasının kullanılmasına imkân verir.



İKİLİ ARAMA AĞAÇLARI

❖ İkili arama ağacı yaklaşımının, çalışma zamanı olarak doğrusal aramadan daha iyidir.

Linear Search

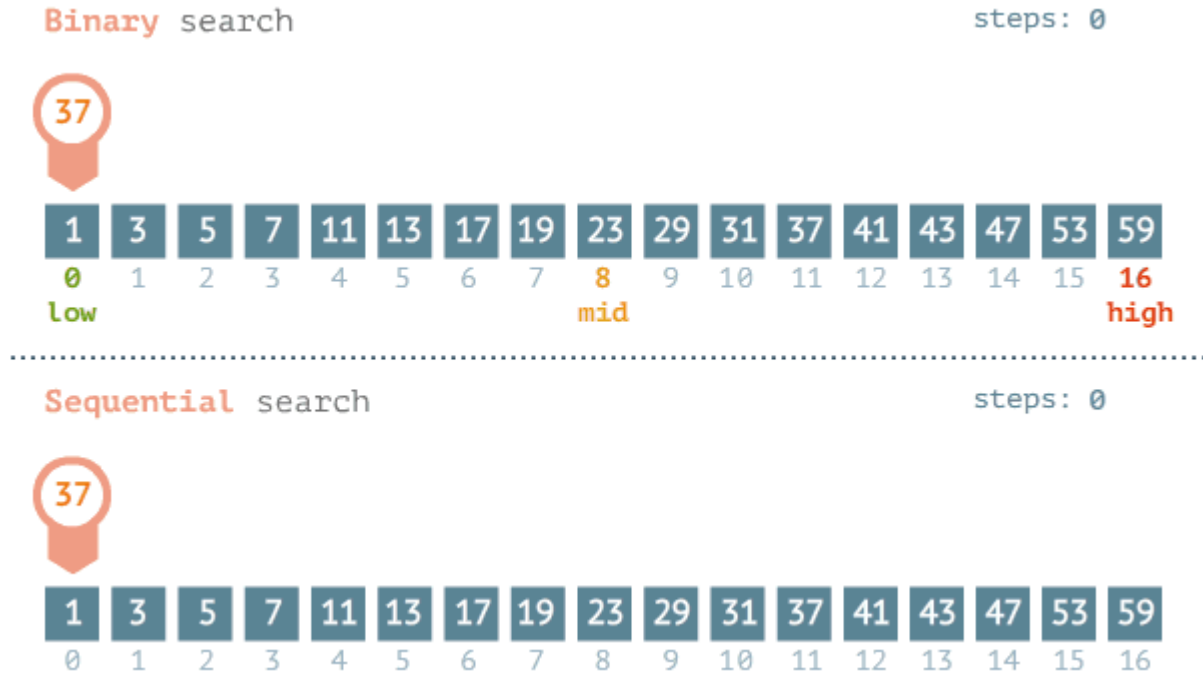


❖ Doğrusal aramanın hesaplama karmaşıklığı $O(n)$ 'dir.

❖ Yani Uzunluğu “ n ” olan bir dizi için en kötü durumda “ n ” kez çalışacaktır.

İKİLİ ARAMA AĞAÇLARI

- ❖ İkili arama ağacı algoritmasında ise hesaplama karmaşıklığı $\Theta(\lg n)$ olmaktadır.
- ❖ Bir başka ifadeyle düğüm derinliği ($h = \lg n$) olmaktadır.



İkili Arama Ağaçları Düğüm Derinliği

Bir düğüm derinliği = AĞAÇ ARAYA YERLEŞTİRMESİ için yapılan karşılaştırmalar. Tüm girdi permütasyonları eşit olasılıklı varsayılırsa:

Ortalama düğüm derinliği

$$= \frac{1}{n} E \left[\sum_{i=1}^n \text{(Boğum } i' \text{ yi araya yerleştirmek için gerekli karşılaştırmaların sayısı)} \right]$$

$$= \frac{1}{n} O(n \lg n) \quad (\text{Çabuk sıralama analizi})$$

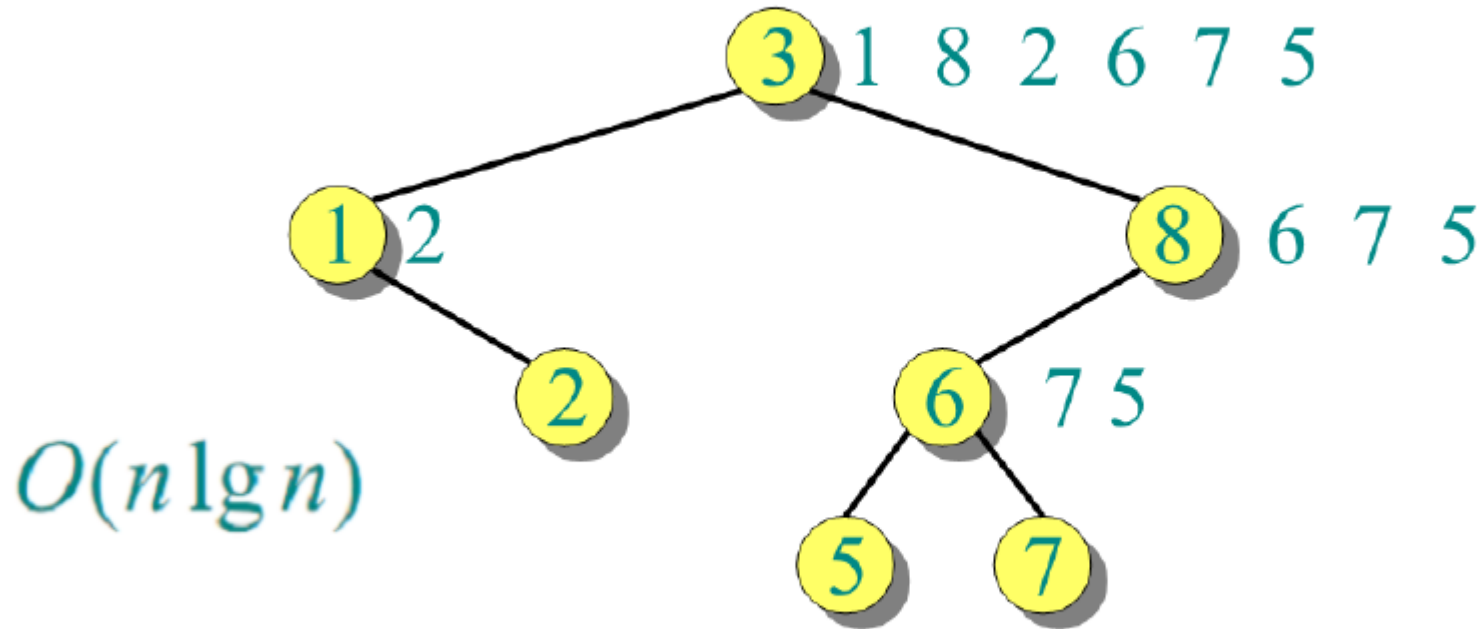
$$= O(\lg n) .$$

* Derinlik neyse o kadar karşılaştırma yaparız.
* Kök düğümde herhangi bir karşılaştırma yapmayız.

* Beklenen ortalama derinlik

İKİLİ ARAMA AĞAÇLARI

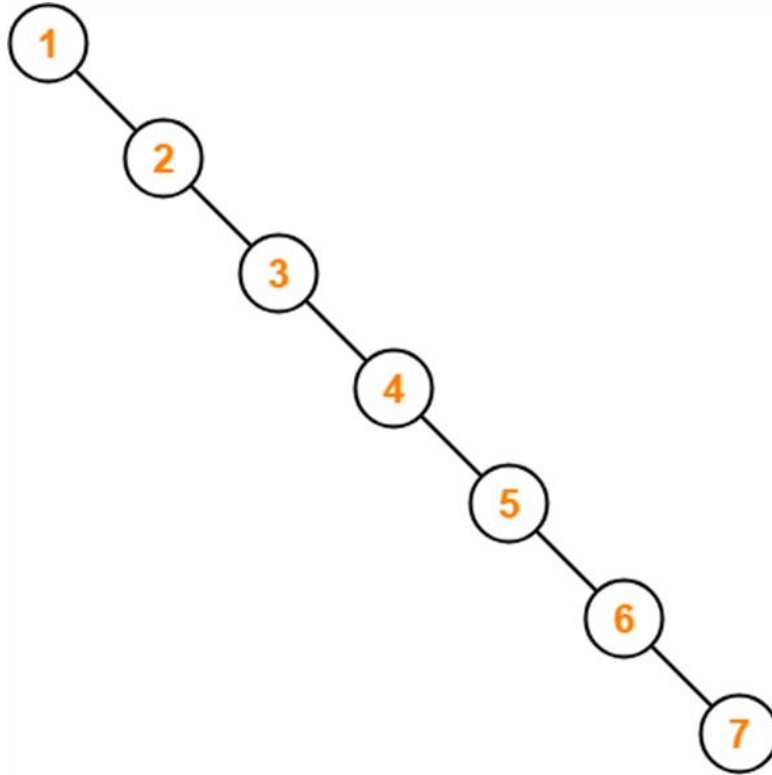
BST sıralaması çabuk sıralama karşılaştırmalarının aynısını, başka bir düzende yapar!



Ağacı oluşturmanın beklenen süresi asimptotik olarak çabuk sıralamanın koşma süresinin aynıdır.

İKİLİ ARAMA AĞAÇLARI: EN KÖTÜ DURUM

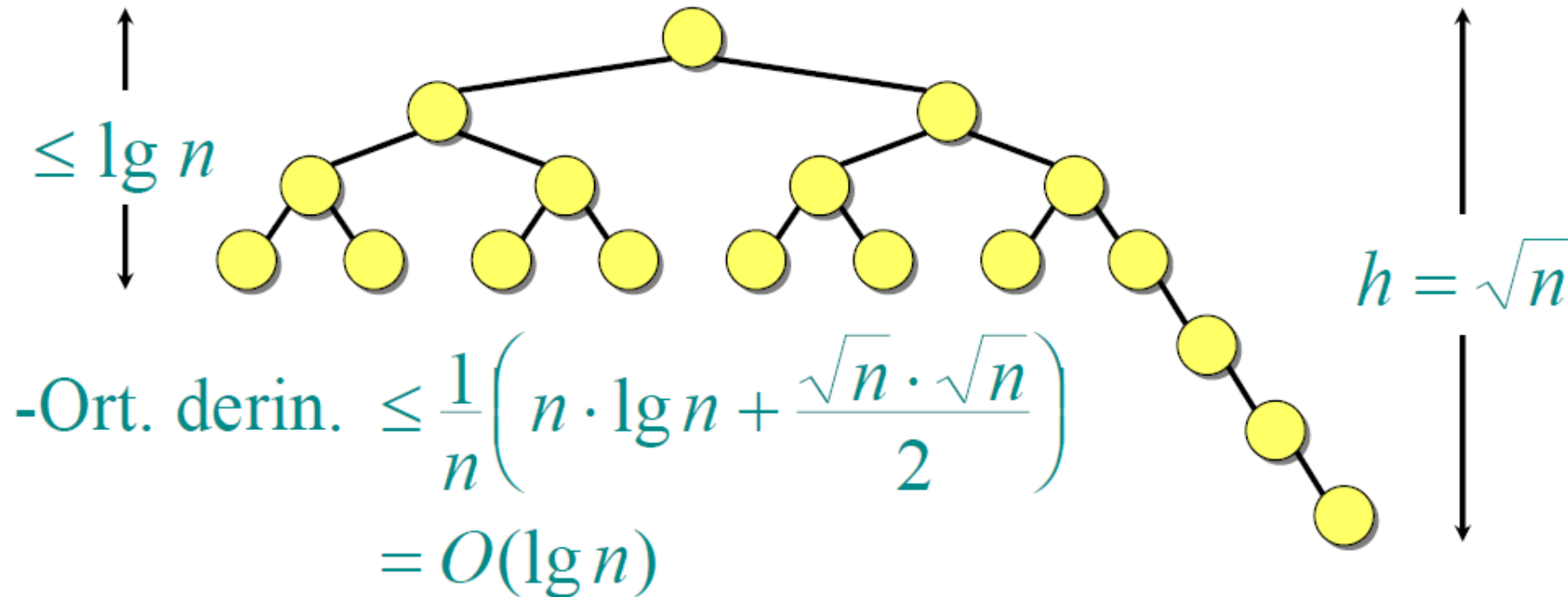
- ❖ İkili arama ağacında veriler tersten sıralanmış bir şekilde geldiğinde hesaplama karmaşıklığı $O(n)$ olmaktadır.
- ❖ Ortaya çıkan ağaç bağlantılı listeye benzemektedir.



İKİLİ ARAMA AĞAÇLARI: BEKLENEN YÜKSEKLİK

Ama, ortalama düğüm derinliğinin rastgele yapılanmış bir ikili arama ağacında (BST) $= O(\lg n)$ olması ağacın beklenen yüksekliğinin de $O(\lg n)$ olduğu anlamına gelmeyebilir (buna rağmen öyledir).

Örnek.



İKİLİ ARAMA AĞAÇLARI: EN KÖTÜ DURUM

- ❖ İkili arama ağacında **ekleme** ve **silme** işlemleri gibi **güncellemeler** yapıldığında ağacın dengesi bozulacaktır.
- ❖ Bu durumda dengeli bir arama ağacı oluşturmak için **çeşitli teknikler** kullanılmaktadır.

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

Orijinali ilk olarak 1972 yılında yapıyı "simetrik ikili B ağaçları" olarak adlandıran **Rudolf Bayer** tarafından bulunmuştur.

Karmaşık ancak çalışma süresi **en kötü durumda bile iyi ve pratikte verimlidir**:

$O(\log n)$ (n ağaçtaki eleman sayısını gösterir) zamanda **arama, ekleme ve çıkarma** işlemleri yapılabilir.

Bilgisayar biliminde karşılaştırılabilir veri parçalarını (sayılar gibi) organize etmek için kullanılabilen özel bir ikili ağaç türüdür.

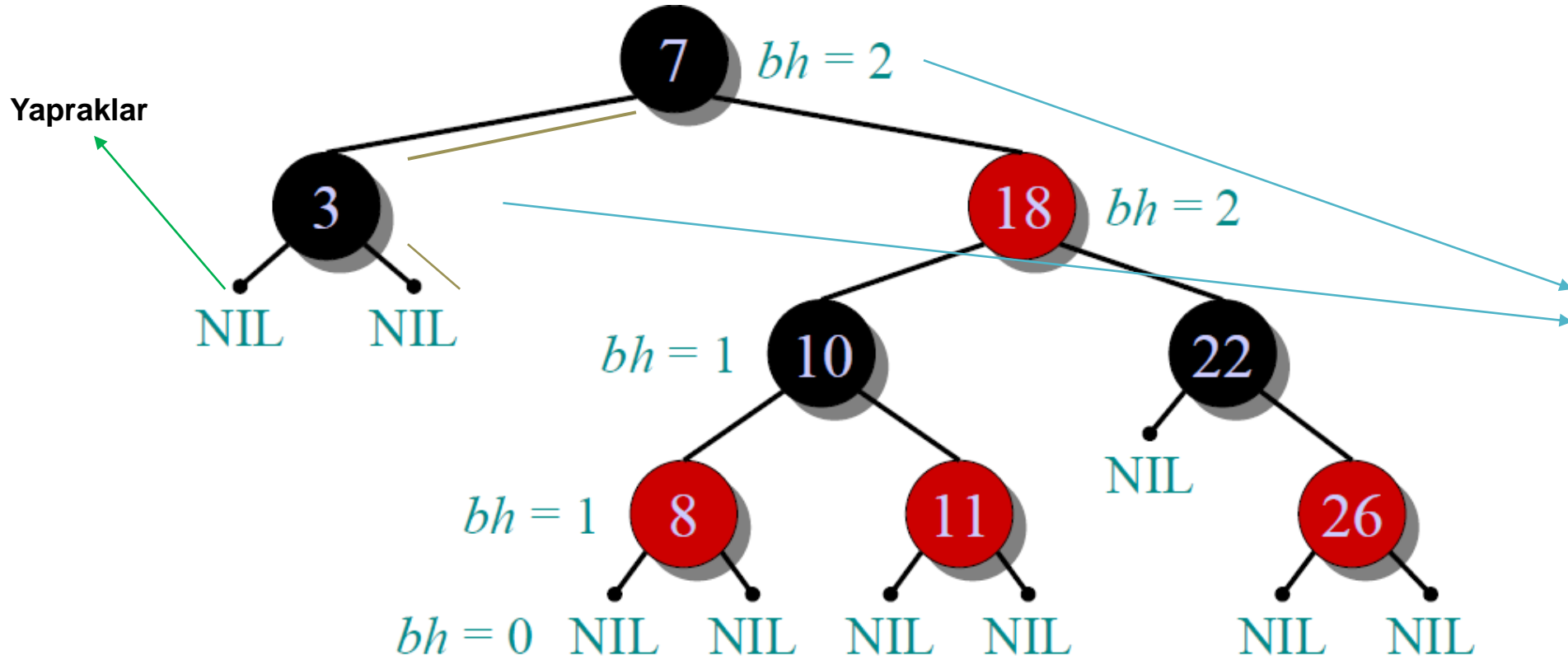
Kırmızı-siyah ağaçlar

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



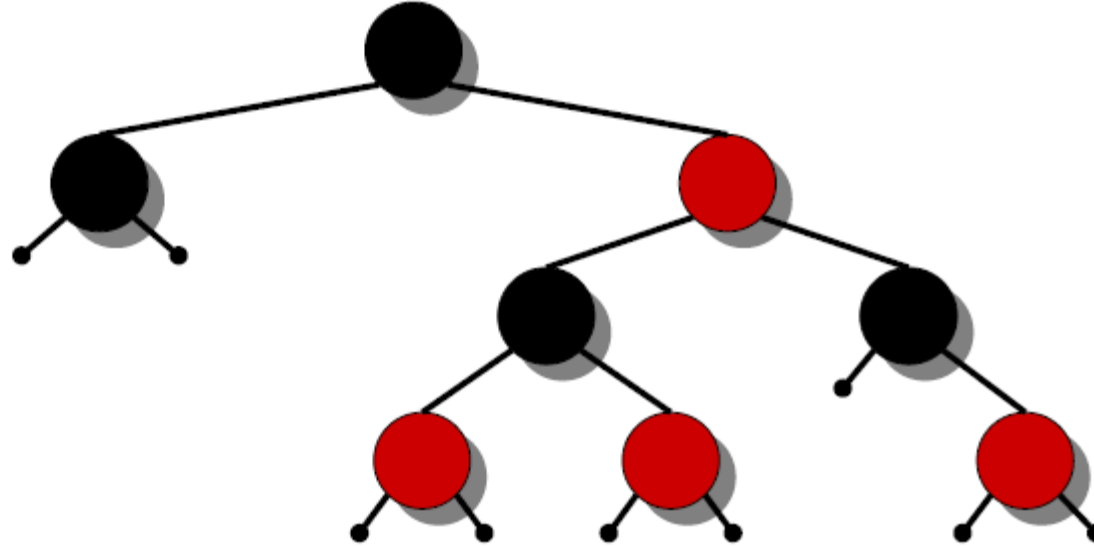
4. Herhangi bir x düğümünden ardıl yaprağa giden basit yollarda aynı sayıda siyah düğüm vardır = *siyah-yükseklik*(x).

Kırmızı-siyah ağacın yüksekliği

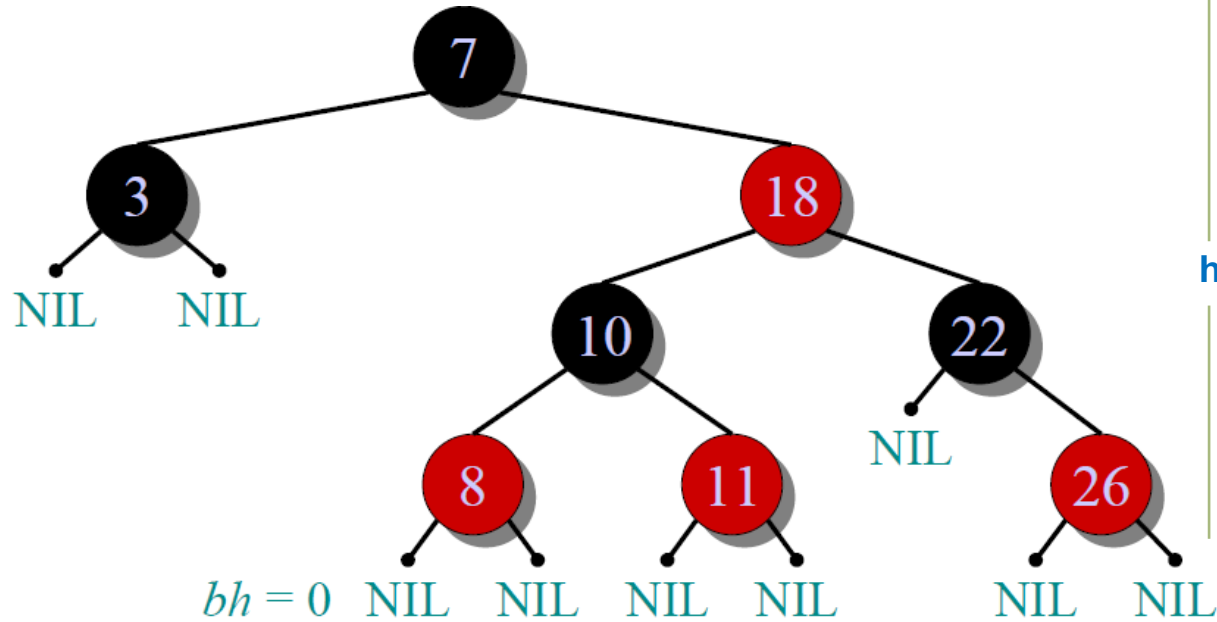
Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği $h \leq 2 \lg(n + 1)$ dir.

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarına yaklaştırın.



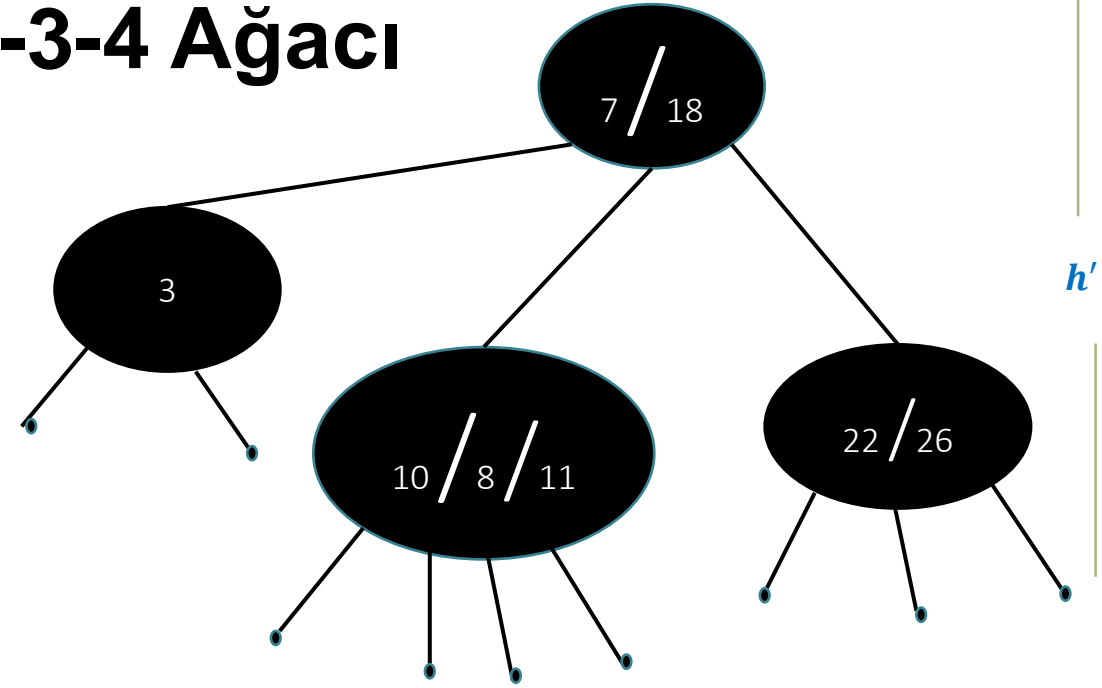
Kırmızı-siyah ağacın yüksekliği



Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği

$$h \leq 2 \lg(n + 1) \text{ dir.}$$

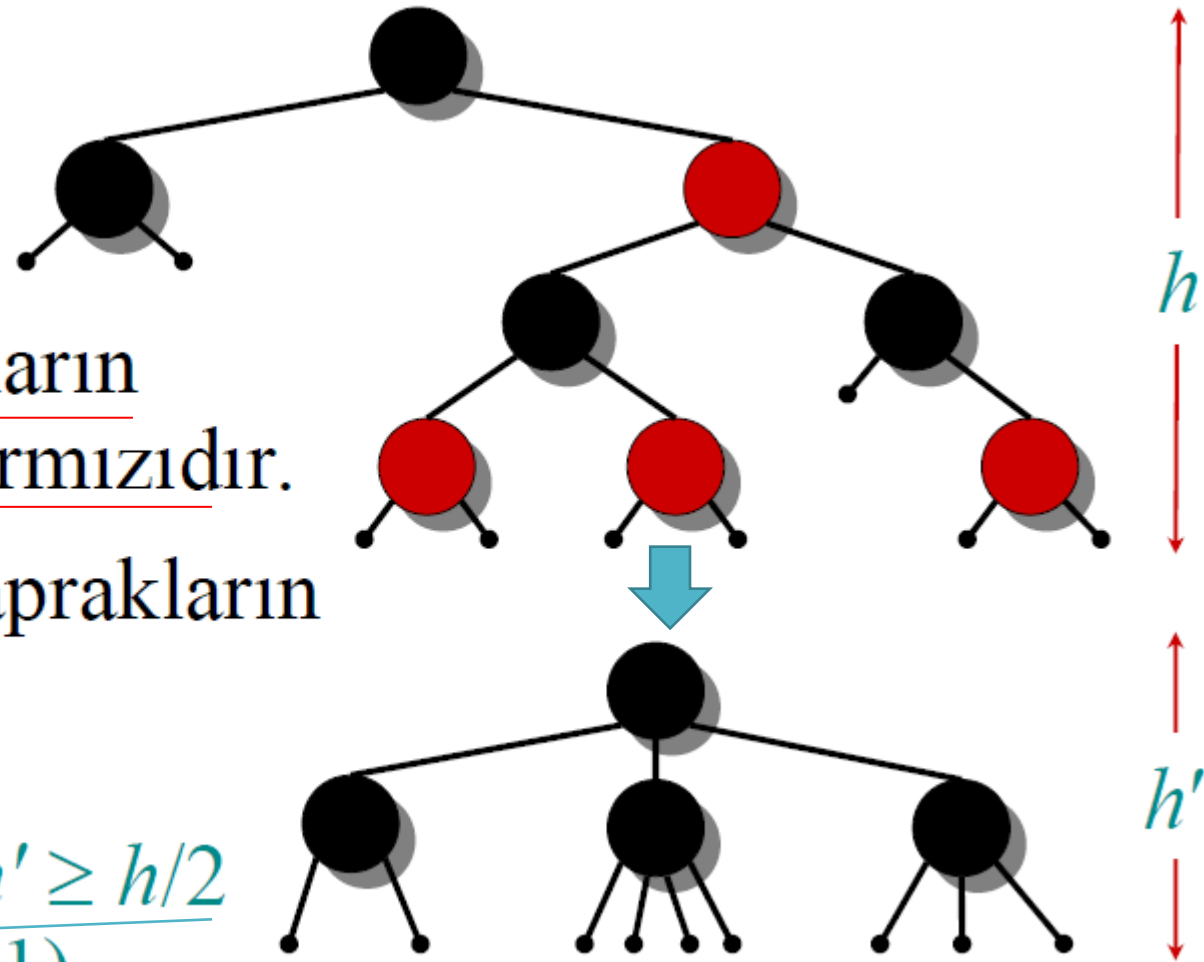
2-3-4 Ağacı



- Her yaprak aynı derinliğe sahip olur (Siyah düğüm sayısı aynıdır. Kökün yüksekliğine eşittir.) 4. maddeden gelir.
- Bu işlem sonucunda oluşan ağacın her düğümünün 2, 3, ya da 4 ardılı olur.
- 2-3-4 ağacının yapraklarının derinliği h' tek biçimlidir.
- Dengeli bir ağaç kurulmuş oldu.

Kırmızı-siyah ağacın yüksekliği kanıt

- 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.

GÜNCELLEŞTİRME İŞLEMLERİ

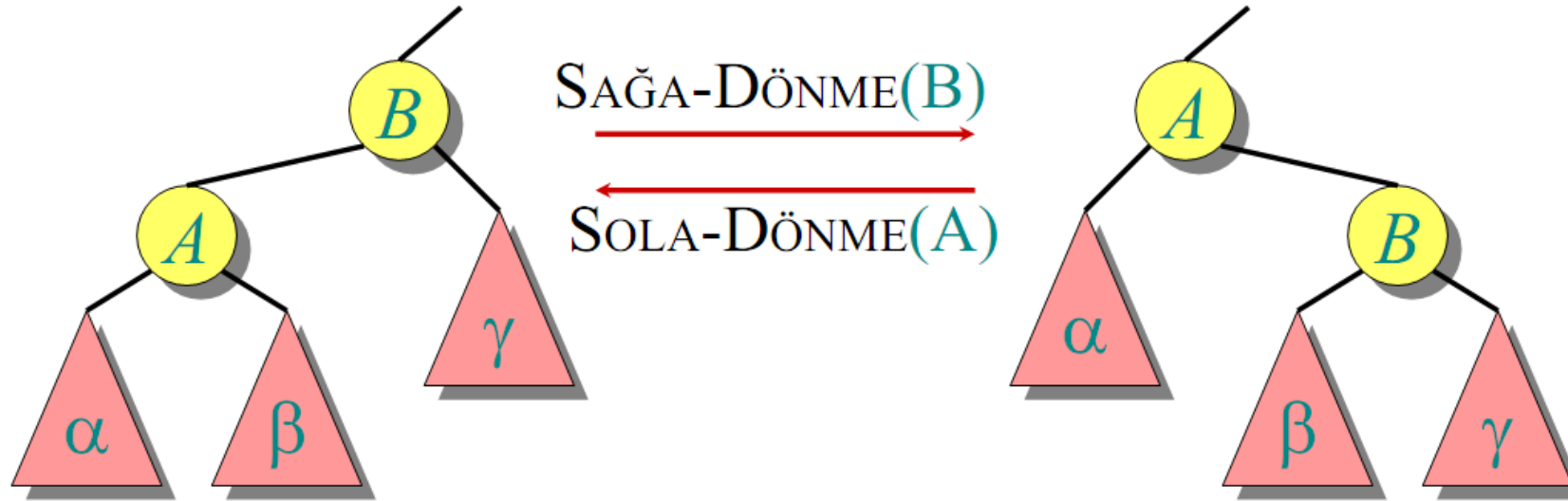
❑Güncelleme işleme zorlu olacaktır.

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

Bu nedenle

- işlemin kendi yapısı,
- renk değişimleri,
- ağacın bağlantılarının *“rotations/rotasyonlar”* kullanılarak ağaç yeniden düzenlenir.

Rotasyonlar / Dönmeler



Rotasyonlar anahtarların sıralı düzenini korurlar:

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

Bir rotasyon $O(1)$ sürede yapılabilir.

Kırmızı-siyah ağaçta araya yerleştirme

FİKİR: Ağaçta x' i araya yerleştirin.

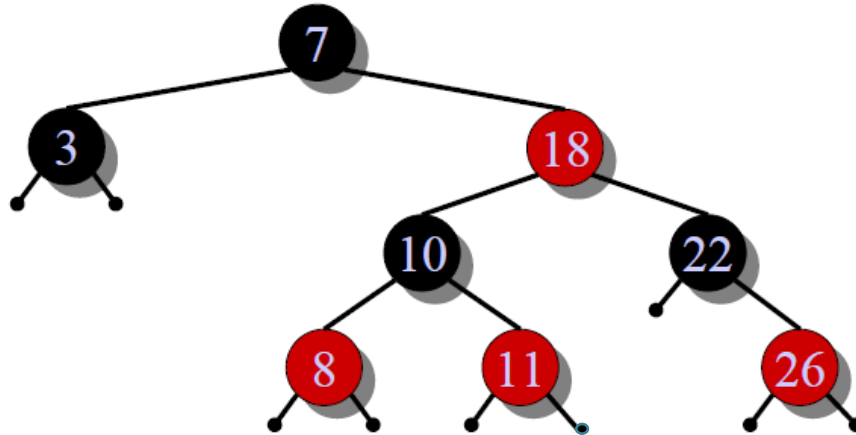
x' i **kırmızı** yapın.

❖ Sadece **kırmızı-siyah** özellik 3 ihlal edilebilir.

İhlali ağaç boyunca yukarı doğru, rotasyonlar ve yeniden renklendirmeyeyle düzelene kadar taşıyın.

Örnek:

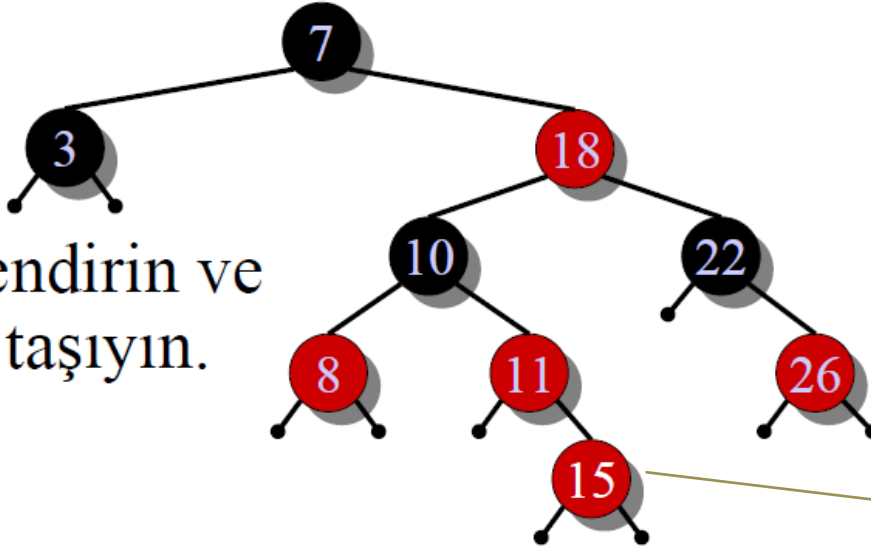
- Ar.Yer. $x = 15$.



Kırmızı-siyah ağaçta araya yerleştirme

Örnek:

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



Fikir:

1. Bir renk seçilir.

Kırmızı renk seçilebilir.

Böylece yoldaki siyah düğümler değişmemiş olur.

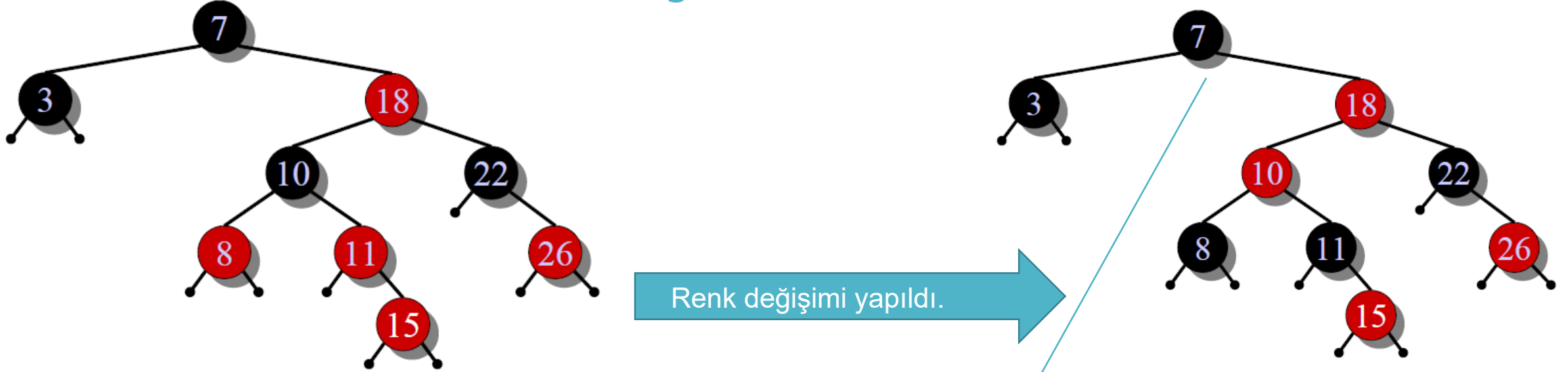
Yalnız kural 3 ihlal edilebilir. (Eğer bir düğüm kırmızı ise, atası siyahtır.)

2. Kural ihlalini yeniden renklendirme yapılarak çözülmeye çalışılır.
3. Renklendirme ile çözülmez ise sağa veya sola rotasyon yapılır.

Kırmızı-siyah ağaçta araya yerleştirme



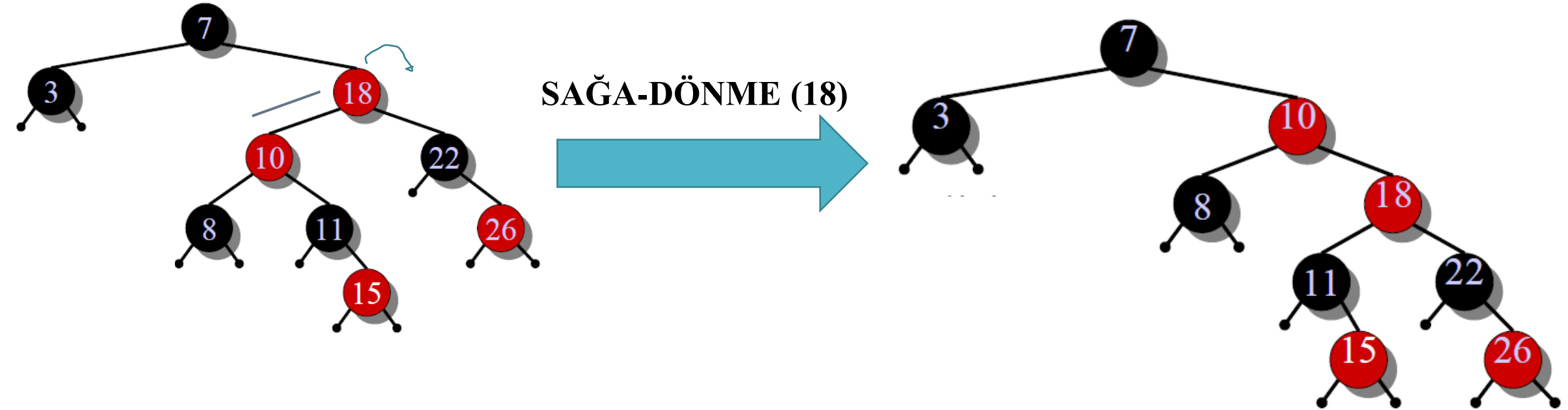
Kırmızı-siyah ağaçta araya yerleştirme



- Başka bir renk değişimi yapılacak durum yok.
- Kök düğümün siyah olarak kalması gerekiyor.
- Eğer 3 kırmızı, 7 siyah yapılırsa kökten yaprağa giden yollarda **siyah düğüm içerme sayısındaki denge bozulacaktır.**
- Rotasyon işlemi yapılması gerekmektedir.

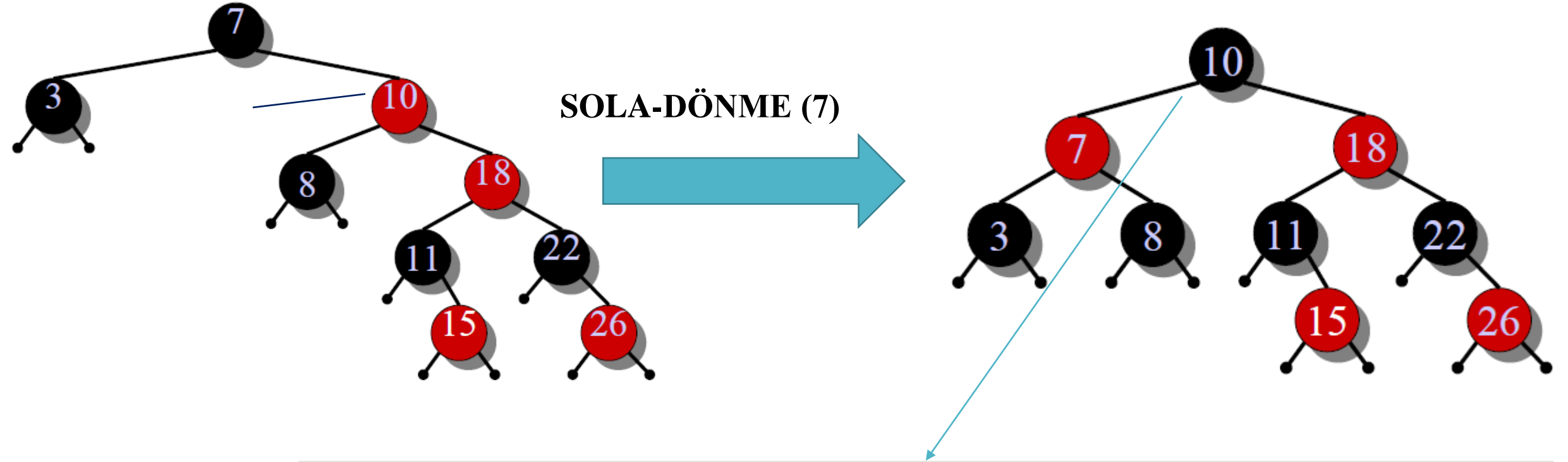
Kural 3 ihlali devam ediyor.

Kırmızı-siyah ağaçta araya yerleştirme



- Kural 3 ihlali var.
- Ayrıca ağaç yine dengesiz bir halde kaldı.
- Tekrar bir döndürme işlemi yapılarak kural 3 ihlali giderilmeye çalışılır.

Kırmızı-siyah ağaçta araya yerleştirme



- Yeniden renklendirme yapılır.
- Böylece kökün siyah olması sağlanır.
- Burada renkler uysa bile en son kontrol yapılır.
 - Her bir yolda siyah düğüm içerme sayısı aynı olup olmadığı kontrol edilir.

Pseudocode-Sözde kod

RB-INSERT(T, x)

 TREE-INSERT(T, x)

$color[x] \leftarrow \text{RED}$ ▷ only RB property 3 can be violated

while $x \neq \text{root}[T]$ and $color[p[x]] = \text{RED}$

do if $p[x] = \text{left}[p[p[x]]]$

then $y \leftarrow \text{right}[p[p[x]]]$

 ▷ $y = \text{aunt/uncle of } x$

if $color[y] = \text{RED}$

then **⟨Case 1⟩**

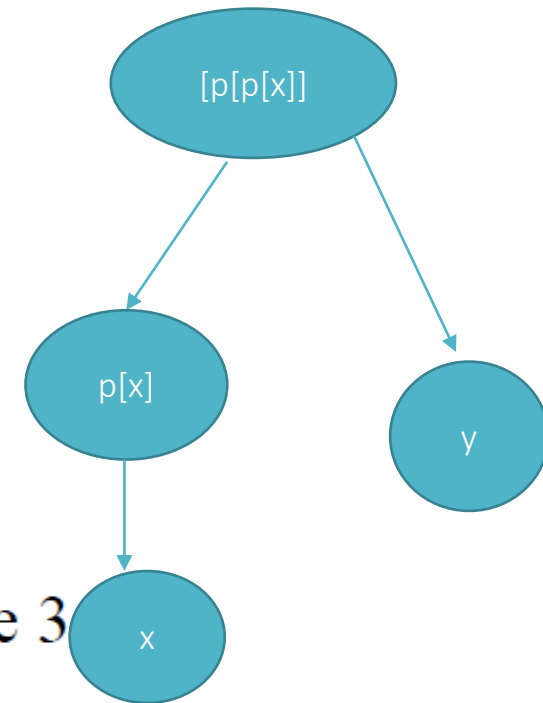
else if $x = \text{right}[p[x]]$

then **⟨Case 2⟩** ▷ Case 2 falls into Case 3



⟨Case 3⟩

else **⟨“then” clause with “left” and “right” swapped⟩**

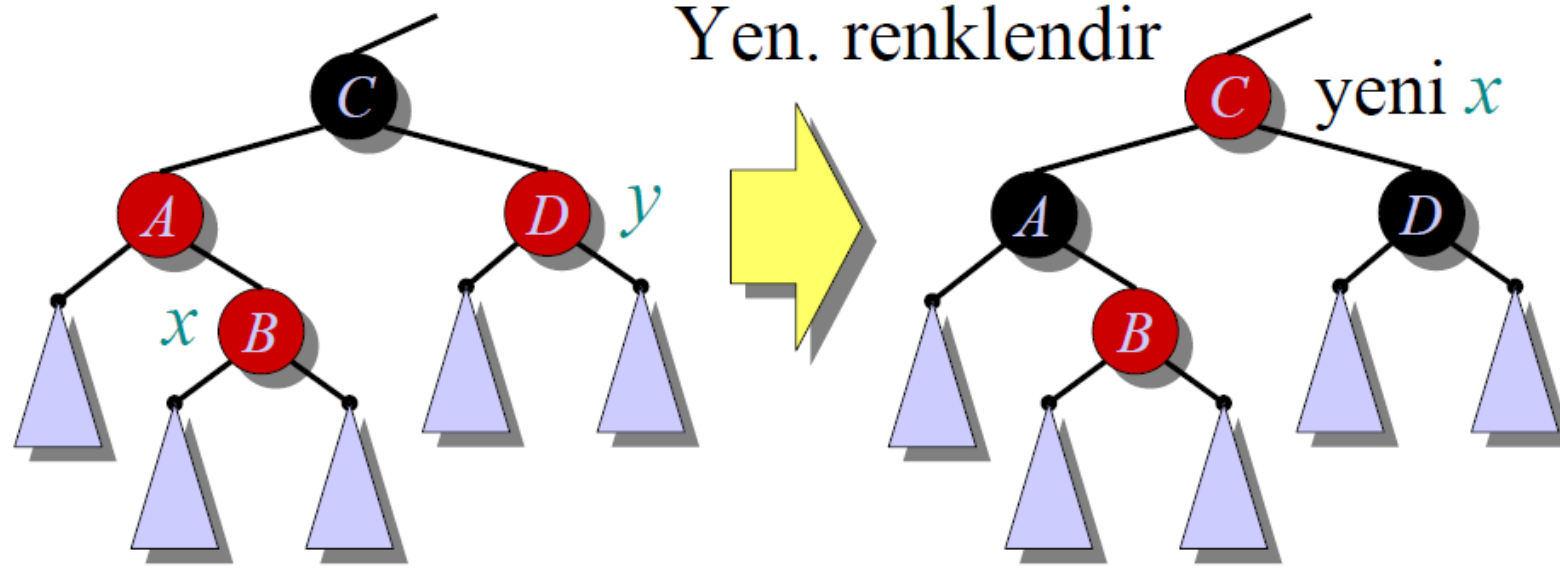
$color[\text{root}[T]] \leftarrow \text{BLACK}$



Grafik Simgelem

-  siyah kökü olan bir ağacı tanımlasın.
-  'ın tümünün siyah-yükseklikleri aynıdır.

Durum 1

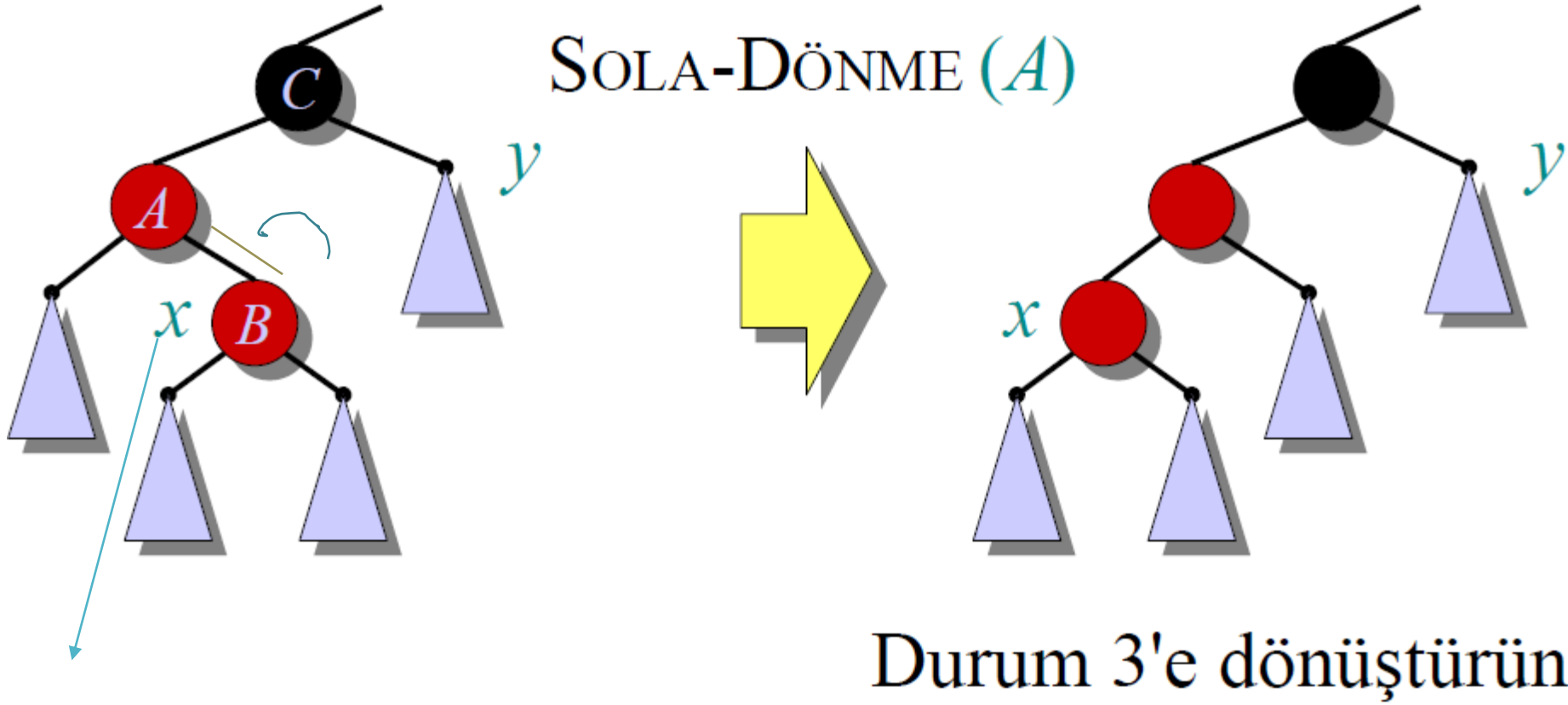


(veya, A 'nın ardılları
yer değiştirir.)

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

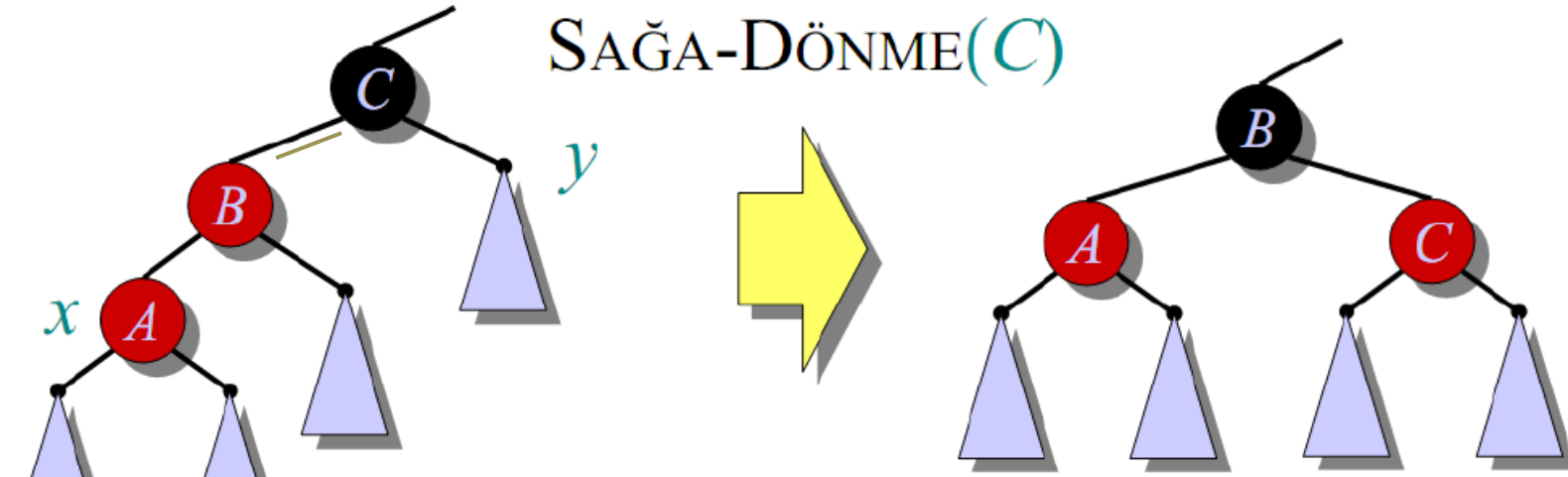
* Herhangi bir yolun içerdiği siyah düğüm sayısı aynı oldu. (Kural 4 sağlandı.)

Durum 2



* X sağ ardıl olduğunu biliyorum. Dolayısıyla sola döndürerek kural 3 ihlalinin durumdan kurtulmaya çalışırım.

Durum 3



Bitti! RB (KIRMIZI-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.

Kaynakça

Algoritmalar : Prof. Dr. Vasif NABİYEV, Seçkin Yayıncılık

Algoritmalara Giriş : Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, Palme YAYINCILIK

Algoritmalar : Robert Sedgewick , Kevin Wayne, Nobel Akademik Yayıncılık

M.Ali Akcayol, Gazi Üniversitesi, Algoritma Analizi Ders Notları

Doç. Dr. Erkan TANYILDIZI, Fırat Üniversitesi, Algoritma Analizi Ders Notları

<http://www.bilgisayarkavramlari.com>