7.Hafta
Dengeli Arama Ağaçları
(Red - Black Tree)

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

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

Red - Black Tree

- **Kırmızı-siyah ağaç** bilgisayar biliminde **bir çeşit kendini-dengeleyen** ikili arama ağacı veri yapısıdır.
- Orijinali ilk olarak 1972 yılında yapıyı "simetrik ikili B-ağaçları" olarak adlandıran Rudolf Bayer tarafından bulunmuştur. Bugünkü ismini 1978 yılında Leo J. Guibas ve Robert Sedgewick tarafından yayımlanan bir makaleyle almıştır.
- 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 yapabilir.

- **Bir kırmızı-siyah ağaç, bilgisayar biliminde karşılaştırılabilir veri parçalarını (sayılar gibi) organize etmek için kullanılabilen özel bir ikili ağaç türüdür.**

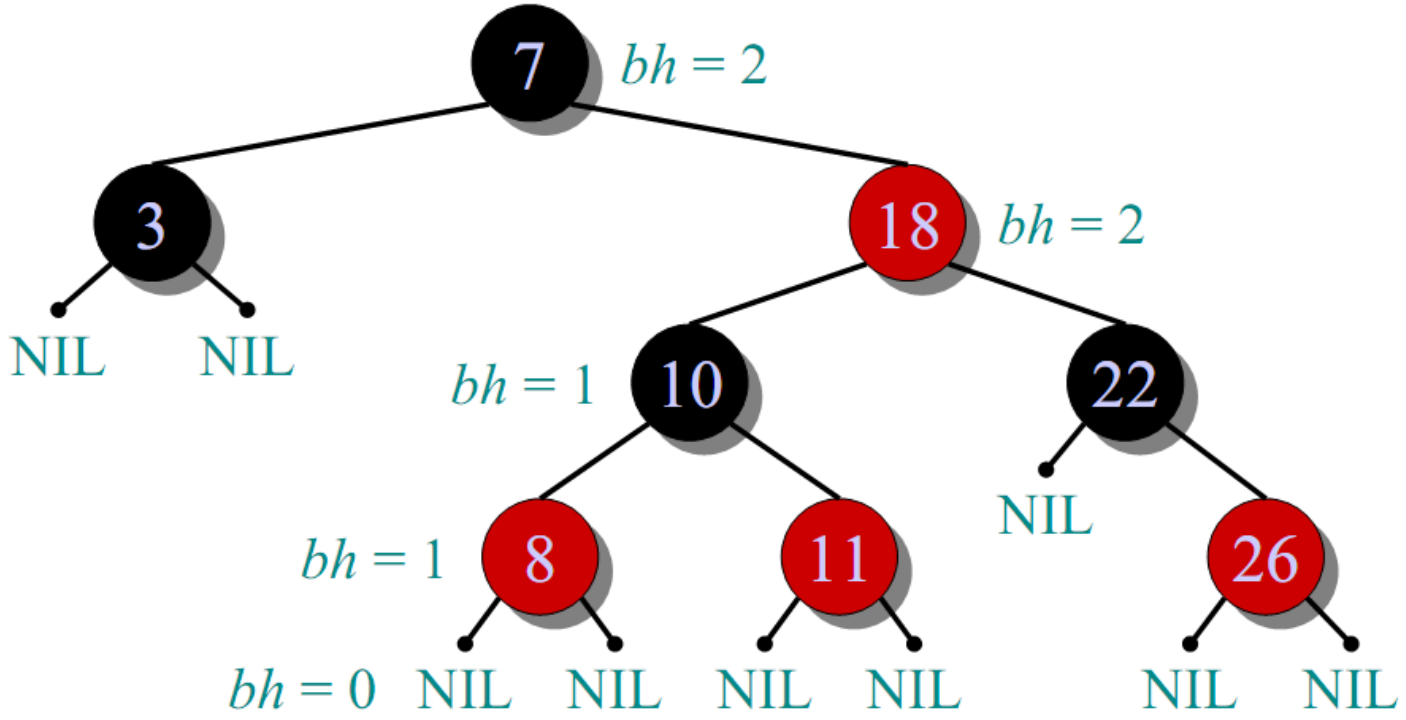
Red - Black Tree

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

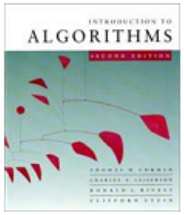
Kırmızı-siyah özellikler:

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



Herhangi bir x düğümünden ardıl yaprağa giden basit yollarda aynı sayıda siyah düğüm vardır $bh = siyah\text{-}yükseklik(x)$.



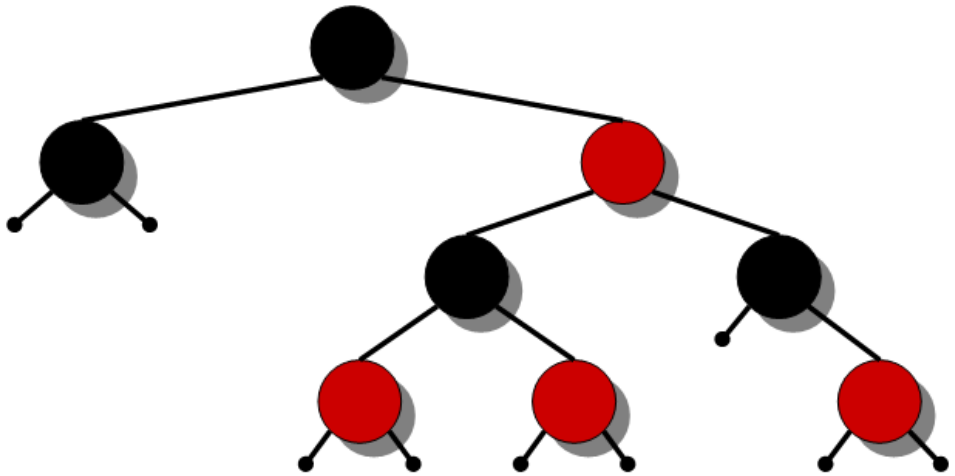
Kırmızı-siyah ağacın yüksekliği

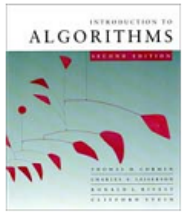
Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği $h \leq 2 \lg(n + 1)$ dir.

Kanıt.

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarına yaklaştırın.





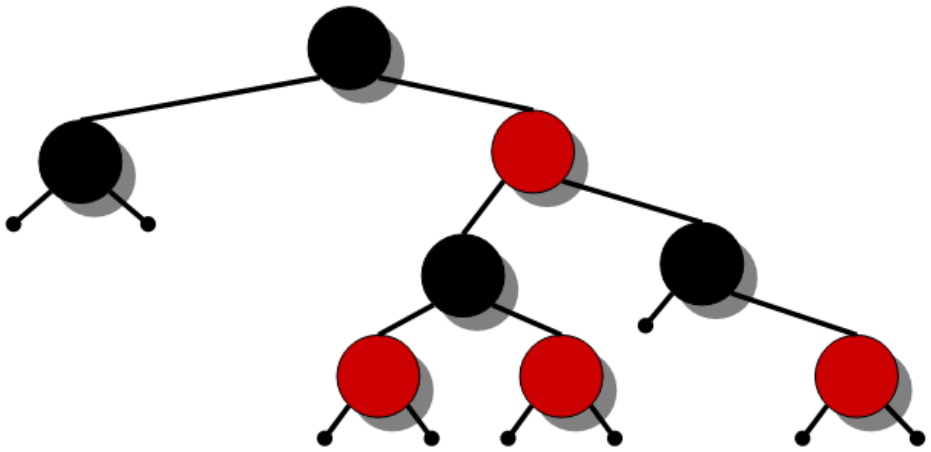
Kırmızı-siyah ağacın yüksekliği

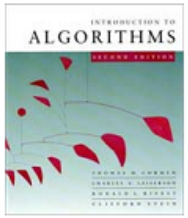
Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği $h \leq 2 \lg(n + 1)$ dir.

Kanıt

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarına yaklaştırın.





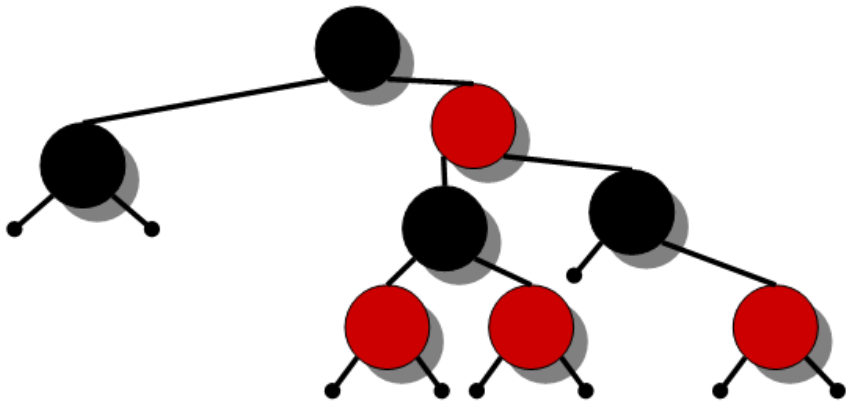
Kırmızı-siyah ağacın yüksekliği

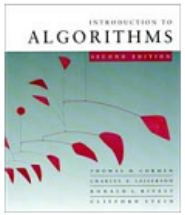
Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği $h \leq 2 \lg(n + 1)$ dir.

Kanıt.

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarına yaklaştırın.





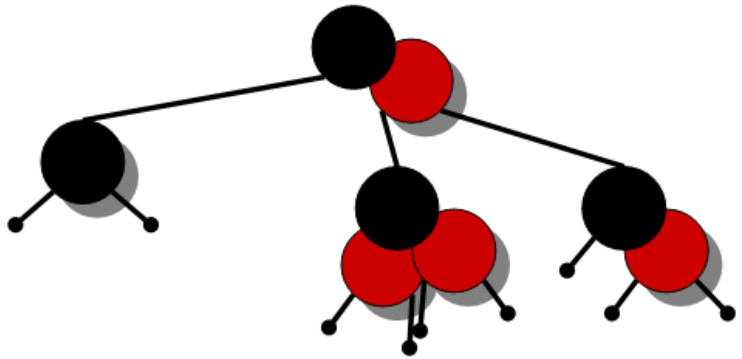
Kırmızı-siyah ağacın yüksekliği

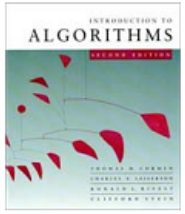
Teorem. n anahtarlı bir kırmızı-siyah ağacın yüksekliği $h \leq 2 \lg(n + 1)$ dir.

Kanıt.

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarıyla birleştirin.





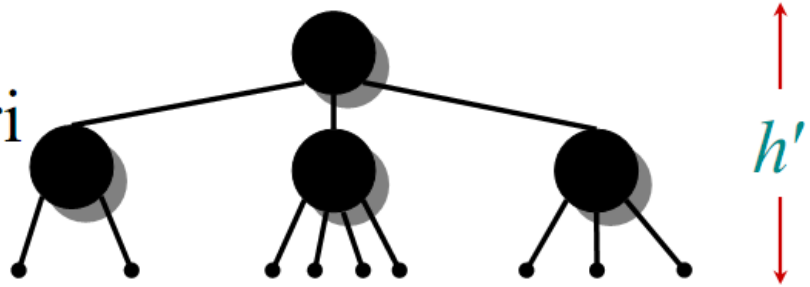
Kırmızı-siyah ağacın yüksekliği

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

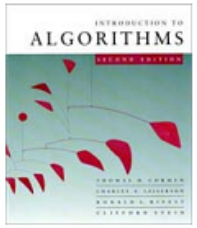
Kanıt. (Kitap tümevarımı kullanıyor. Dikkatle okuyun.)

SEZGİ YÖNTEMİ:

- Kırmızı düğümleri siyah atalarıyla bütünleştirin.

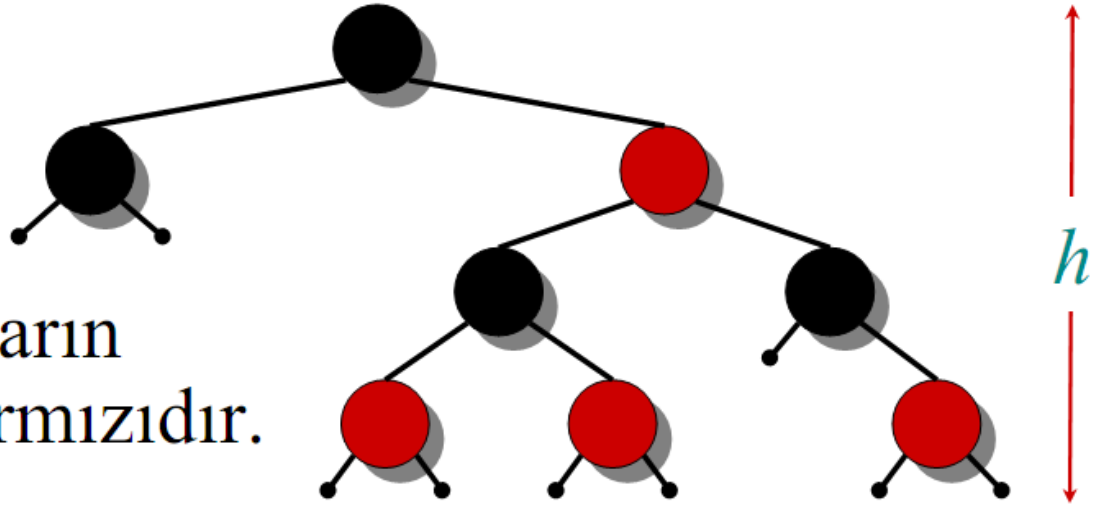


- Bu işlem sonucunda oluşan ağacın her düğümünün 2, 3, ya da 4 ardılı olur.
- 2-3-4 ağacının yapraklarının derinliği h' tekbiçimlidir.



Kanıtlama (devamı)

- Elimizde $h' \geq h/2$ olur, çünkü her yoldaki yaprakların en çok yarısı kırmızıdır.

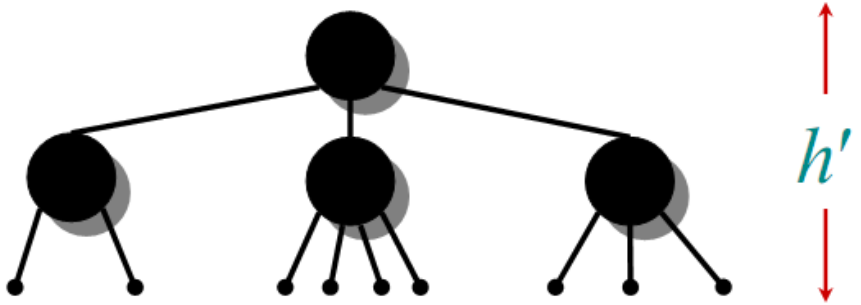


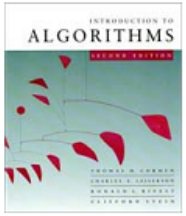
- Her ağaçtaki yaprakların sayısı: $n + 1$

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

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

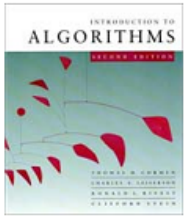
$\Rightarrow h \leq 2 \lg(n + 1).$ □





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.

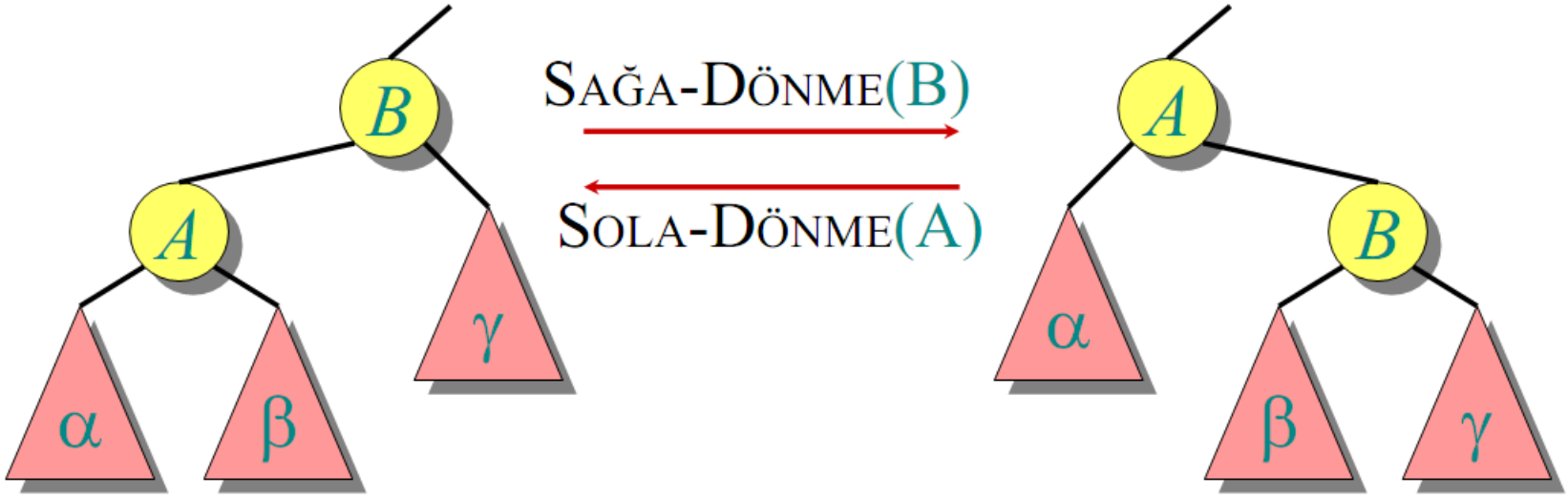


Değiştirme işlemleri

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.

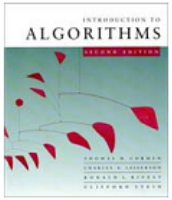
Rotasyonlar / Dönmeler



Rotasyonlar anahtarların sıralı düzenini korurlar:

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

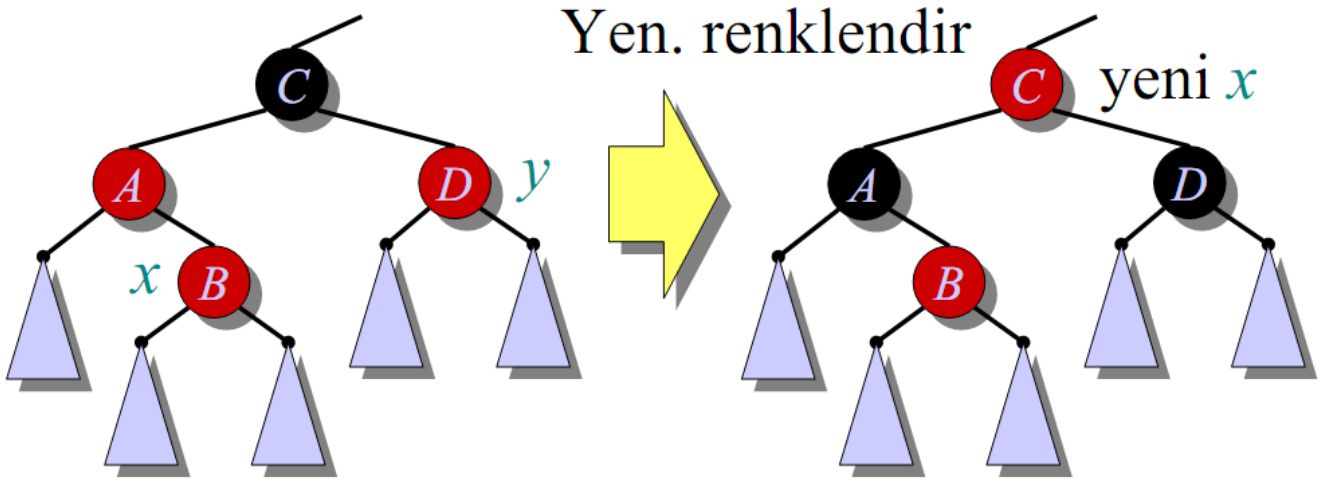
Bir rotasyon $O(1)$ sürede yapılabilir.



Grafik simgelem

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

Durum 1:

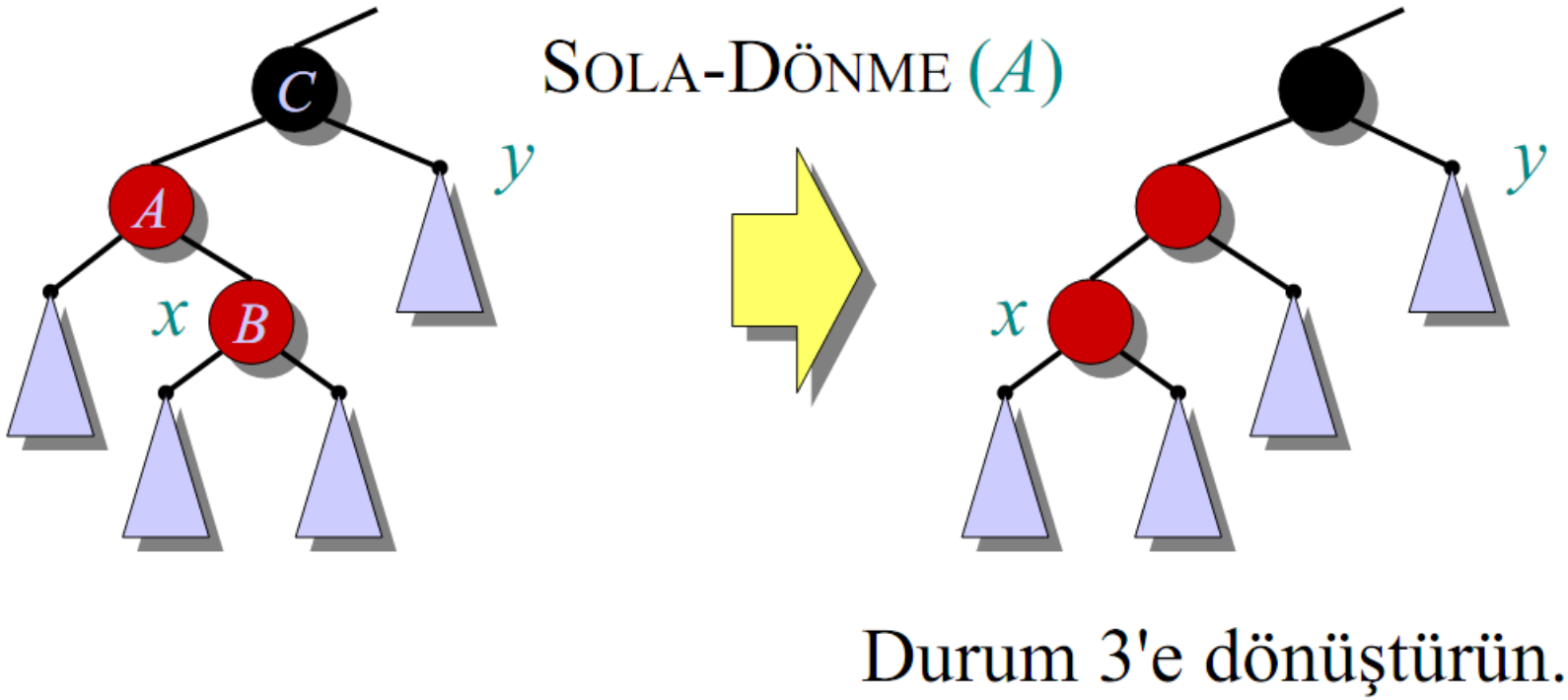


(veya, *A*'nın ardılları yer değiştirir.)

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

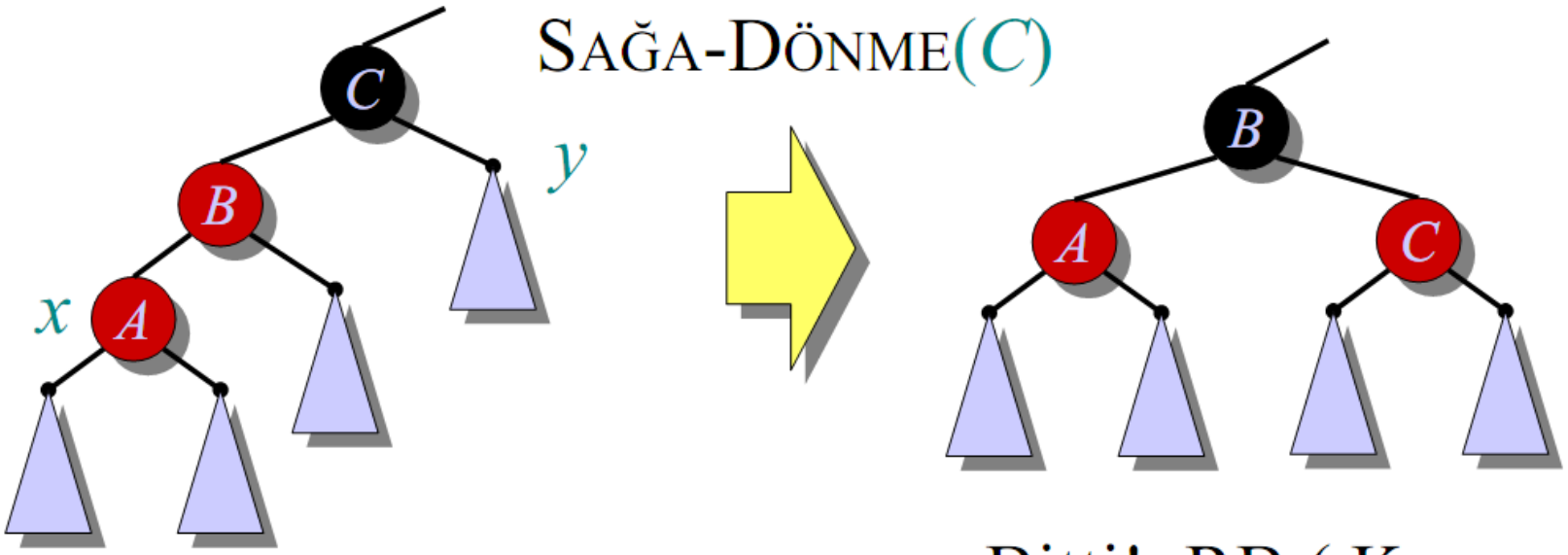
- *Özellik 3 bozuldu. Kırmızı düğümün çocukları siyah olmak zorunda. C düğümünün çocuklarının ikisi de kırmızı olduğundan döndürme işlemi yapılmadan yeniden renklendirilecek. C kırmızı ve çocukları siyah. (Eğer C kök olsaydı o da siyah olacaktı)*

Durum 2:



- *Ağaçta siyah düğümlerin sayısı bozulduysa (bh) veya C'nin bir siyah bir kırımızı çocuğu var ise döndürme işlemi gerçekleştirilecek. Yeni dönüşüm Durum 3 'ü meydana getirir.*

Durum 3:



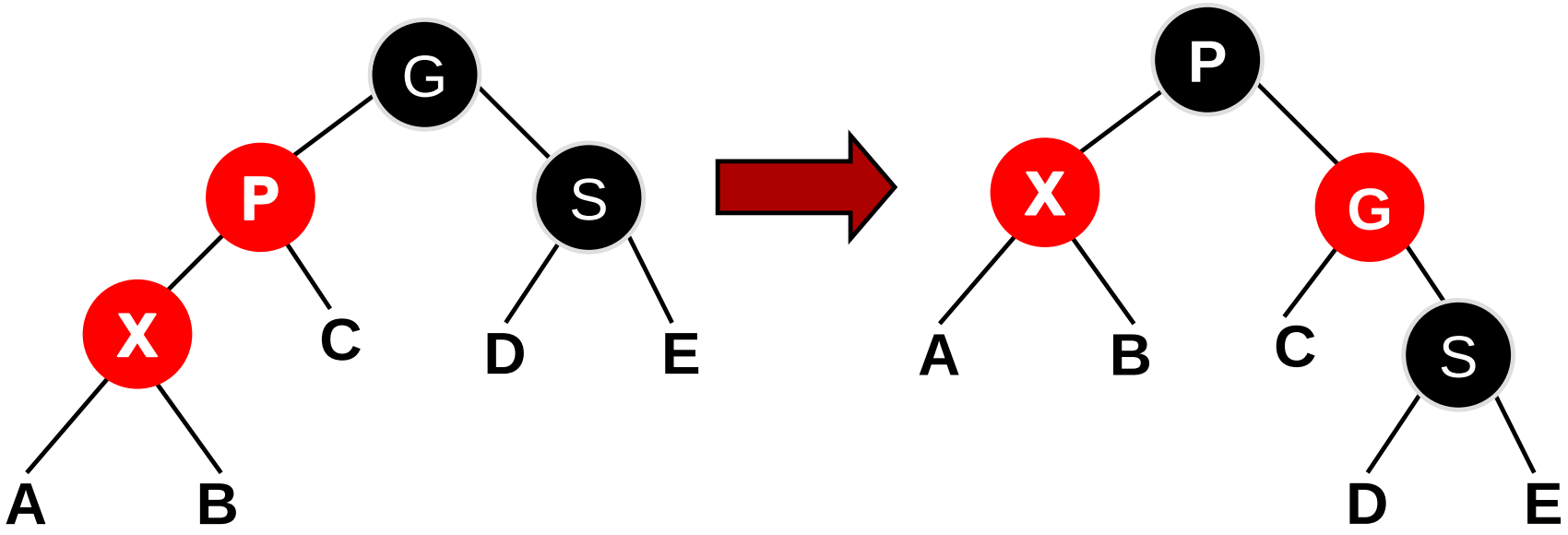
Bitti! RB (Kırmızı-siyah) 3. özelliğin ihlali artık mümkün değil.

- *Özellik 3 ihlali devam ettiğinden yeniden döndürme işlemi ve renklendirme yapılacak.*

Tek Döndürme(Single Rotation)

- Eklemeden sonraki durum:
  - Ardışık red (P & X)
  - P'nin kardeşi S black
  - X **dış** düğüm (left-left veya right-right)

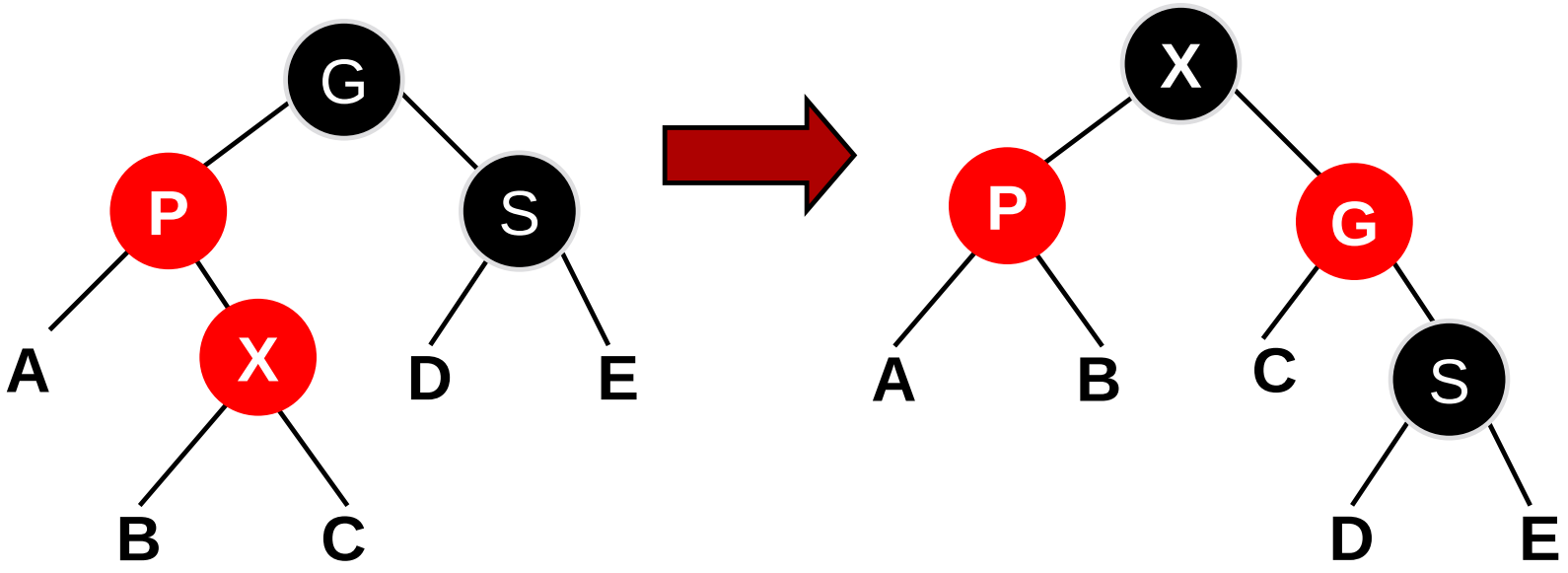
X: Yeni Düğüm
P: Ebeveyn
S: Kardeş
G: Ata



Çift Döndürme (Double Rotation)

- Eklemeden sonraki durum:
  - Ardışık red (P & X)
  - P'nin kardeşi S black
  - X **iç** düğüm (left-right veya left)

X: Yeni Düğüm
P: Ebeveyn
S: Kardeş
G: Ata



Pseudocode

```
RB-INSERT(T, x)
    TREE-INSERT(T, x)
    color[x] ← RED      ▷ only RB property 3 can be violated
    while x ≠ root[T] and color[p[x]] = RED
        do if p[x] = left[p[p[x]]
            then y ← 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]] ← BLACK
```

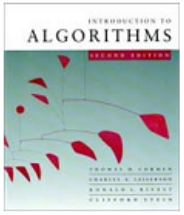
```
RB_Insert(x)
  Tree_Insert(x);
  x.color = RED;
  // 3. kural ihlali,
  while (x!=root && x.p.color == RED)
    if (x.p == x.p.p.left)
      y = x.p.p.right;
      if (y.color == RED)
          x.p.color = BLACK;
          y.color = BLACK;
          x.p.p.color = RED;
          x = x.p.p;
      else   // y.color == BLACK
          if (x == x.p.right)
              x = x.p;
              leftRotate(x);
          x.p.color = BLACK;
          x.p.p.color = RED;
          rightRotate(x.p.p);

   else    // x.p == x.p.p.right
      (yukarıdakine benzer, fakat "right" ve "left" yer değiştirecek)
```

Case 1: y= aunt/uncle of x is RED

Case 2

Case 3

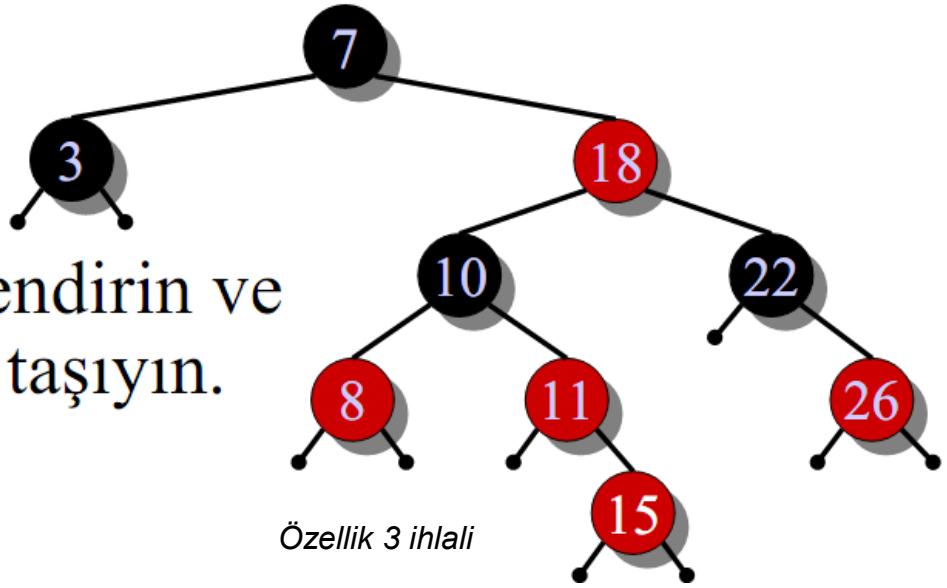


Kırmızı-siyah ağaçta araya yerleştirme

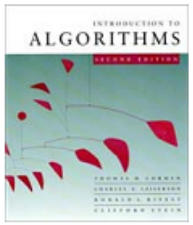
FİKİR: Ağaçta x' i araya yerleştirin. x' i kırmızı yapın. Sadece kırmızı-siyah özellik 3 ihlal edilebilir. İhlali ağaç boyunca yukarı doğru, rotasyonlar ve yeniden renklendirmeyle düzelene kadar taşıyın.

Örnek:

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



Özellik 3 ihlali



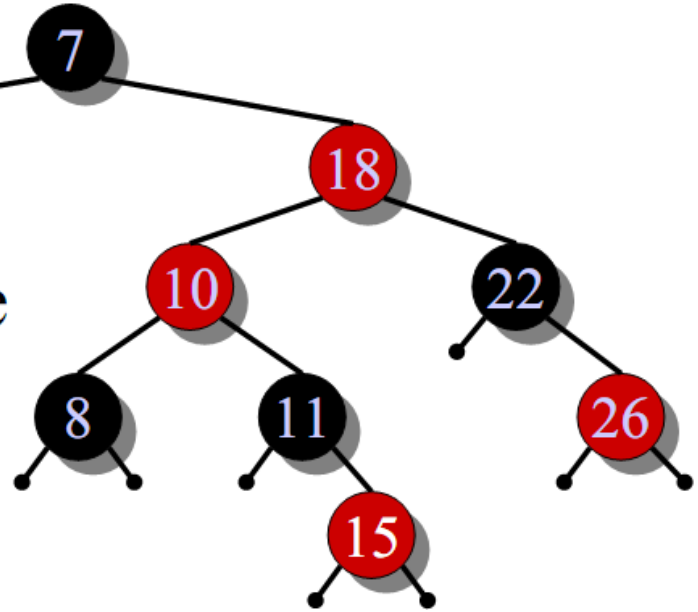
Kırmızı-siyah ağaçta araya yerleştirme

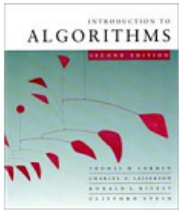
FİKİR: Ağaçta x' i araya yerleştirin. x' i kırmızı yapın. Sadece kırmızı-siyah özellik **3** ihlal edilebilir. İhlali ağaç boyunca yukarı doğru, rotasyonlar ve yeniden renklendirmeyle düzelene kadar taşıyın.

Örnek:

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

Özellik 3 ihlali devam ettiğinden yeniden döndürme işlemi ve renklendirme yapılacak.



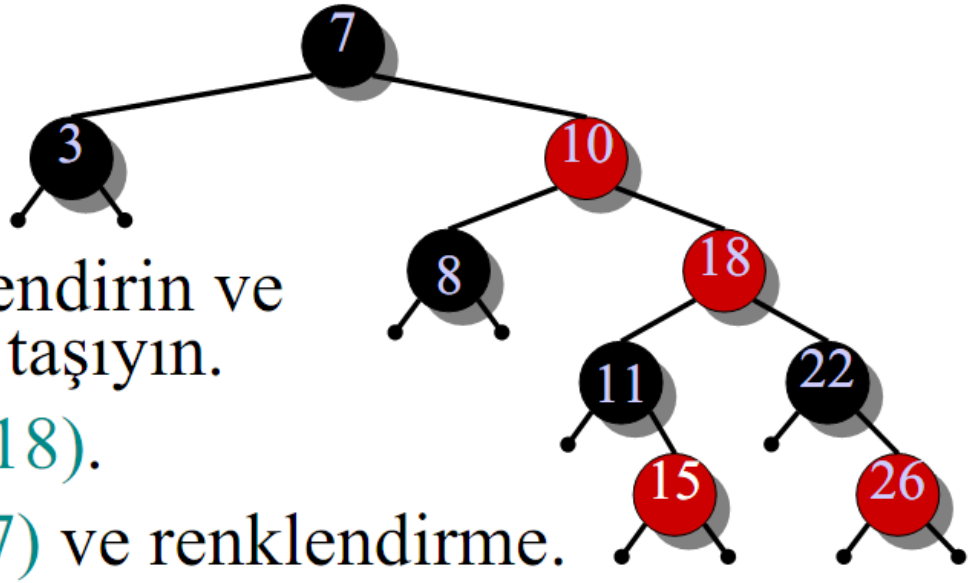


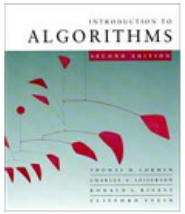
Kırmızı-siyah ağaçta araya yerleştirme

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.



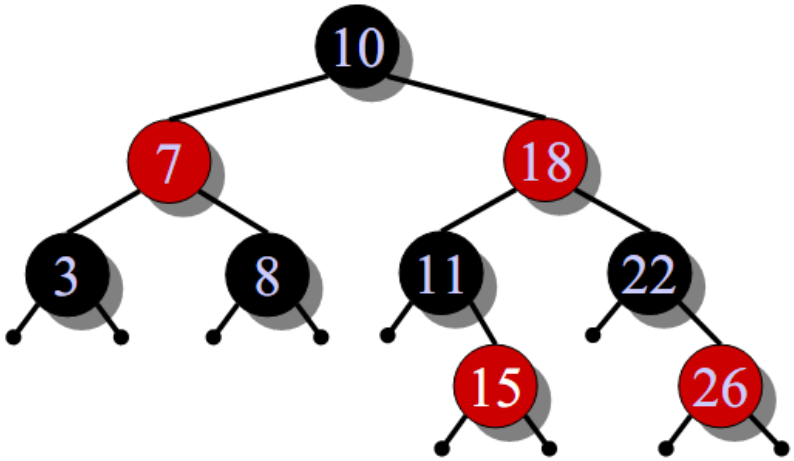


Kırmızı-siyah ağaçta araya yerleştirme

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.



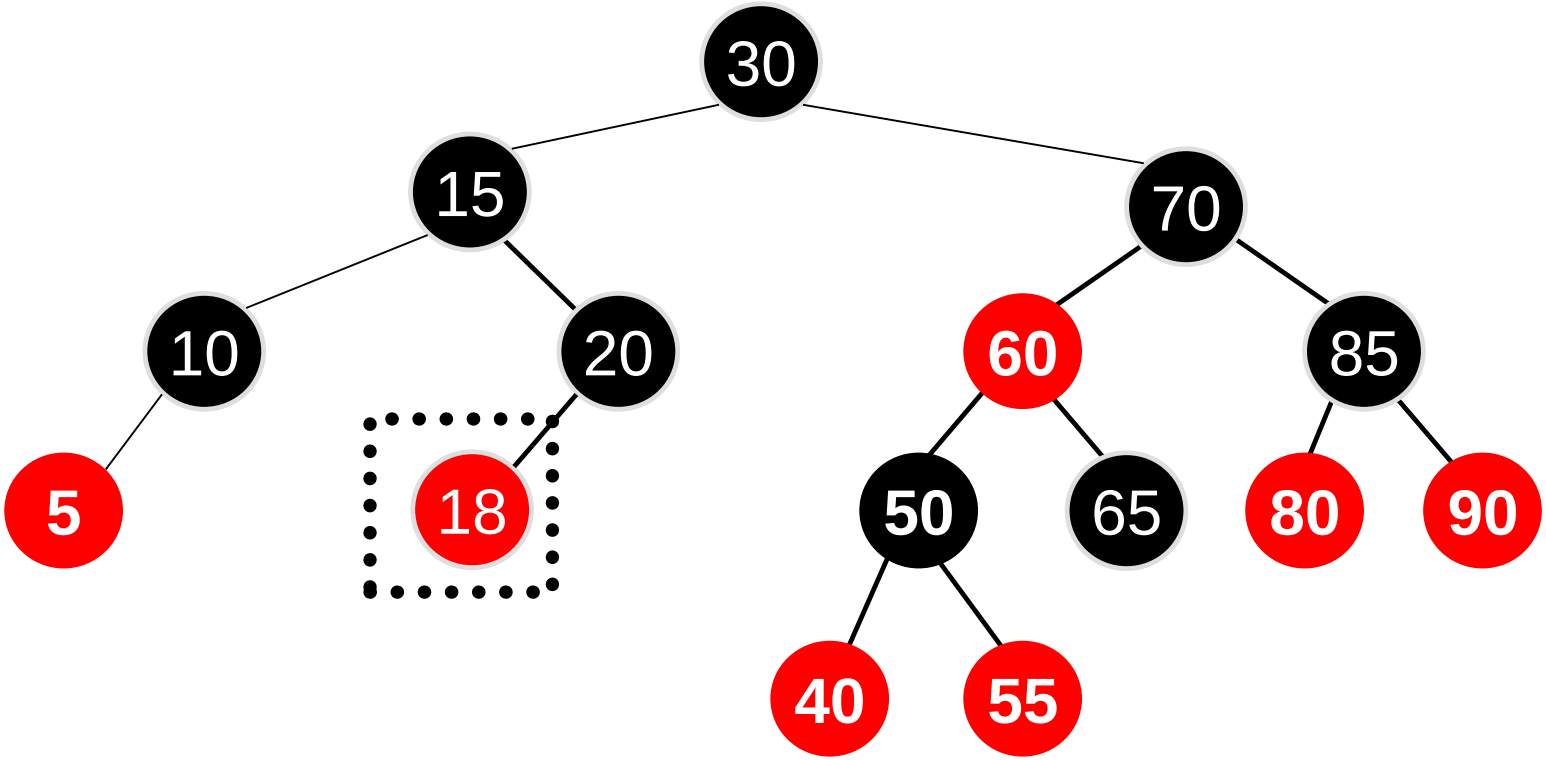
Çözümleme

- 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.

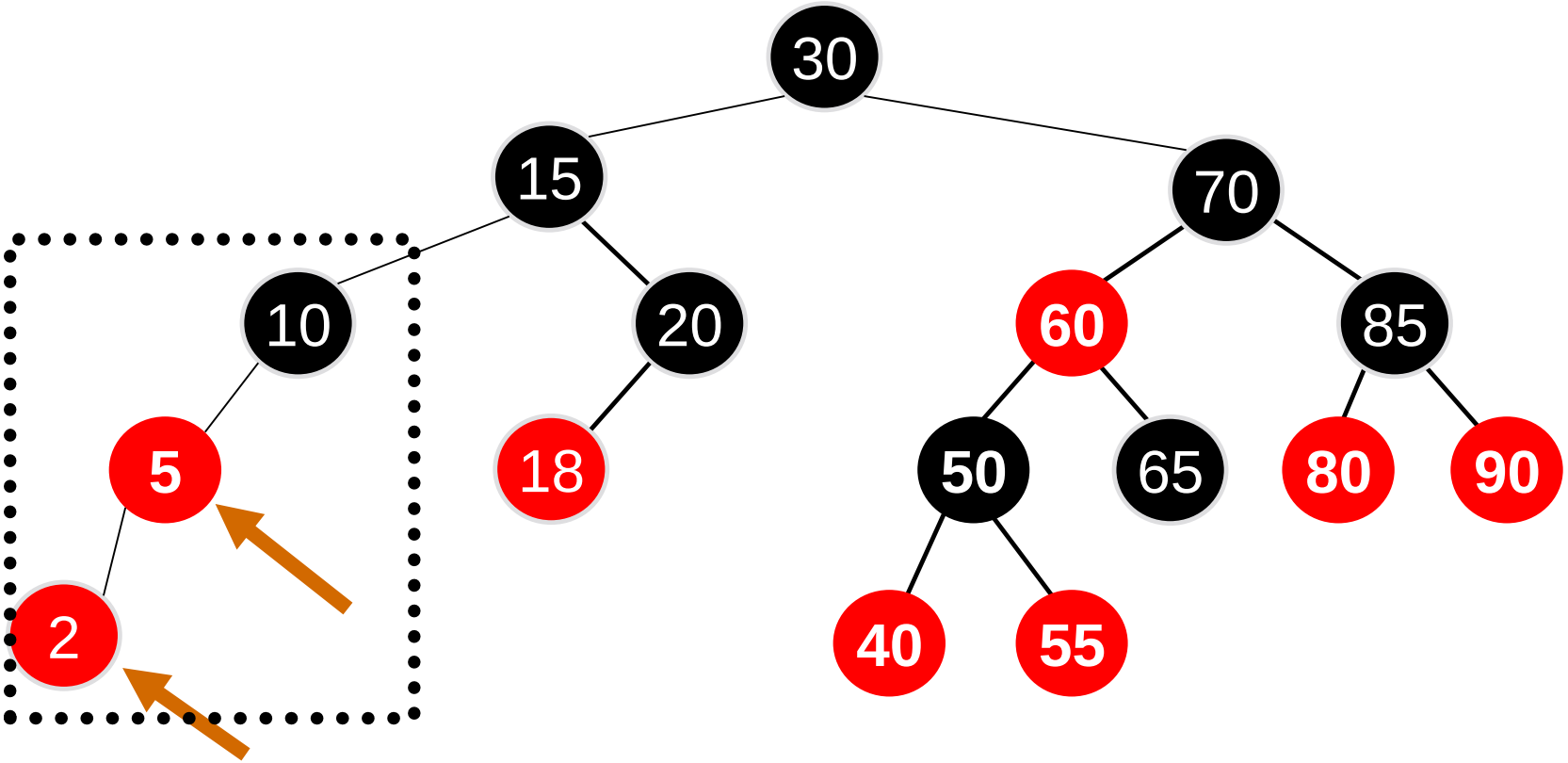
Yürütüm süresi: $O(\lg n)$ ve $O(1)$ rotasyon.

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

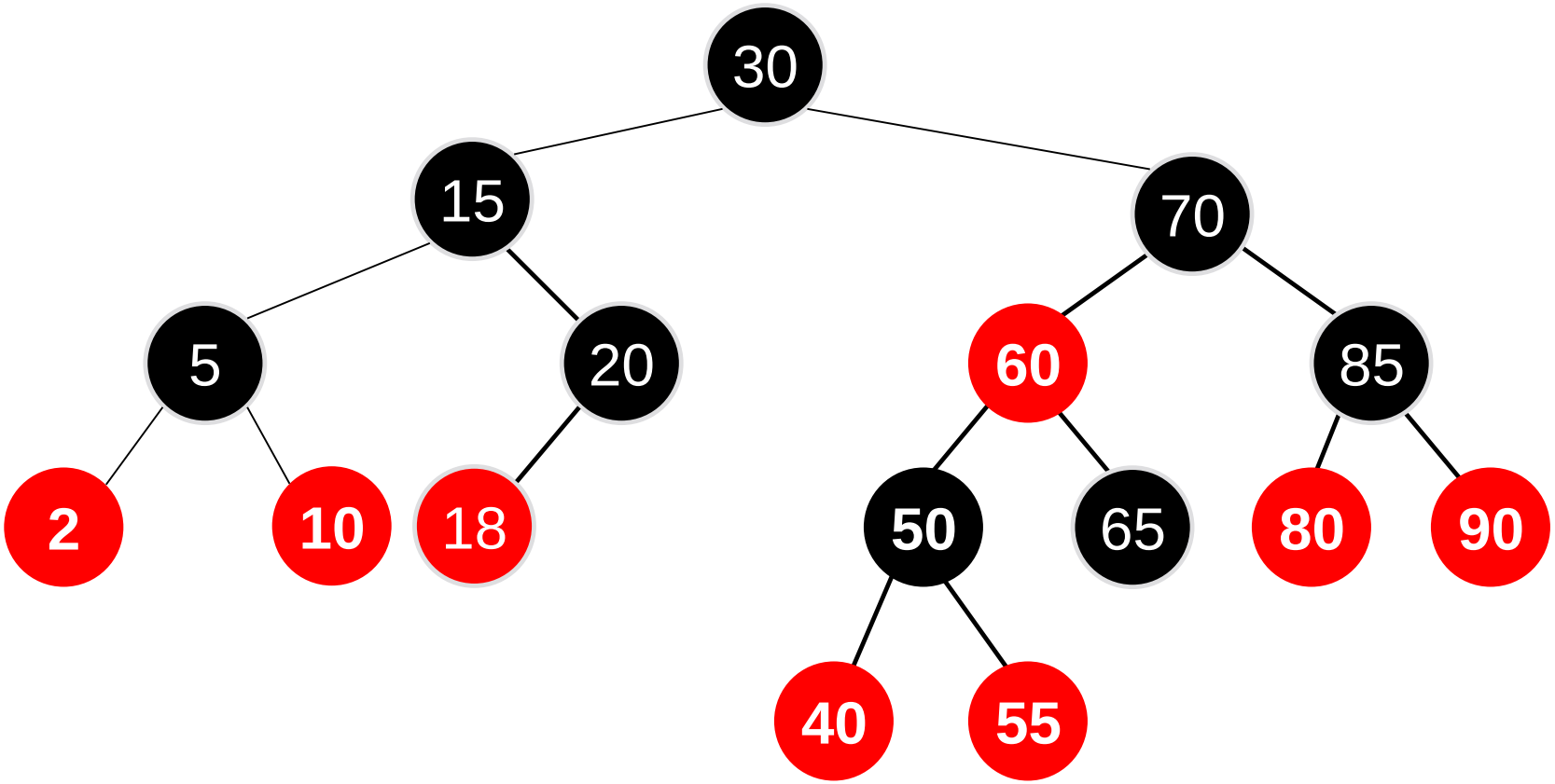
Örnek: 18 değerinin eklenmesi



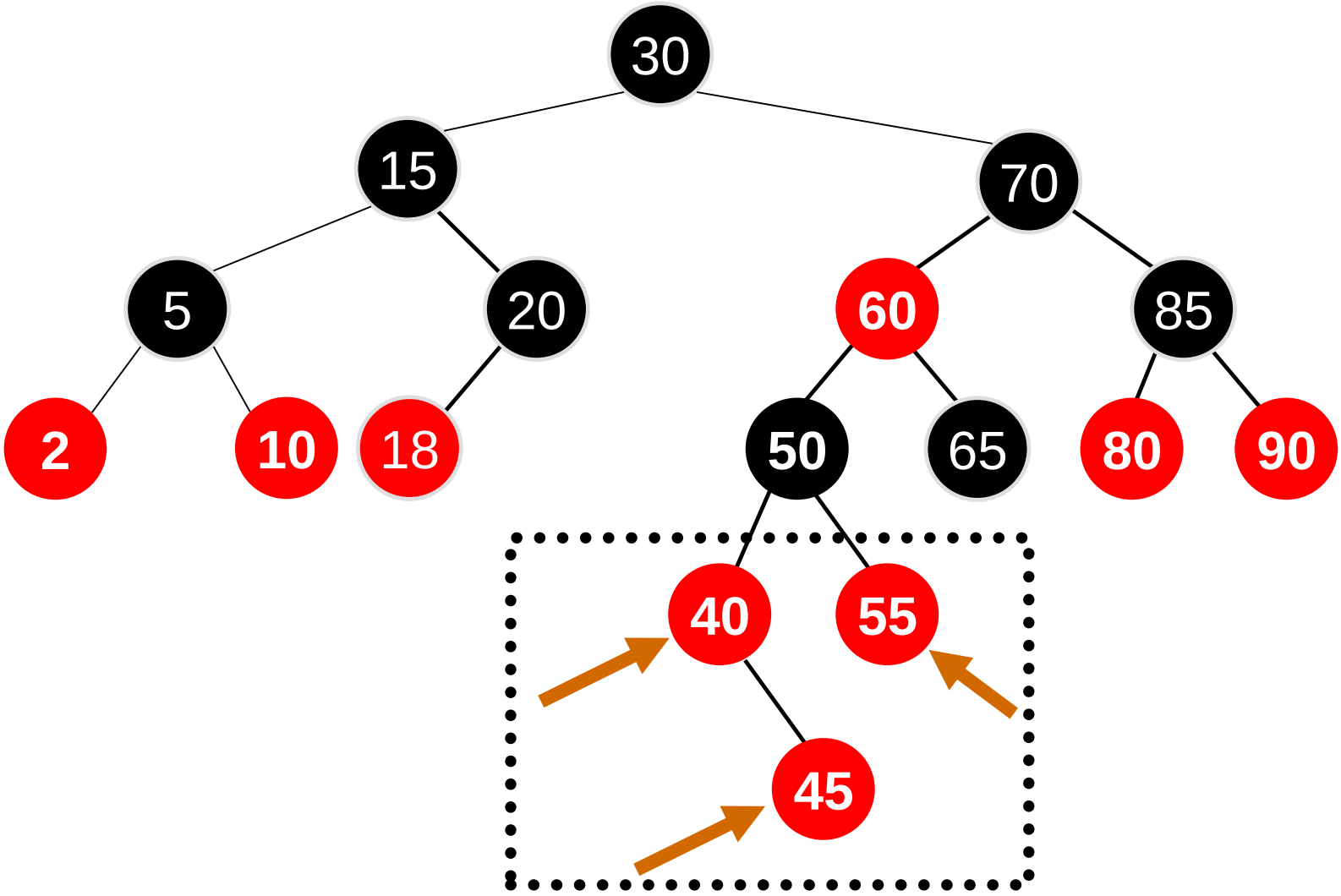
Örnek: 2 değerinin eklenmesi



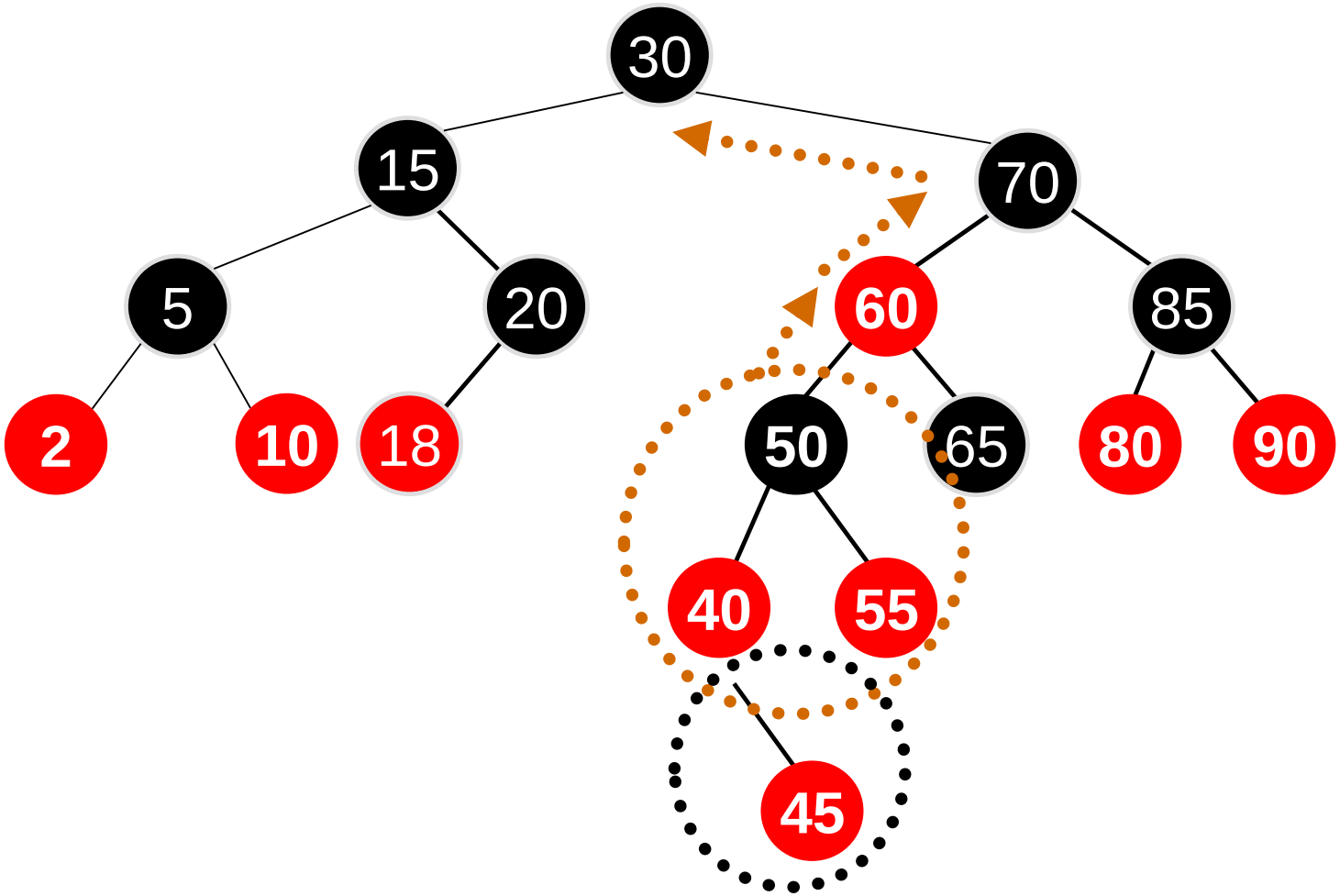
Örnek: 2 değerinin eklenmesi



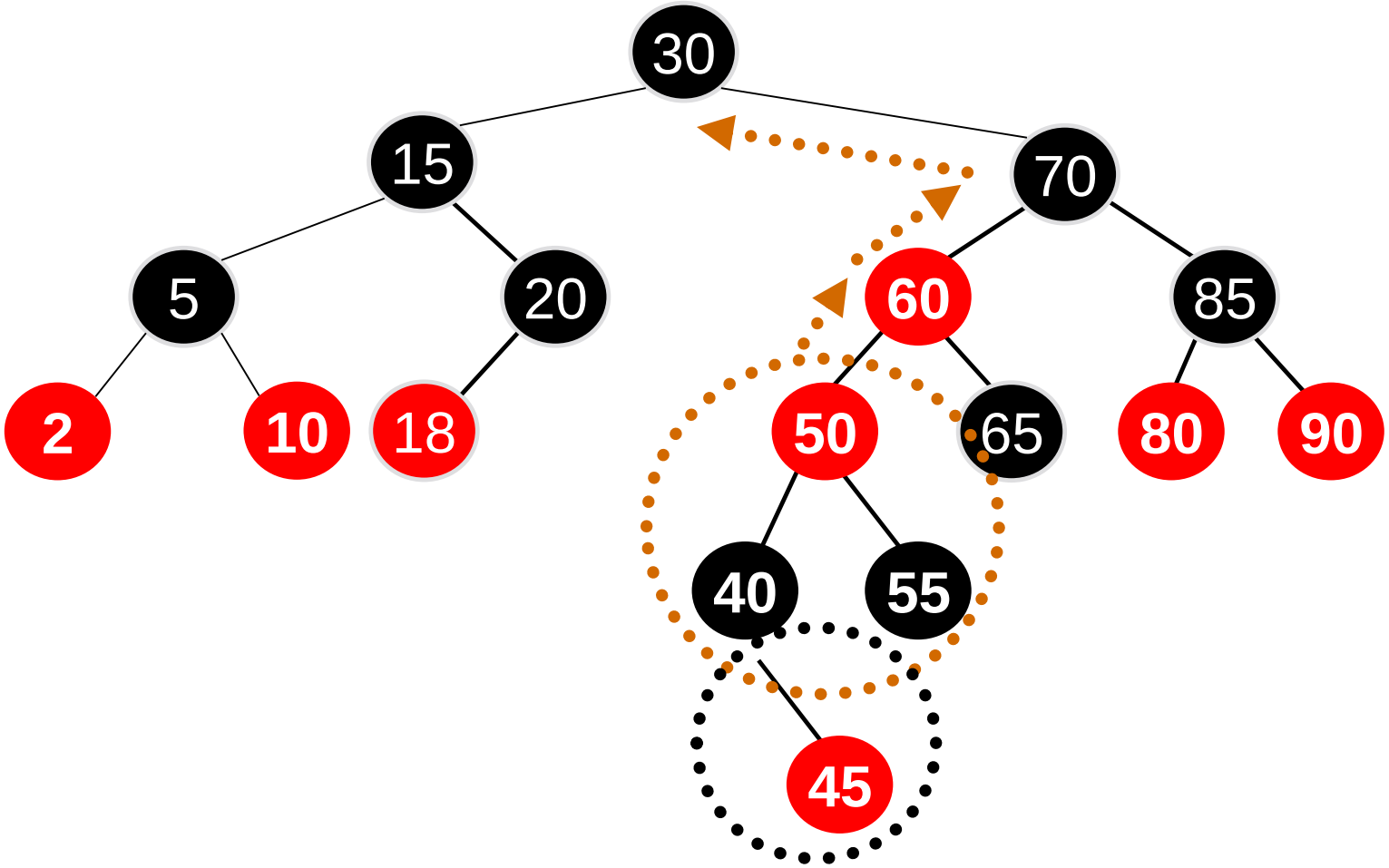
Örnek: 45 değerinin eklenmesi



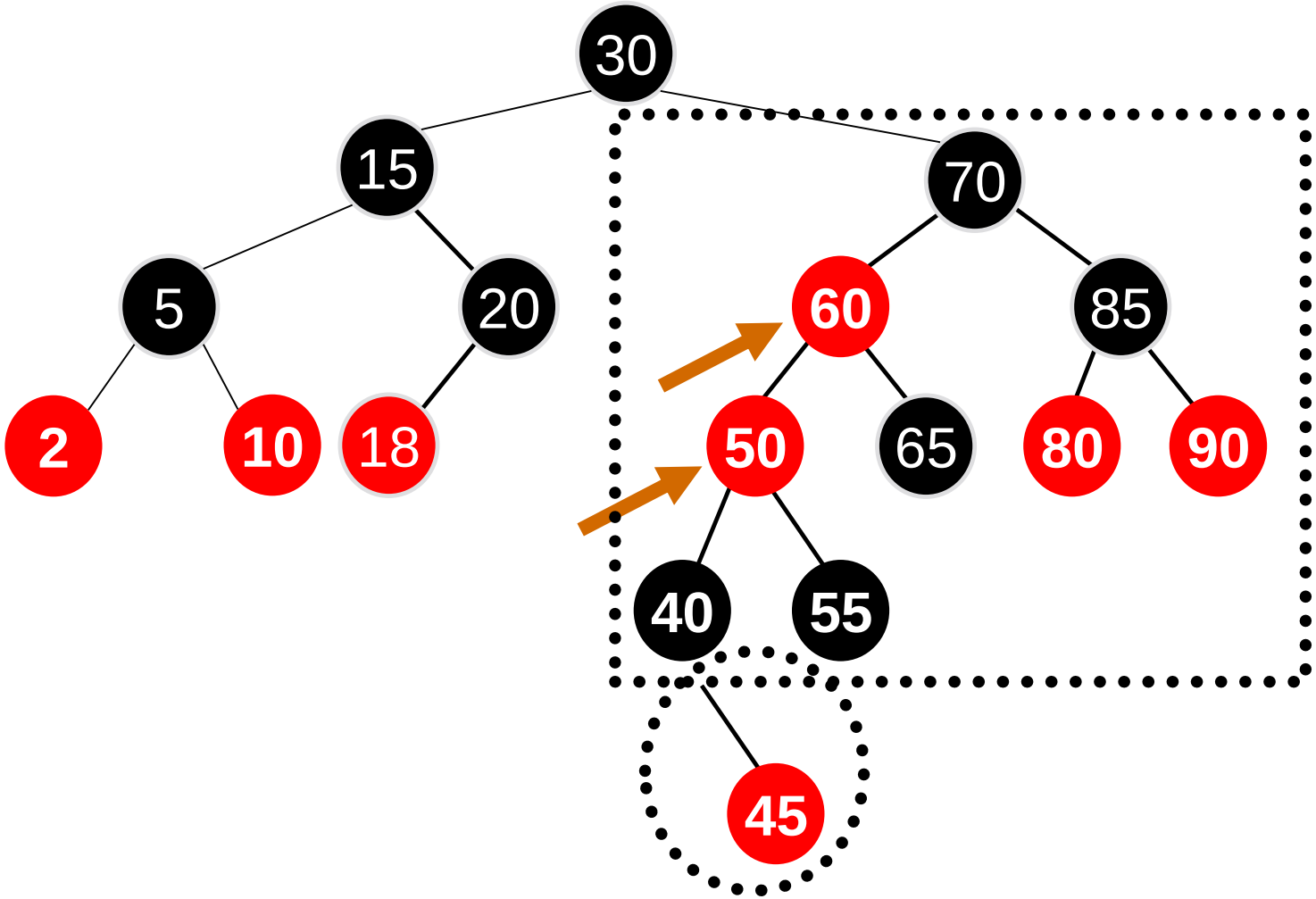
Örnek: 45’i Ekle



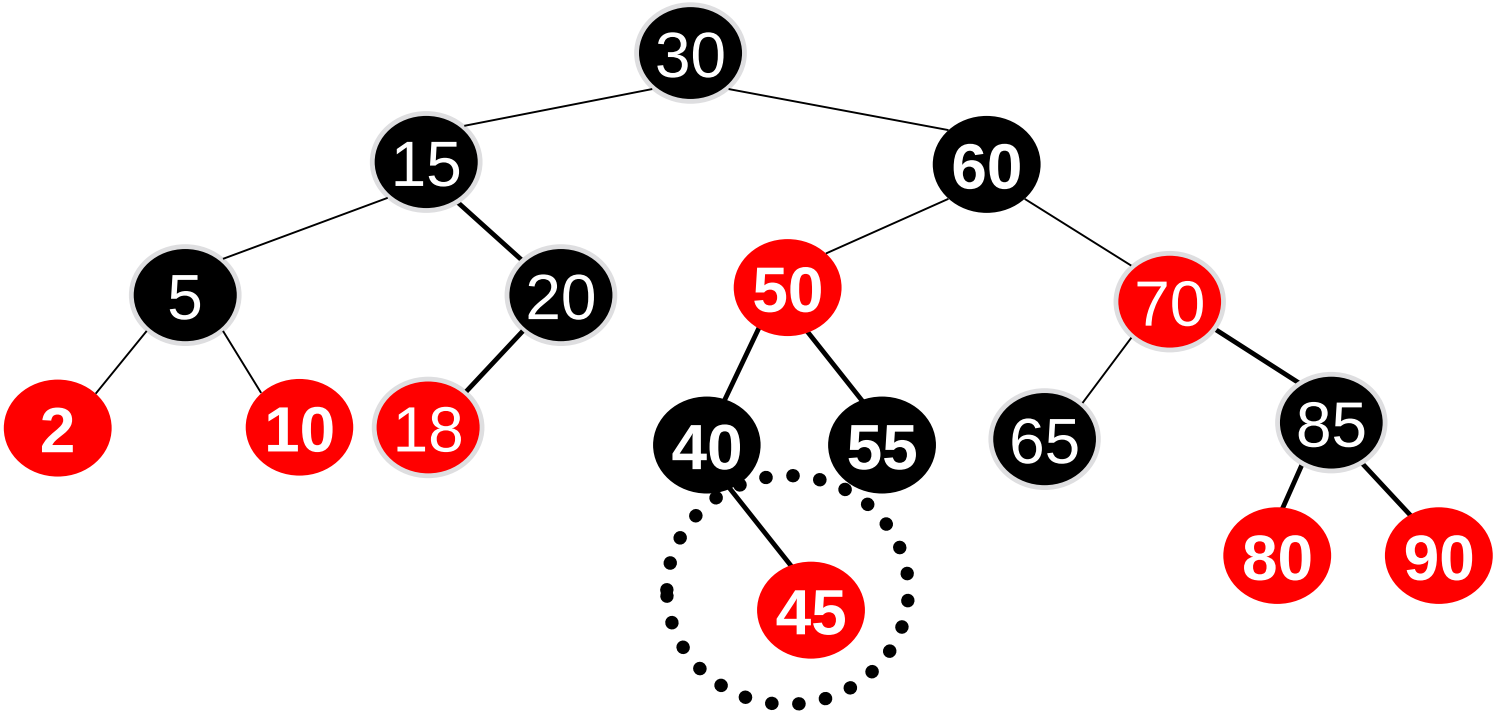
Örnek: 45’i Ekle



Örnek: 45'i Ekle (Tek Döndürme)

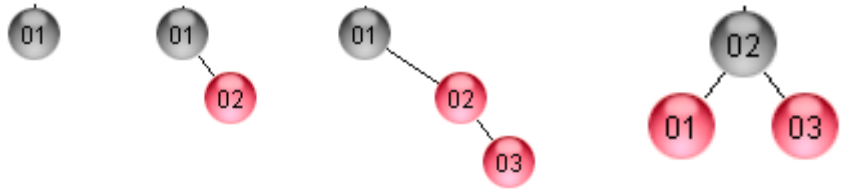


Örnek: 45’i Ekle (Tek Döndürme)



Örnek: Red-Black tree ekleme

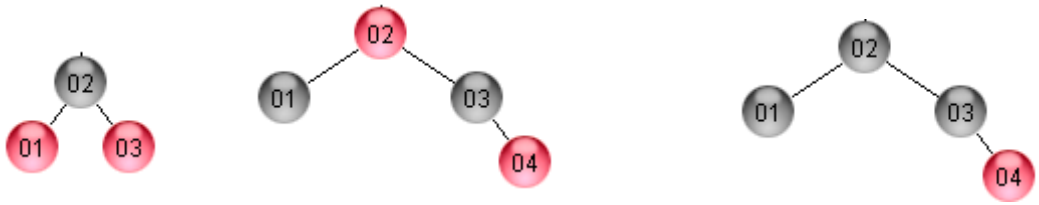
- Eklenecek değerler: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ,11, 12, 13, 14



- 3.ve 4. özellik ihlali var. 3 nolu düğümün atası siyah olmalı. 2 nolu düğüm siyah 1 nolu düğüm kırmızı olacak şekilde yeniden renklendirme yapılır.
- 4. özellik dikkate alınırsa bu defa bh değeri bütün yollarda aynı değil bu defa sola döndürme işlemi yapılacak.
  - 3 eklendi, 2 Siyah, 1 Kırmızı, Root (kök) 1 Sola döndü

Örnek: Red-Black tree ekleme

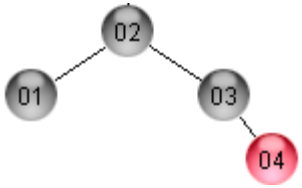
- Eklenecek değerler: 1,2,3,4,5,6,7,8,9,10,11,12,13,14
-

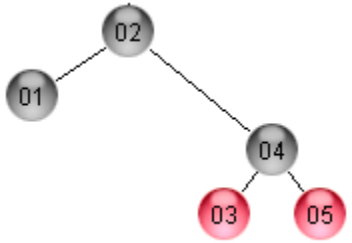


- 4 nolu düğüm eklendiğinde 3 ve 4 nolu düğümler kırmızı olduğundan 3. özellik ihlali mevcut. Yeniden renklendir:
- 3 nolu düğüm siyah atası kırmızı yapılacak.
- Ata düğüm kırmızı olduğunda çoçukları mutlaka siyah olmalı bu nedenle 1 nolu düğüm de siyah yapıldı. Ayrıca kök düğüm kırmızı olmayacağından 2 nolu düğümde siyah yapıldı. (bh değerleri aynı)
  - 4 eklendi, 2 Kırmızı, 1 Siyah, 3 Siyah, 2 Siyah

Örnek: Red-Black tree ekleme

- Eklenecek değerler: 1,2,3,4,5,6,7,8,9,10,11,12,13,14

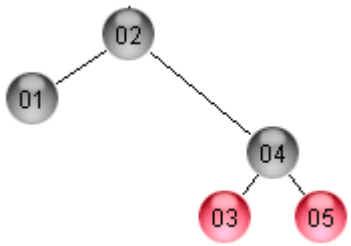


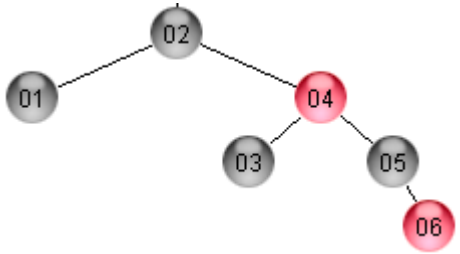


- 5 nolu düğüm eklendiğinde kırmızı olduğundan özellik 3 ihlali oluştu. 4 nolu düğümü siyah, atası olan 3 kırmızı olacak şekilde yeniden renklendirme yapıldı. Bu defa 4. özellik ihlali oluştu. Sola döndürme işlemi yapılarak ağaç dengelendi.
  - 5 eklendi, 4 B, 3R, 3 sola döndü

Örnek: Red-Black tree ekleme

- Eklenecek değerler:
  1,2,3,4,5,6,7,8,9,10,11,12,13,14

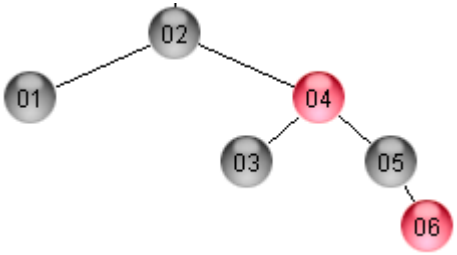


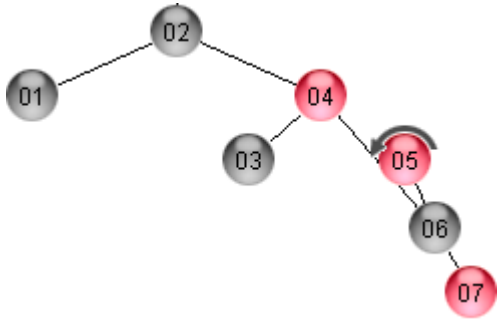


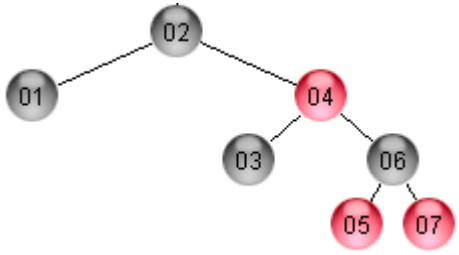
- 3. özellik ihlali yeniden renklendirme yapıldı.
  - 6 eklendi, 3B, 5B, 4R

Örnek: Red-Black tree ekleme

- Eklenecek değerler: 1,2,3,4,5,6,7,8,9,10,11,12,13,14



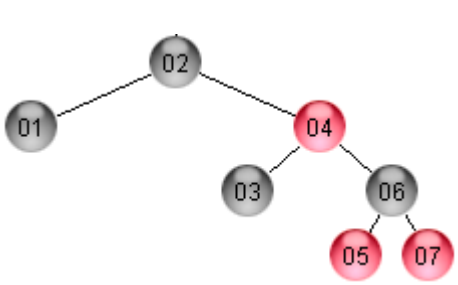


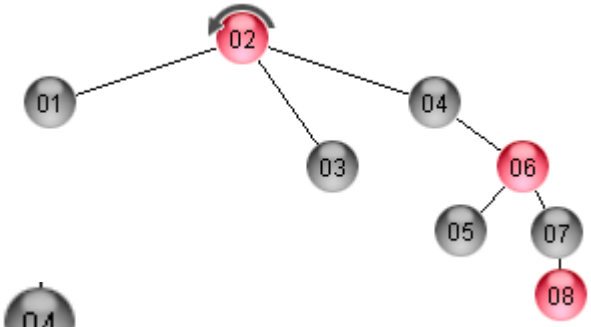


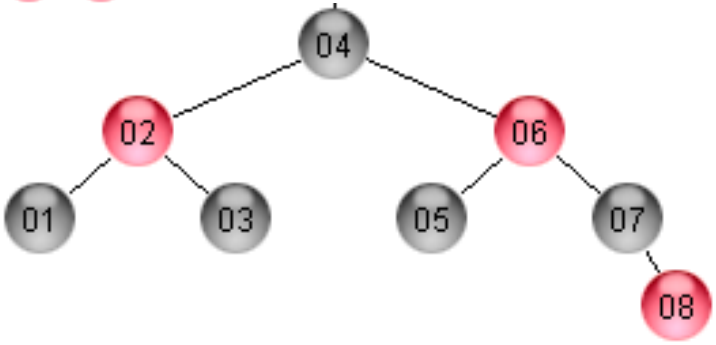
- 3. ve 4. özellik ihlali
  - 7 Eklendi, 6B, 5R, 5 sola

Örnek: Red-Black tree ekleme

- Eklenecek değerler: 1,2,3,4,5,6,7,8,9,10,11,12,13,14



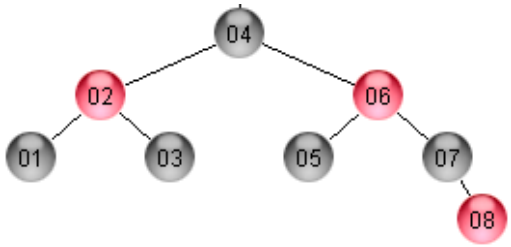


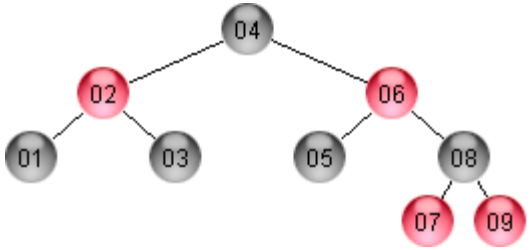


- 8 eklendi, 5B, 7B,6R, 4B,2R, 2 Sol

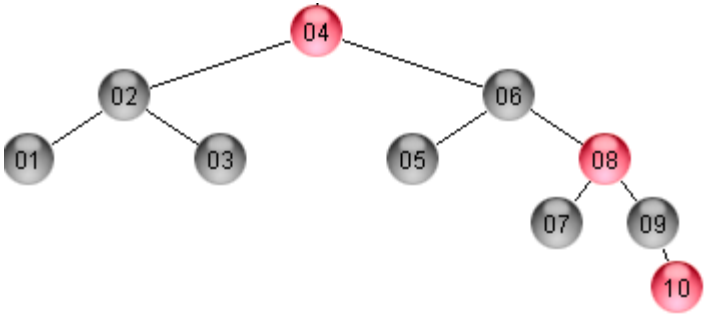
Örnek: Red-Black tree ekleme

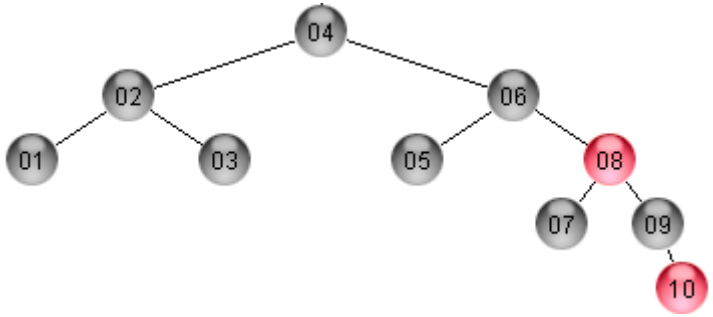
- Eklenecek değerler: 1,2,3,4,5,6,7,8,9,10,11,12,13,14





  - 9 Eklendi,8B,7R, 7 Sol

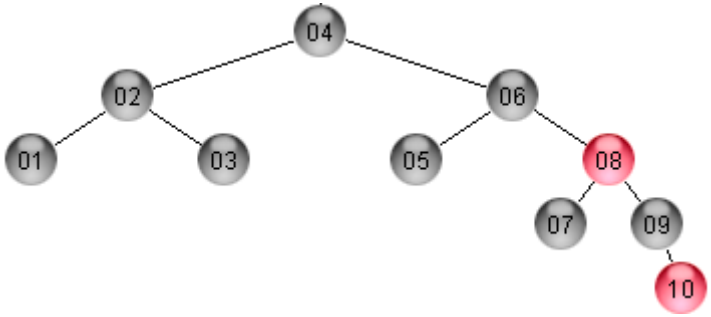


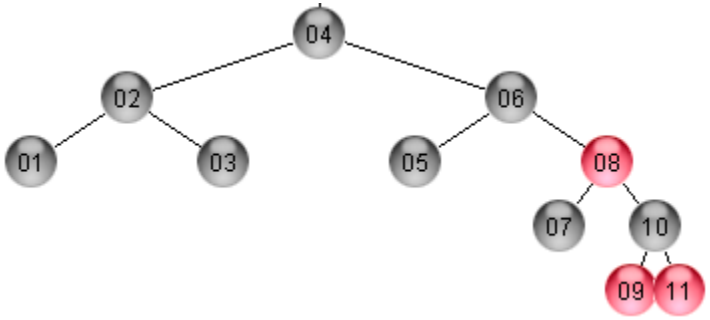


  - 10 Eklendi, 7B,9B,8R,2B,6B,4R,4B

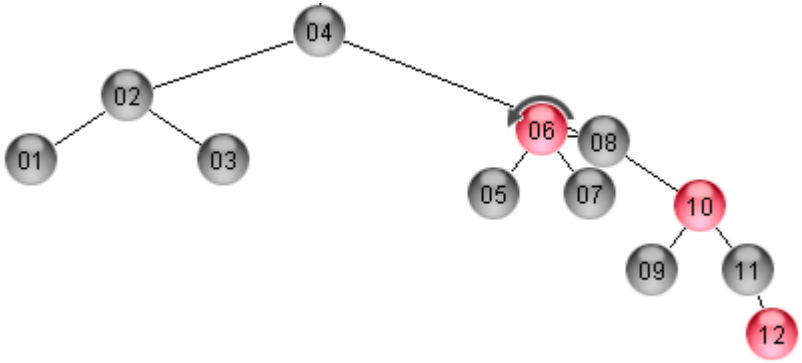
Örnek: Red-Black tree ekleme

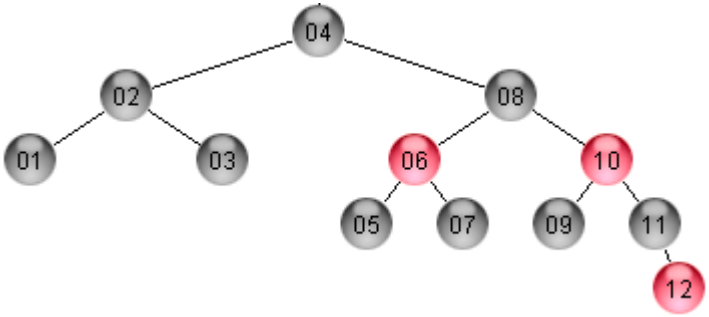
- Eklenecek değerler: 1,2,3,4,5,6,7,8,9,10,11,12,13,14





- 11 Eklendi, 10B,9R, 9 Sol

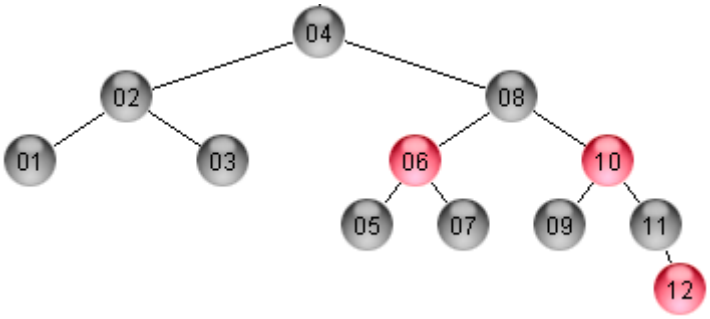


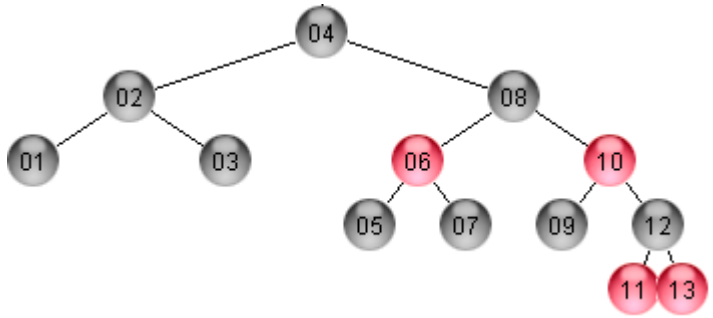


- 12 Eklendi,9B,11B,10R, 8B,6R, 6 Sol

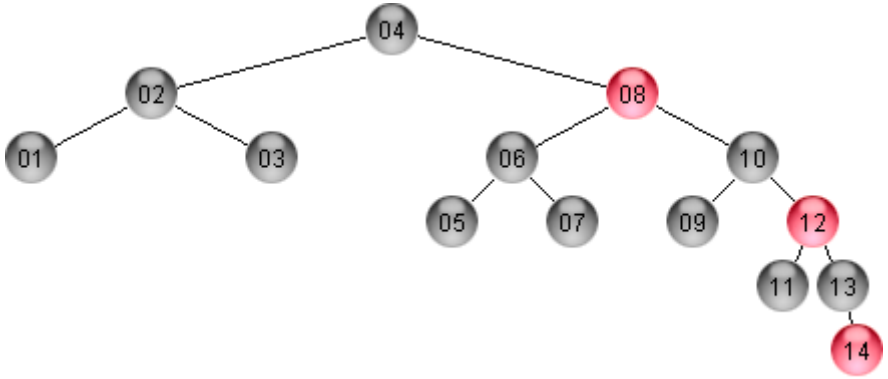
Örnek: Red-Black tree ekleme

- Eklenecek değerler: 1,2,3,4,5,6,7,8,9,10,11,12,13,14





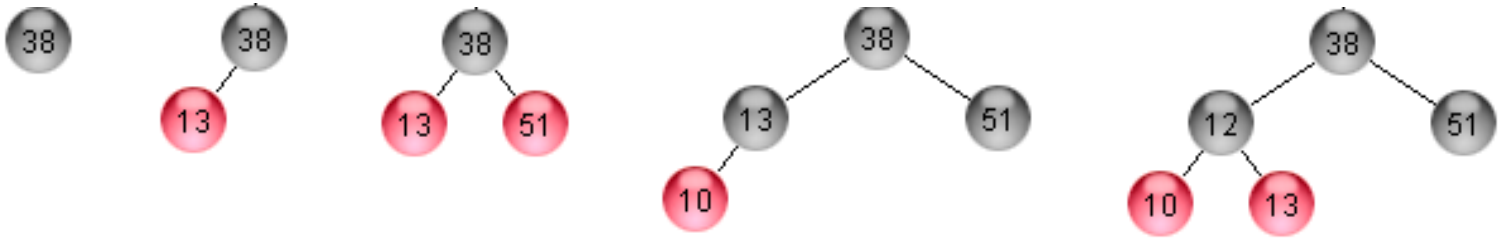
- 13 Eklendi, 12B,11R, 11 Sol



- 14 Eklendi,11B,13B,12R, 8R,6B, 10B

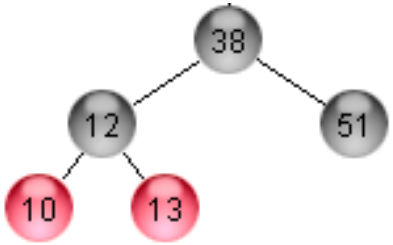
Örnek: Red-Black tree ekleme

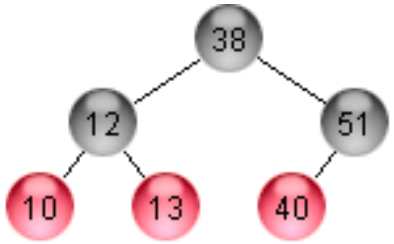
- Aşağıdaki değerleri her eklemede Red-Black tree özelliğini uygulayınız.
- 38,13,51,10,12,40,84,25

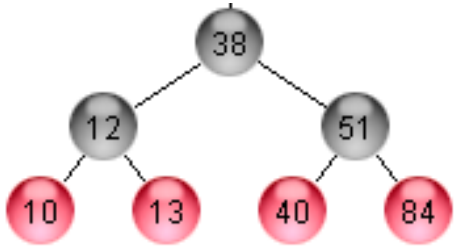


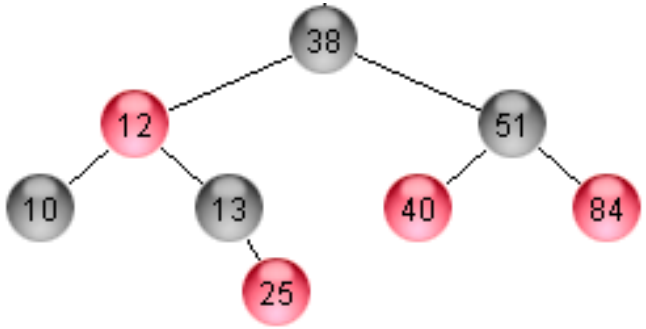
Örnek: Red-Black tree ekleme

- 38,13,51,10,12,40,84,25



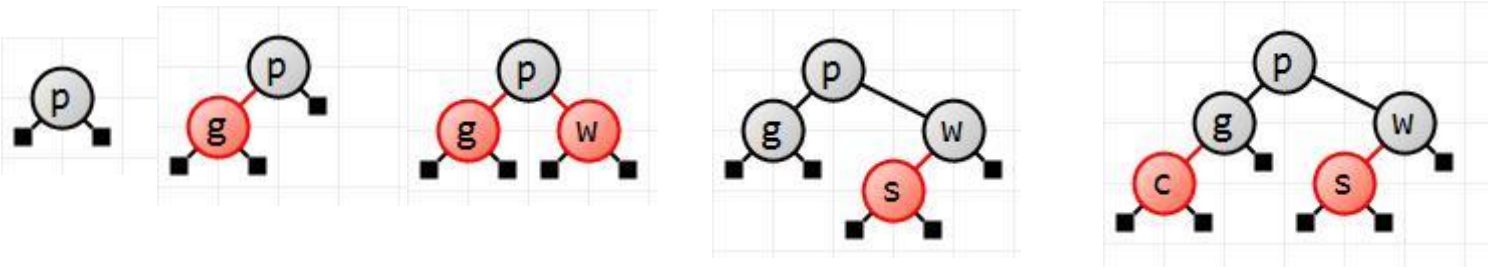


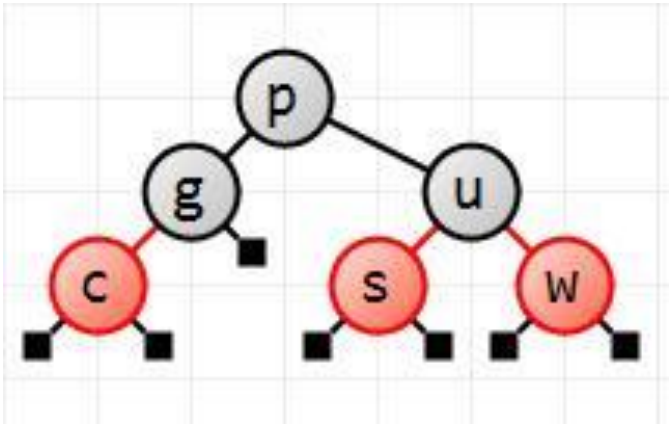




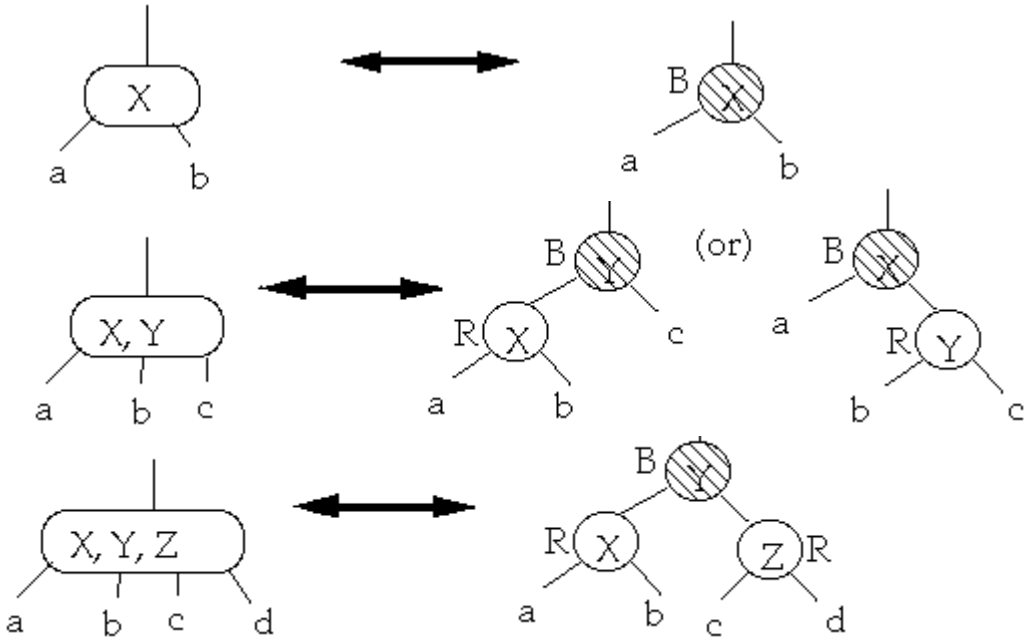
Örnek: Red-Black tree ekleme

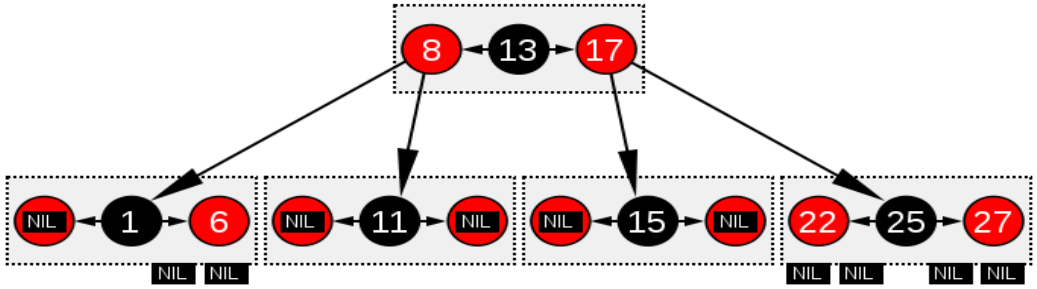
- p,g,w,s,c,u





2-3-4-tree ve Red-Black tree gösterimi

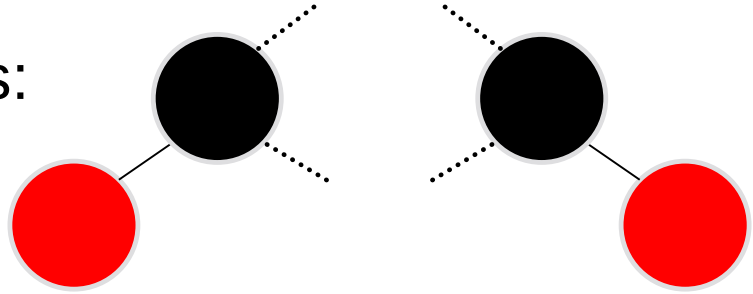




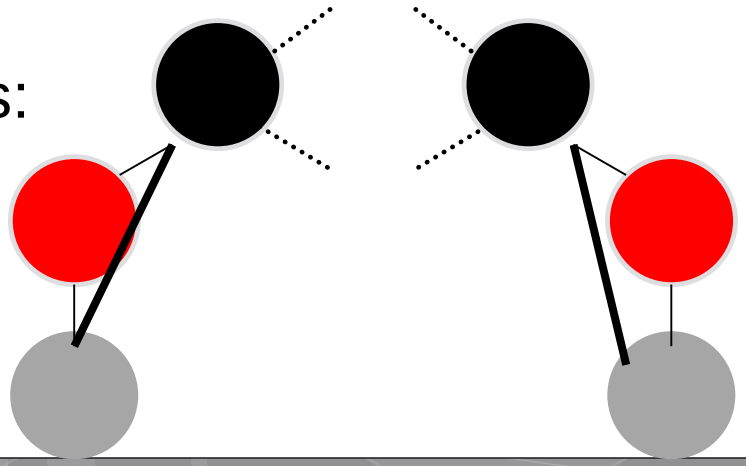
Red-Black Tree: Silme

- AVL ağaçlarında olduğu gibi çocukları olan düğüm silindiğinde soldaki en büyük düğüm veya sağdaki en küçük düğüm alınır
- Eğer silinen düğüm kırmızı ise, problem yok.

Leaf nodes:



Single child nodes:

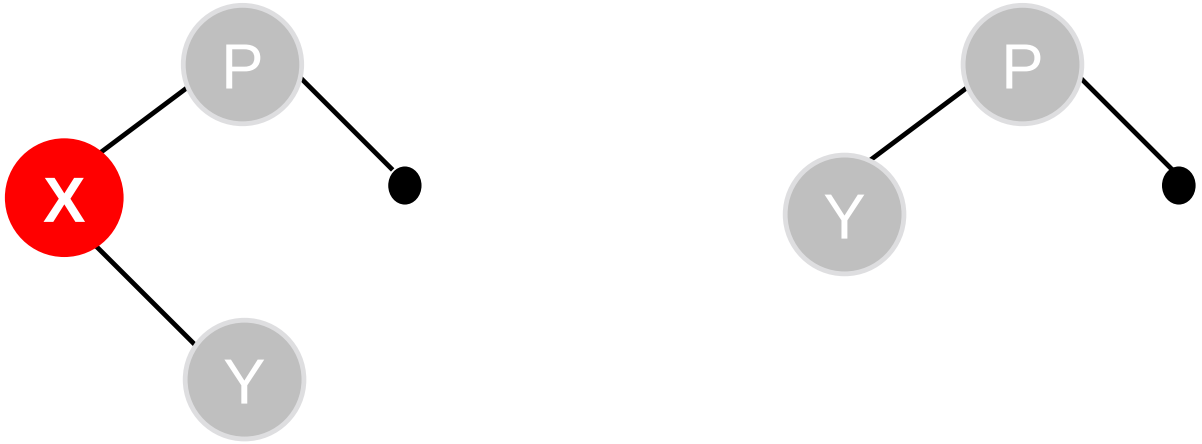


Red-Black Tree: Silme

- Eğer silinen düğüm kırmızı ise, problem yok.
- Eğer siyah ise kırmızı yap ve sil

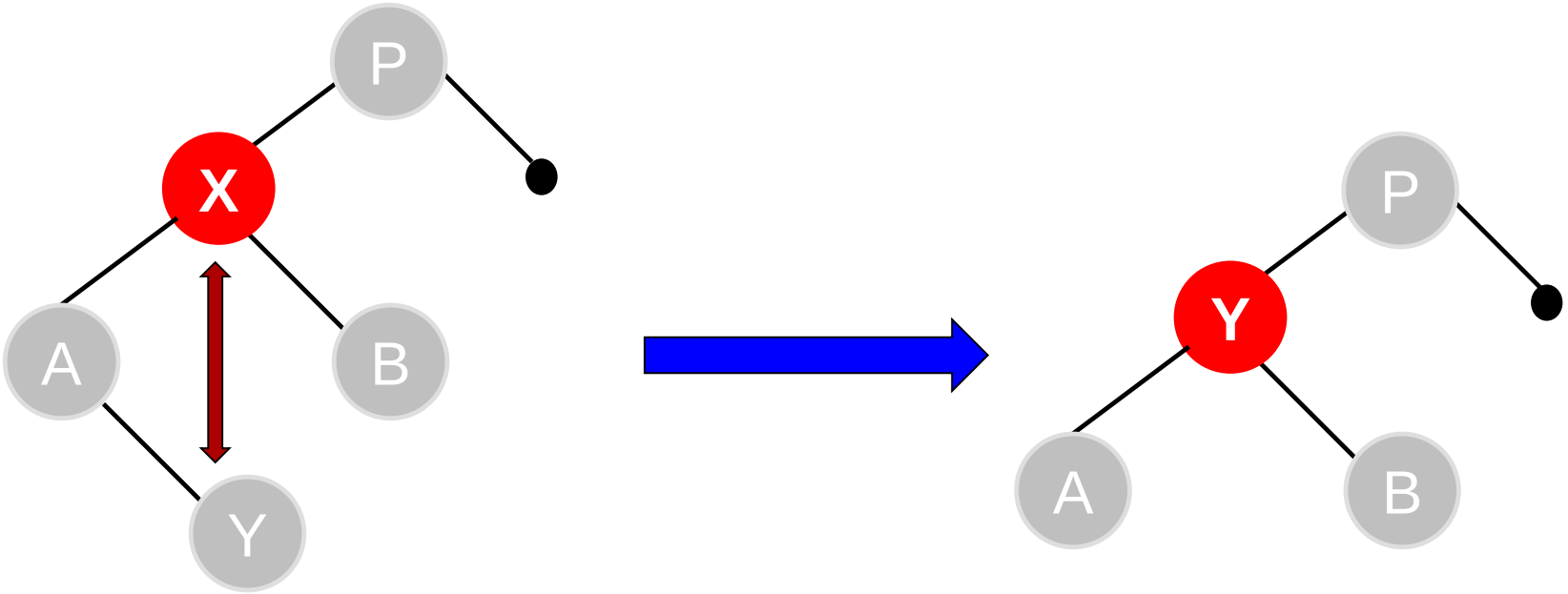
Red-Black Tree: Deletion

- Silinen düğümün tek çocuğu var ise yer değiştir ve kırmızı ise sil.



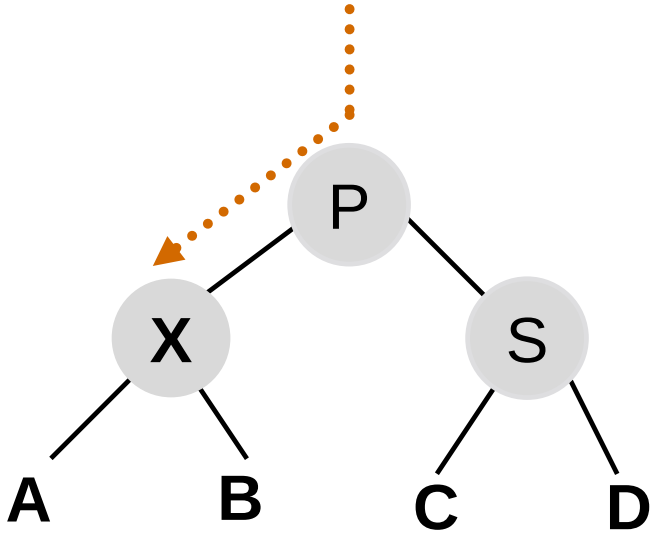
Red-Black Tree: Deletion

- Silinen düğümün iki çocuğu var ise soldaki en büyük düğüm ile yer değiştir ve kırmızı ise sil.



Top-Down Deletion

- Silinecek düğüm siyah ise → 4. özellik ihlali
- Silinecek düğümün daima kırmızı olmasını sağla.
- Kökten başlayarak, yukarıdan-aşağı doğru seyahat ederek silinecek düğüm için bak.

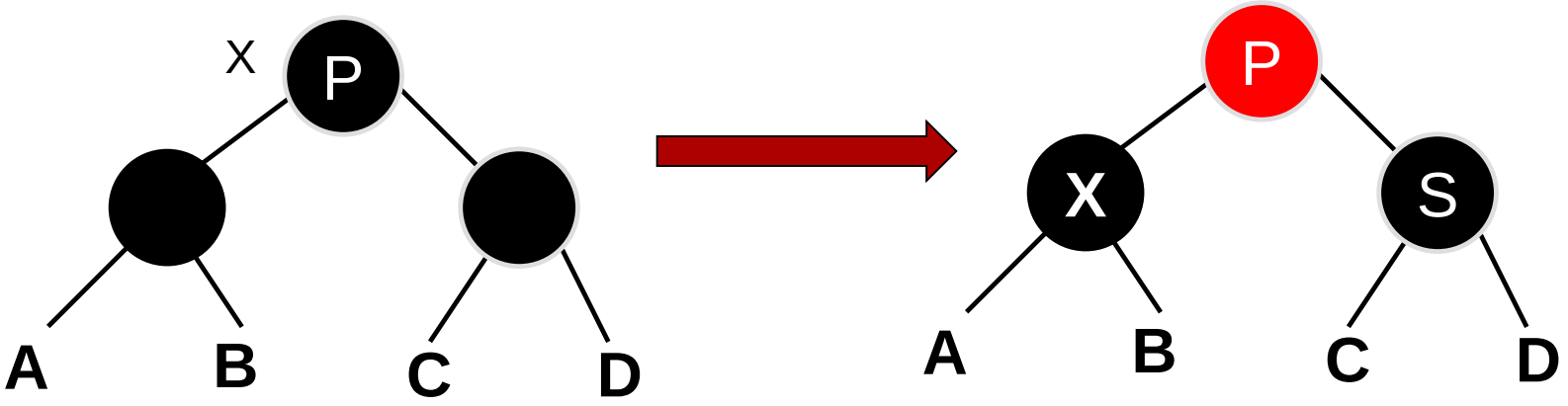


X: ziyaret edilen düğüm
P: ebeveyn
S: kardeş

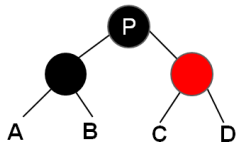
Fikir: X'i kırmızı yaptığından emin ol!

Muhtemel Durumlar- Adım 1

- **Adım 1-A1:** Kökün her iki çocuğu da siyah ise;
  a) Kökü kırmızı yap
  b) X'i kökün uygun çocuğuna taşı
  c) Adım 2'ye geç

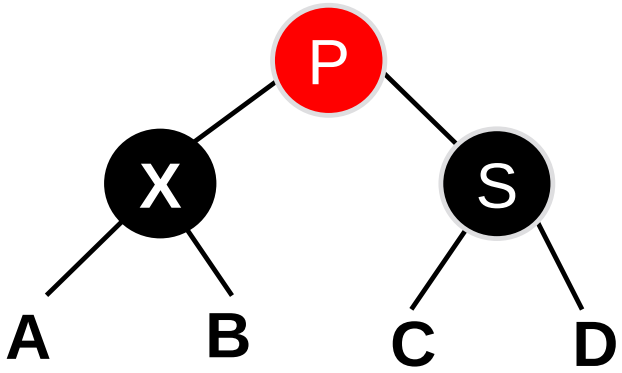


- **Adım 1-A2:** Kökün her iki çocuğu da siyah değil ise, **X**'i kök (X=P) olarak tasarla ve **Adım 2-B**'yi işlet.



Muhtemel Durumlar-Adım 2

- Silinecek düğüme ulaşıncaya kadar devam et; **P** kırmızı, **X** ve **S** siyah.



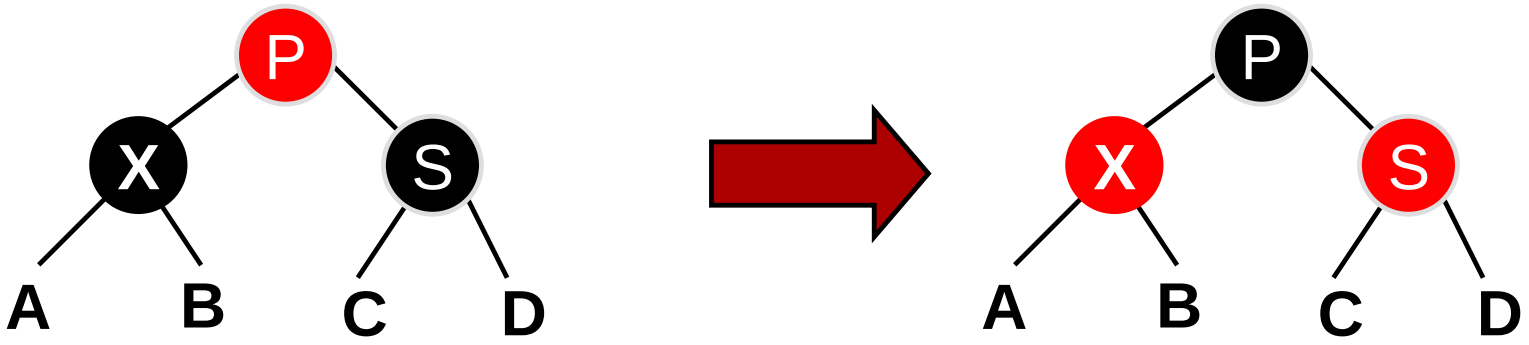
- X'i kırmızı yaptığımız zaman, X ve S'nin çocuklarına bağlı olarak muhtemelen döndürme işlemi yapılacak. Burada 2 durum mevcut.
  - **Adım 2-A:** X'in çocukları (A & B) siyahtır (black)
  - **Adım 2-B:** X'in çocuklarından en az biri kırmızı (red) (A, B, veya ikisi)

Adım 2-A

- **X**’ in çocukları (**A** ve **B**) siyah ise **S**’ nin çocuklarına bağlıdır.
  - **Adım 2-A1: S**’ in her iki çocuğu da siyah, **P, X, S** yeniden renklendir.
  - **Adım 2-A2: S**’ in sol çocuğu kırmızı, çift döndürme
  - **Adım 2-A3: S**’ in sağ çocuğu kırmızı (Her iki çocuğu da kırmızı ise sağ çocuğa göre işlem yap), tek döndürme

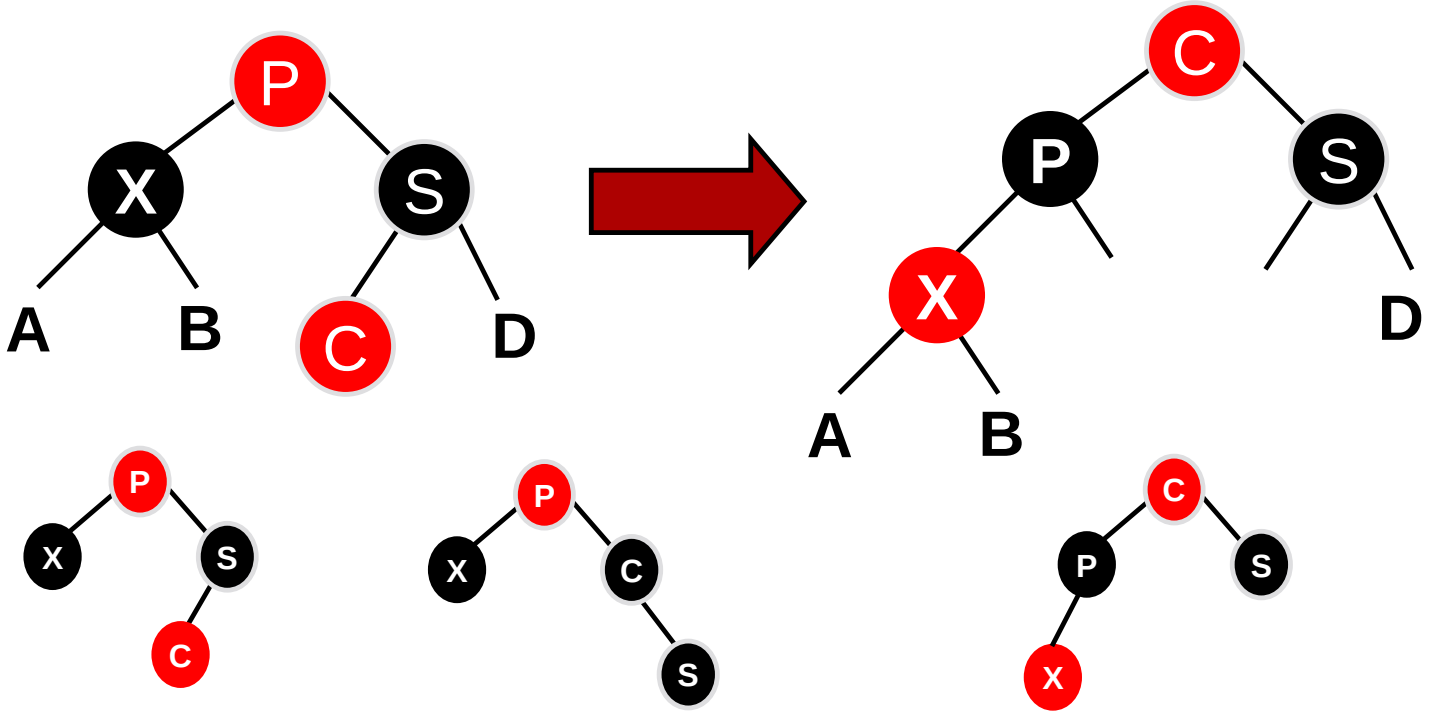
Adım 2-A

- **Adım 2-A1:** S'in her iki çocuğu da siyah ise, yeniden renklendir. (**Adım 1-A1** uygulanır**)**



Adım 2-A

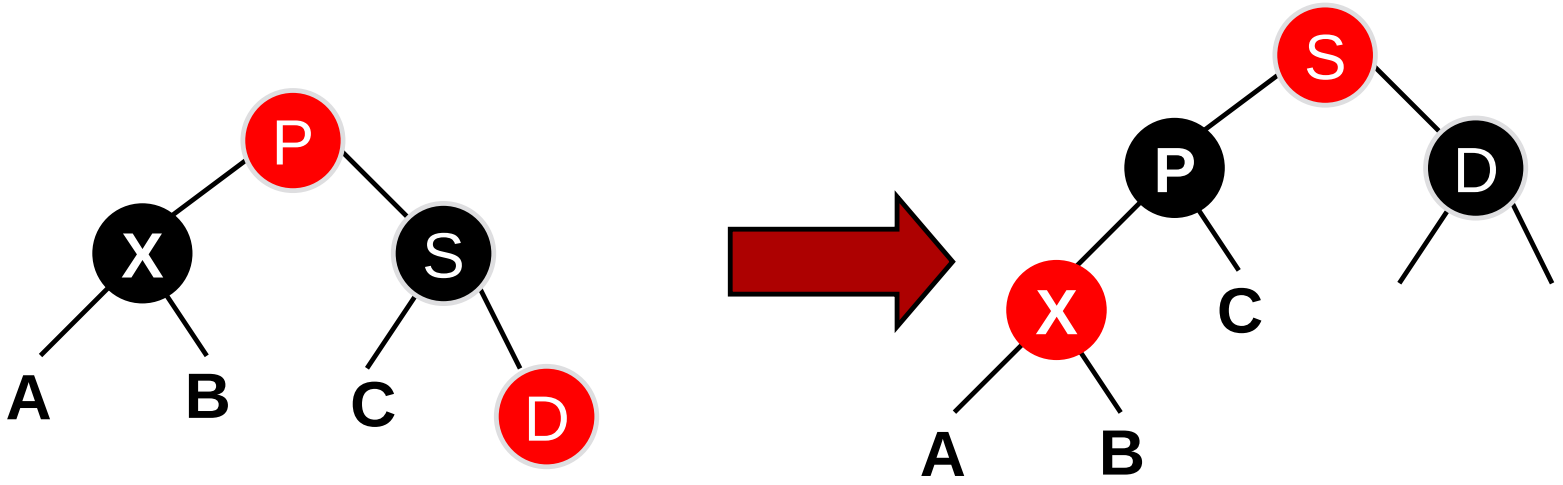
- **Adım 2-A2:** S'in sol çocuğu kırmızı ise, çift döndürme yap ve renklendir(P siyah ise sadece X kırmızı olur)



Her düğüm yerleştiği düğümün rengini alıyor, son adımda X direk kırmızı yapılıyor

Adım 2-A

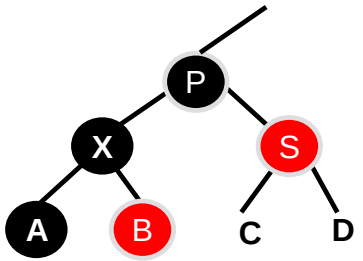
- **Adım 2-A3:** S'in sağ çocuğu kırmızı (Her iki çocuğu da kırmızı ise sağ çocuğa göre işlem yap) ise, tek döndürme yap ve renklendir



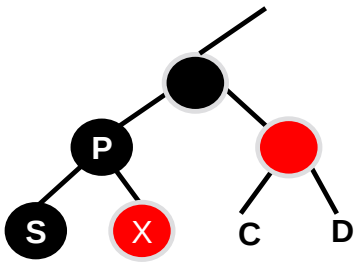
Her düğüm yerleştiği düğümün rengini alıyor, son adımda X direk kırmızı yapılıyor

Adım 2-B

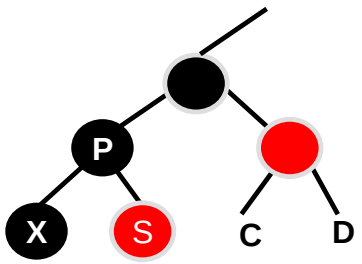
- X'in çocuklarından en az biri kırmızı ise; X'i uygun çocuğa taşı



  - **Adım 2-B1:** Eğer yeni X kırmızı ise taşımaya devam et



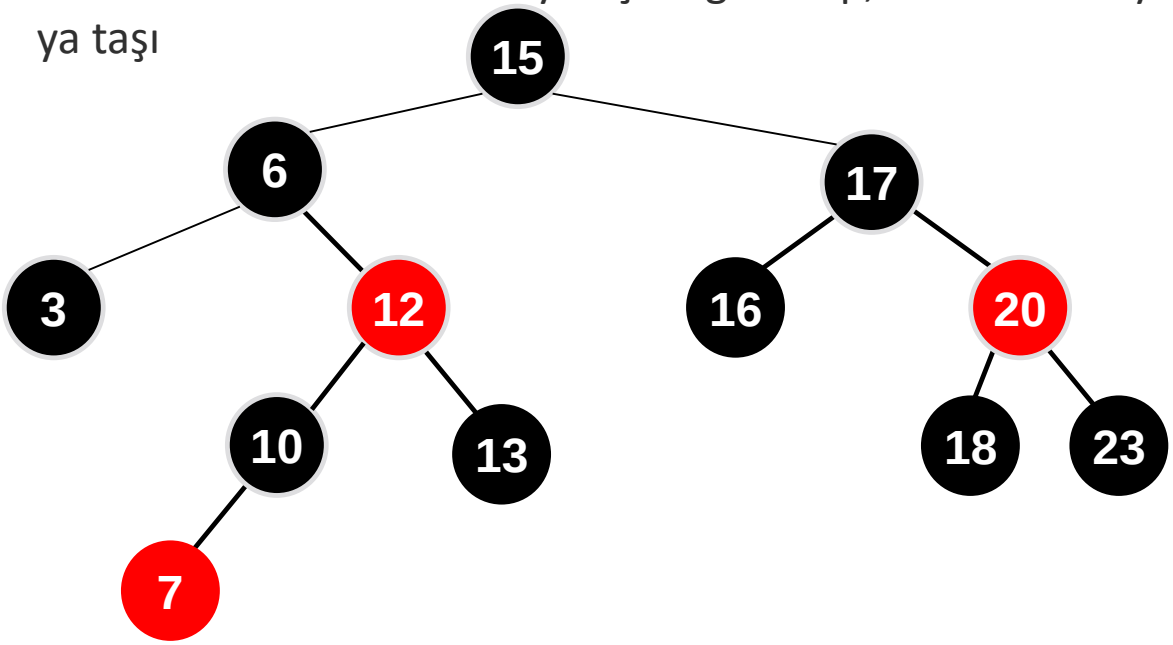
  - **Adım 2-B2:** Eğer yeni X siyah ise (P Siyah, S kırmızı), S'yi P' nin etrafında dönder P ve S'yi yeniden renklendir ve Adım 2'ye git.



Adım 3-4

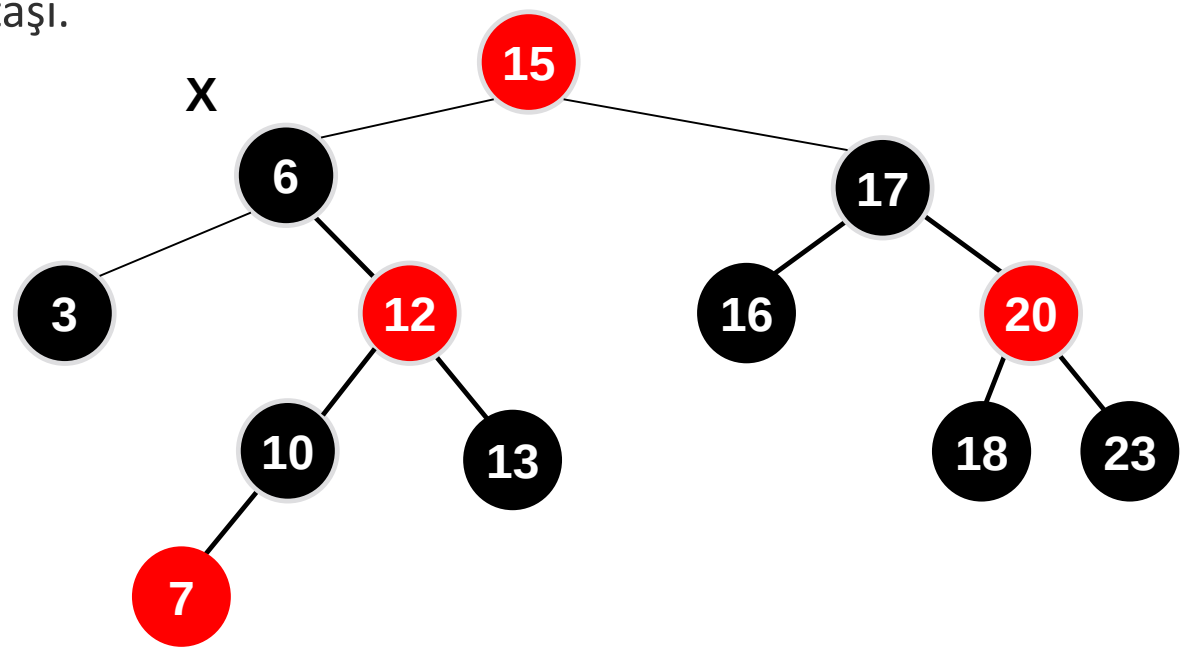
- **Adım 3-** Sonunda silinecek düğümü bul ve sil
- **Adım 4-** Kökü yeniden renklendir (Siyah yap).

- **Örnek: R-B** ağacından **10** değerini siliniz.
  - **Adım 1: A1**-Kök iki tane siyah çocuğa sahip, kökü kırmızı yap X' 6 ya taşı



Örnek

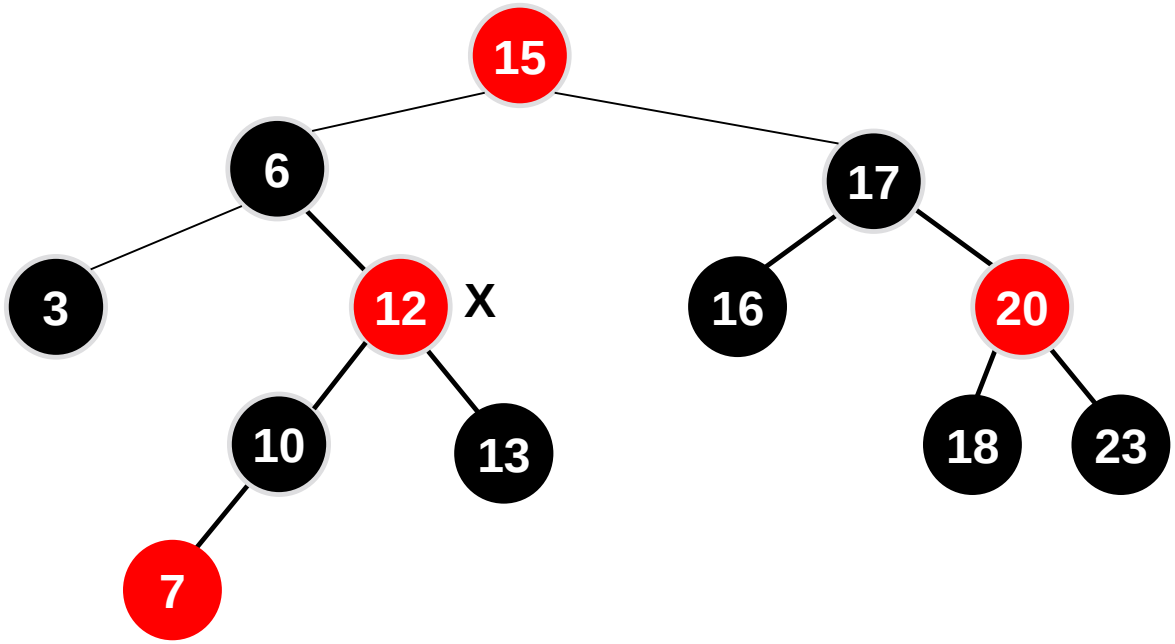
- **Adım 1:** Kök iki tane siyah çocuğa sahip, kökü kırmızı yap X' 6 ya taşı.



- X'in çocuklarında biri kırmızı (Adım 2). X'i uygun çocuğa taşı, yeni X(12) aynı zamanda kırmızı (Adım 2-B1)

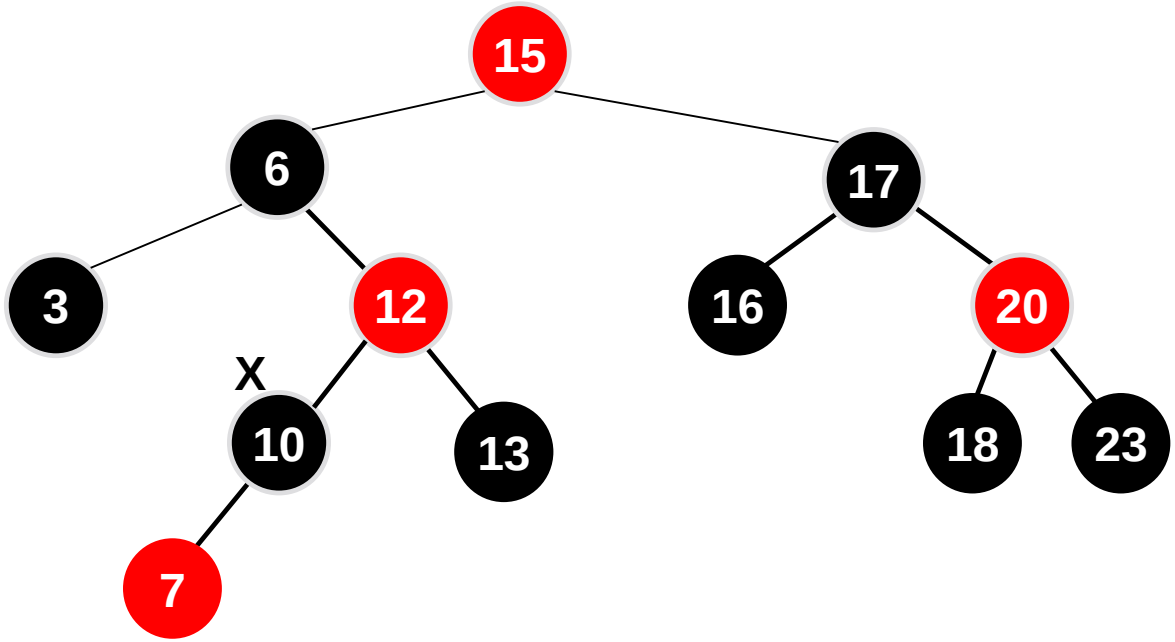
Örnek

- X taşımaya devam et X (10) .



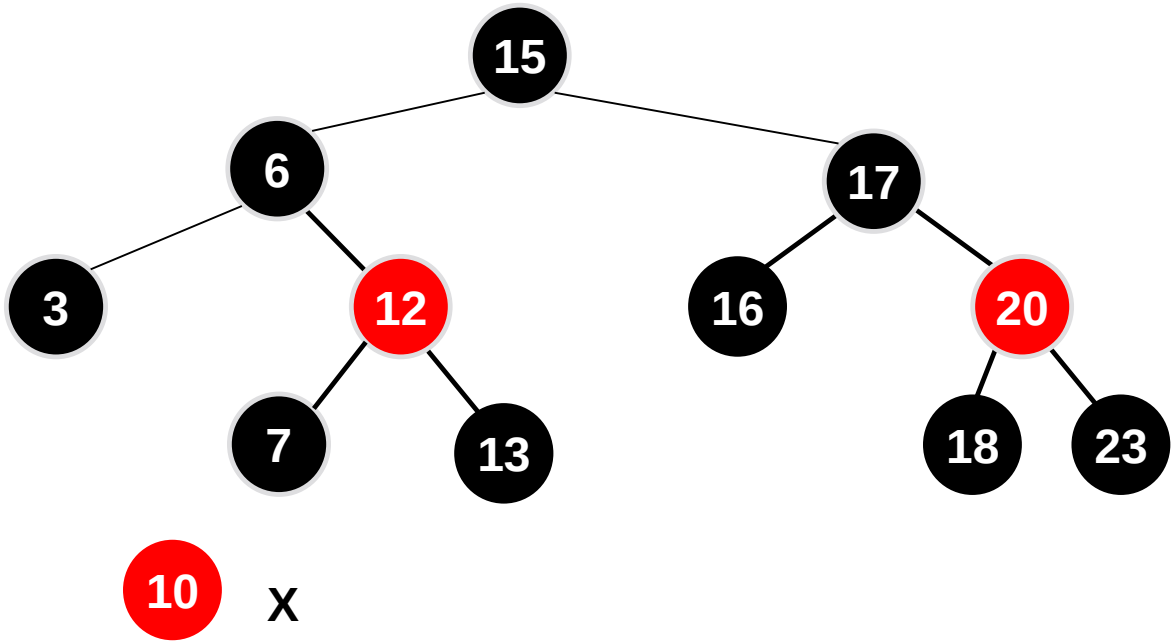
Örnek

- Silinecek düğüm bulundu. **Adım 3**, çocuğu ile yer değiştir ve sil.



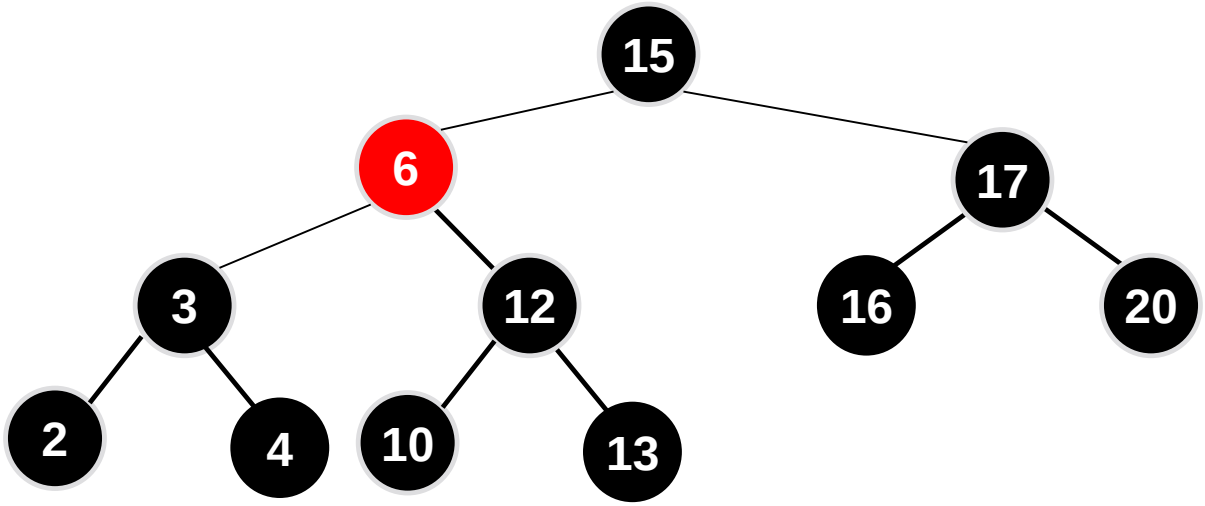
Örnek

- Adım 4: Kök değerini siyah yap



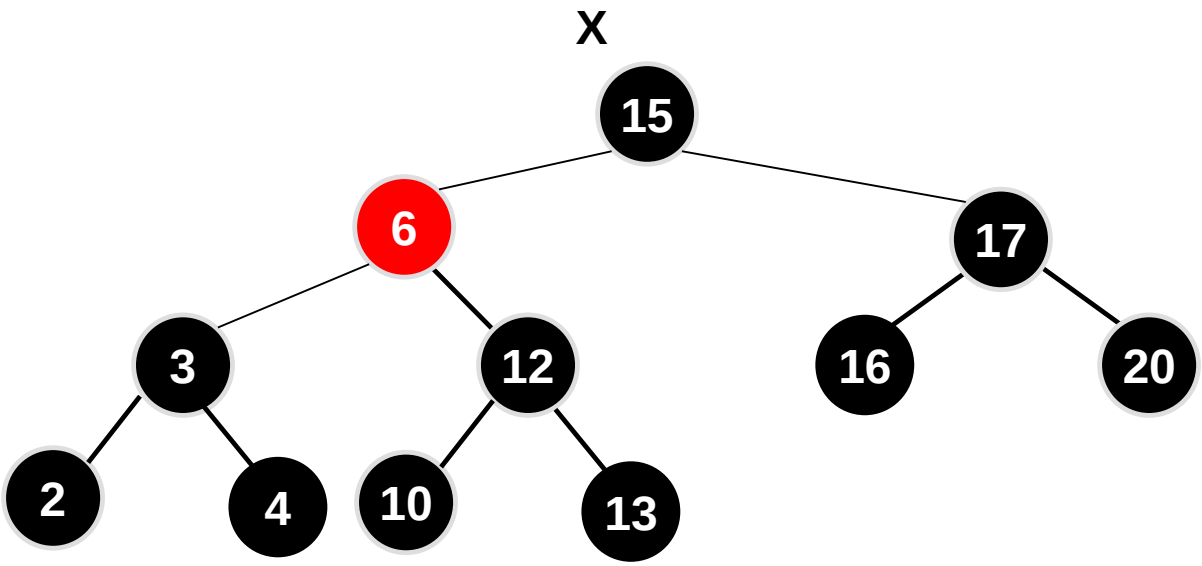
Örnek

- Örnek: R-B ağacından10 değerini siliniz.



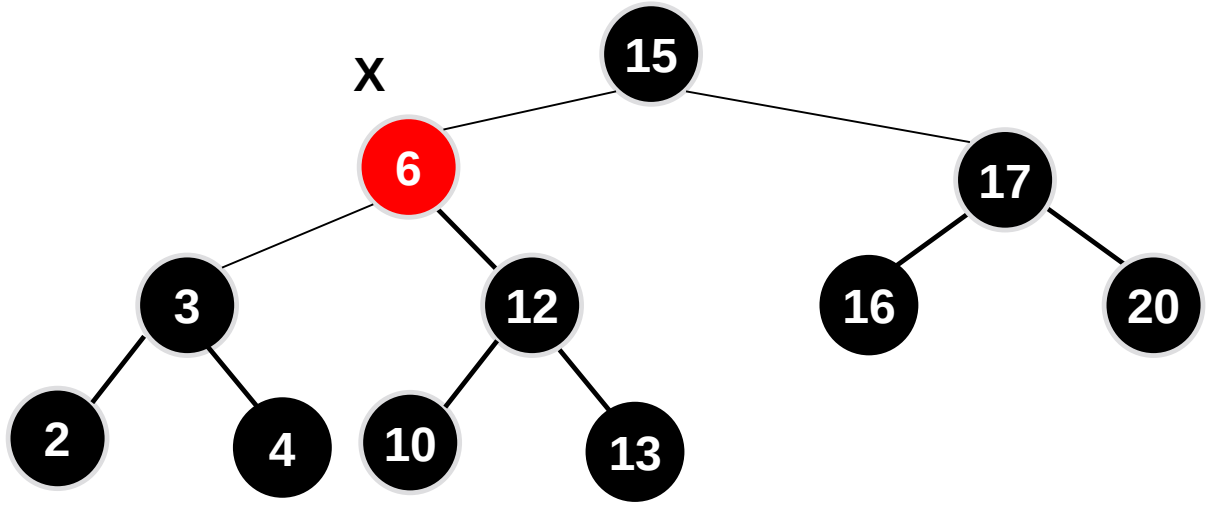
Örnek

- **Adım 1: A2-**Kök bir tane siyah çocuğa sahip, X=kök yap
- **Adım 2-B:** X en az bir tane kırmızı çocuğa sahip. X uygun çocuğa taşı.



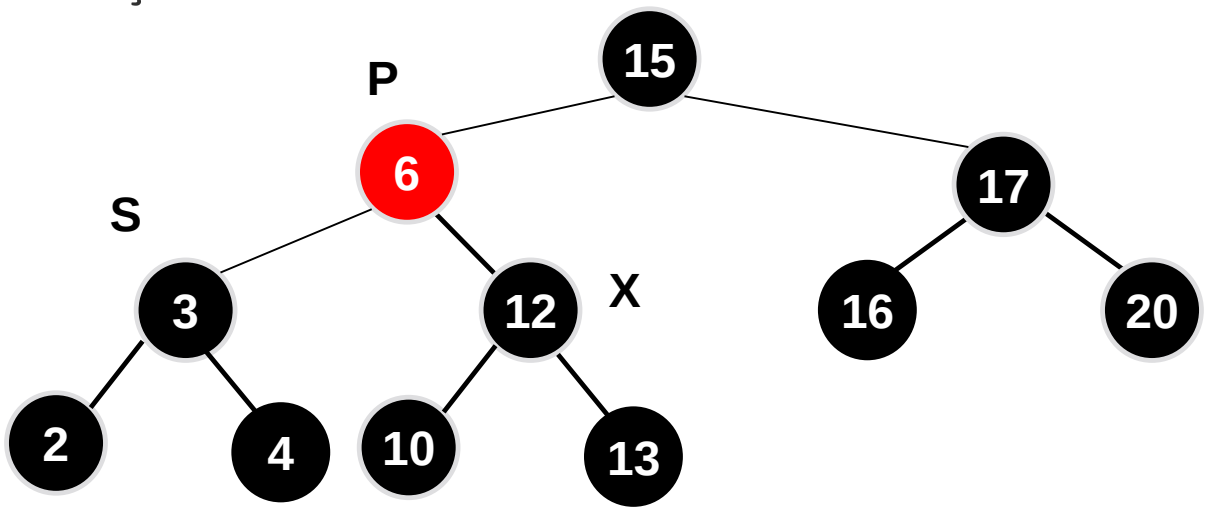
Örnek

- **Adım 2-B1:**Yeni X (6) kırmız olduğundan uygun çocuk düğüme taşımaya devam et X(12)



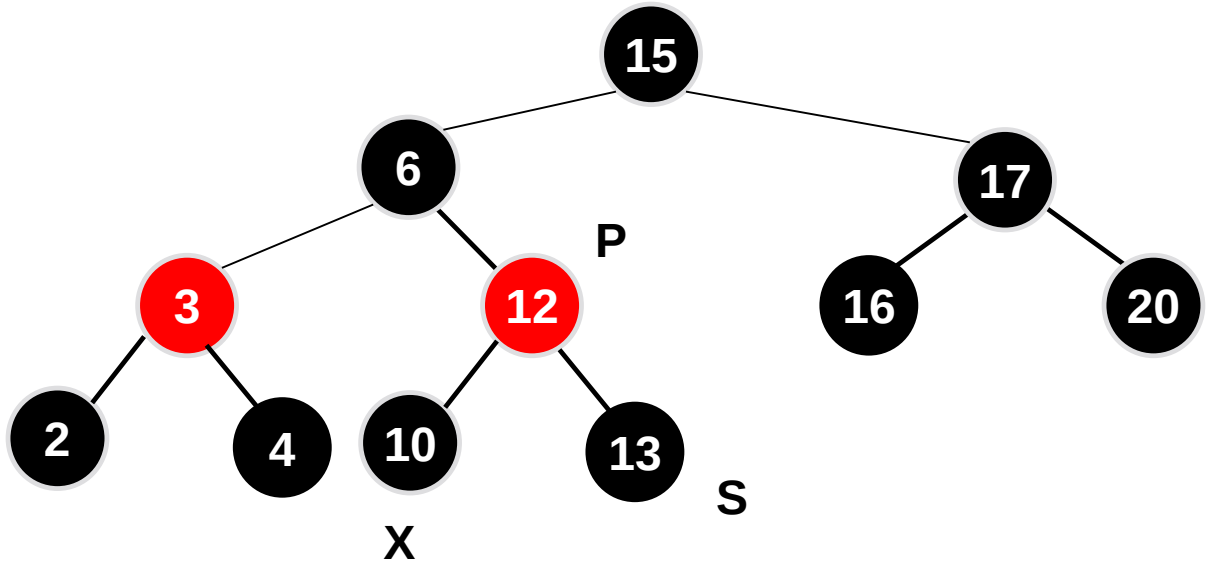
Örnek

- X iki tane siyah düğüme sahip. X' in kardeşi S de 2 tane siyah düğüme sahip (Adım 2-A1).
- **Adım 2-A1:** S'in her iki çocuğu da siyah ise, X, P, S yeniden renklendir.
- X'i 10'na taşı.



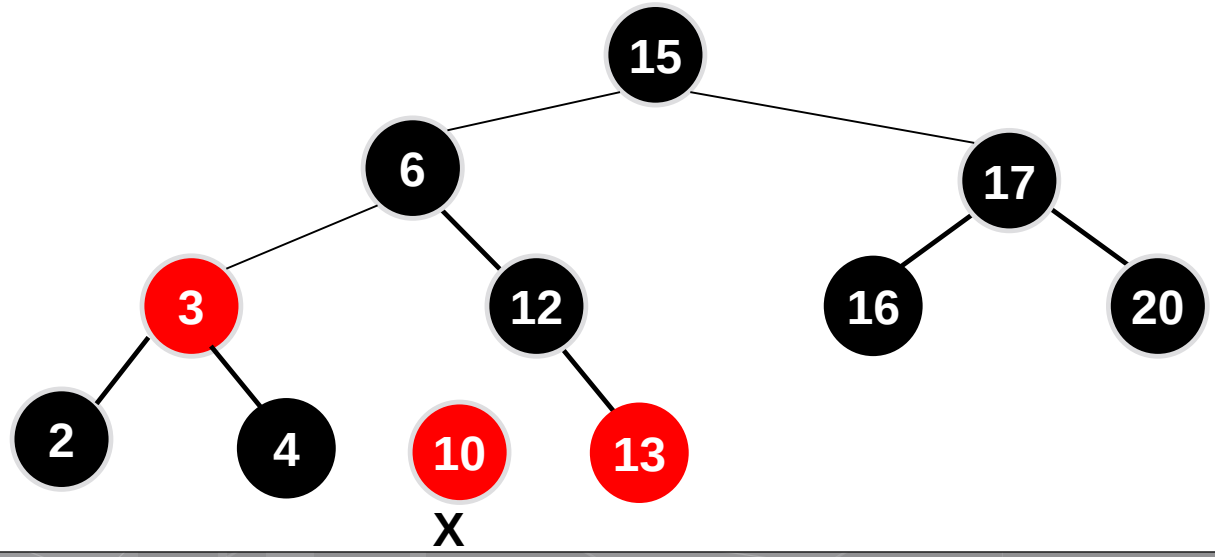
Örnek

- X, hedef olan yaprak düğüm olduğundan silinebilir fakat rengi siyah Adım 2 ye git.
- X iki tane siyah düğüme sahip. X'in kardeşi S de 2 tane siyah düğüme sahip (Adım 2-A1).
- **Adım 2-A1:** S'in her iki çocuğu da siyah ise, X, P, S yeniden renklendir.



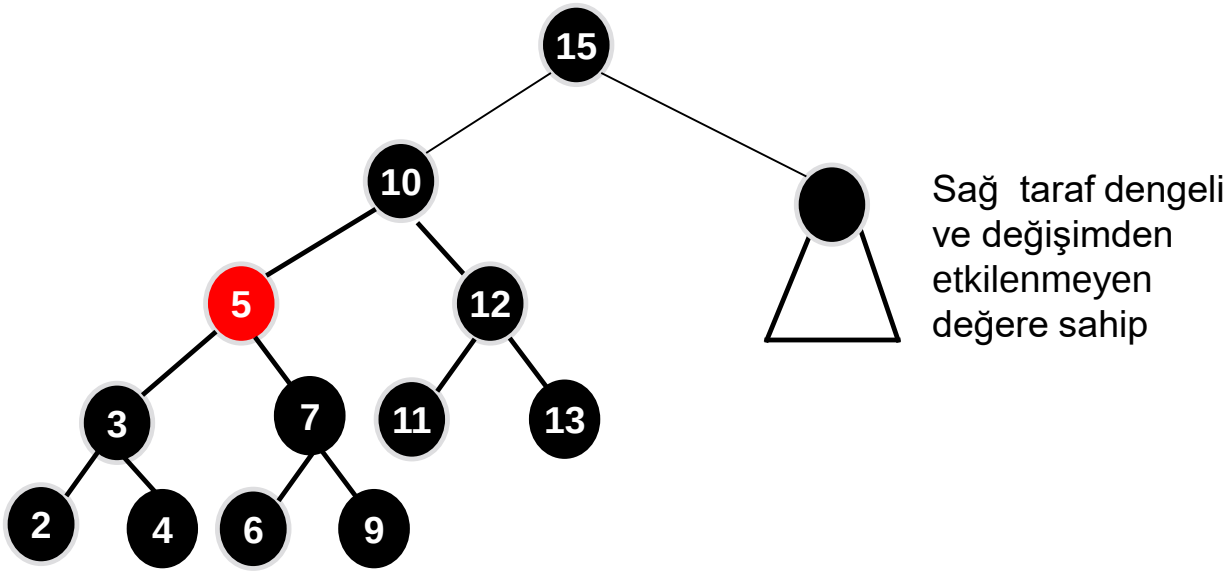
Örnek

- Adım 3: Kırmızı olan 10 düğümü artık silinebilir.
- Adım 4: Kökü siyah yap.



Örnek

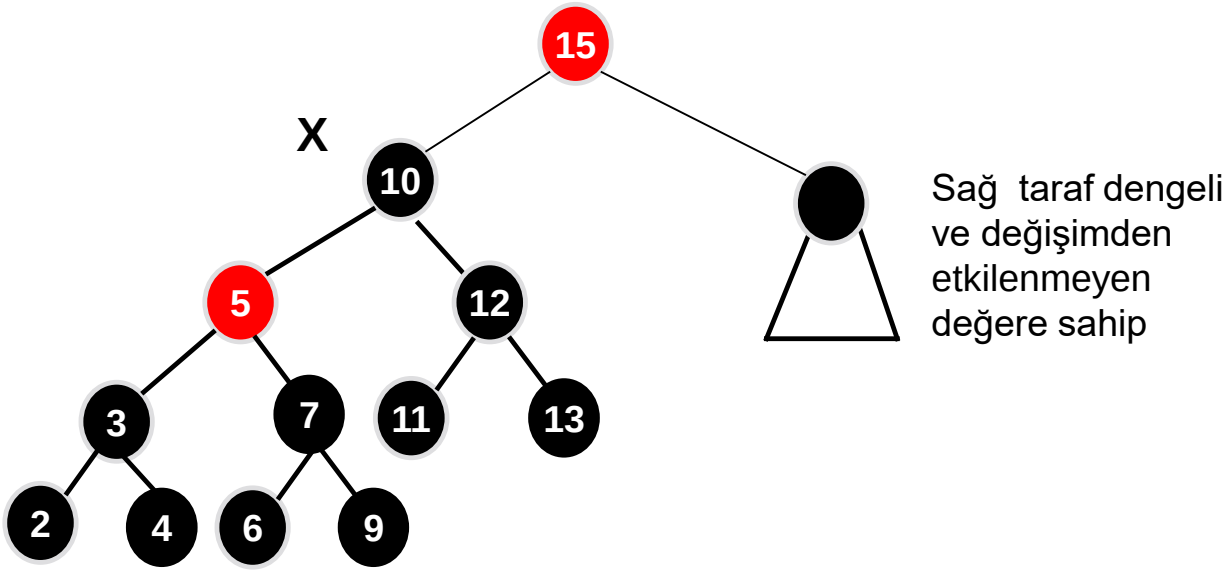
- **Örnek: R-B** ağacından 11 değerini siliniz.



- **Adım 1:** Kök iki tane siyah çocuğa sahip. Kökü kırmızı yap. X'i uygun çocuğa taşı (X(10)).

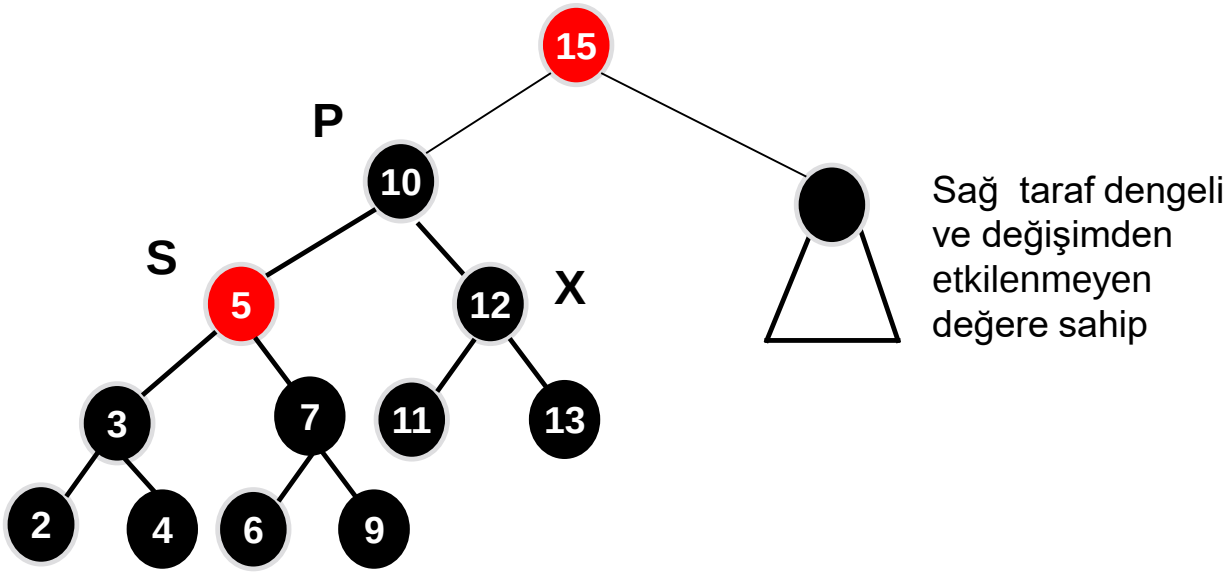
Örnek

- X kırmızı bir çocuğa sahip (Adım 2). X'i 12 ye taşı .

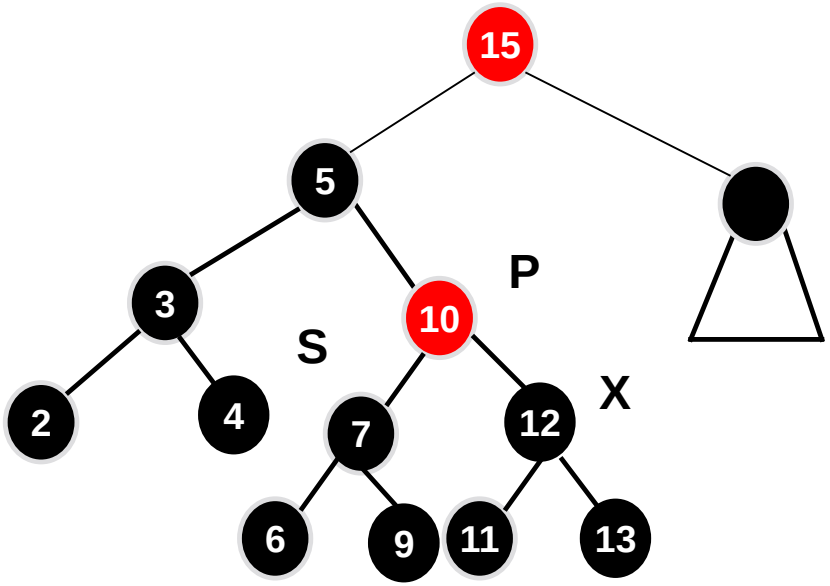


Örnek

- **Adım 2-B2:** Eğer yeni X siyah ise (P Siyah, S kırmızı), S'yi P' nin etrafında döndür P ve S'yi yeniden renklendir ve Adım 2'ye git.



Örnek

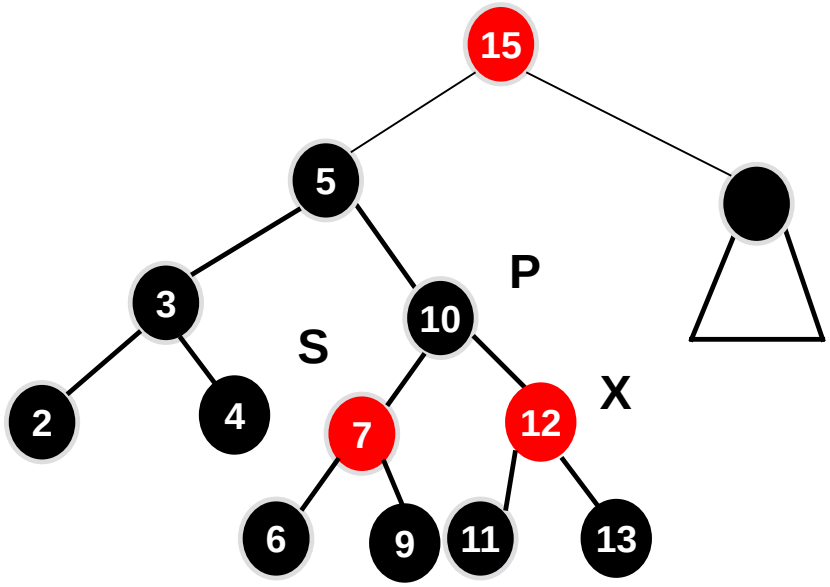


Sağ taraf dengeli ve değişimden etkilenmeyen değere sahip

- X ve S siyah çocuklara sahip (Adım 2-A1). X,P ve S' yi yeniden renklendir.

Örnek

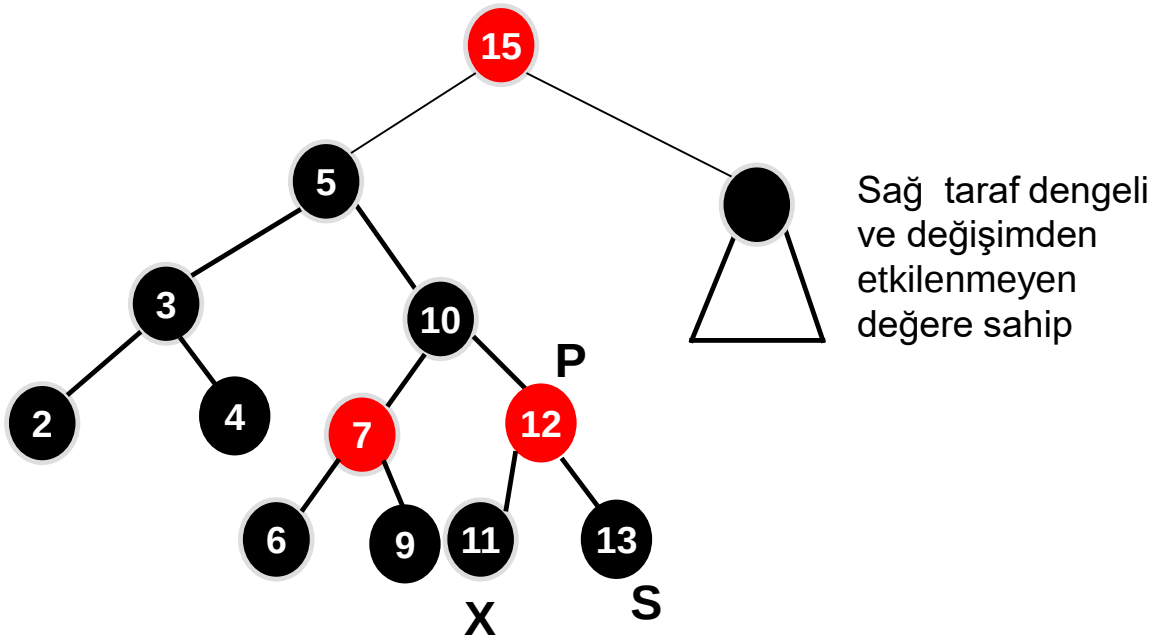
- X, P ve S'yi yeniden renklendir. X'i 11 taşı.



Sağ taraf dengeli ve değişimden etkilenmeyen değere sahip

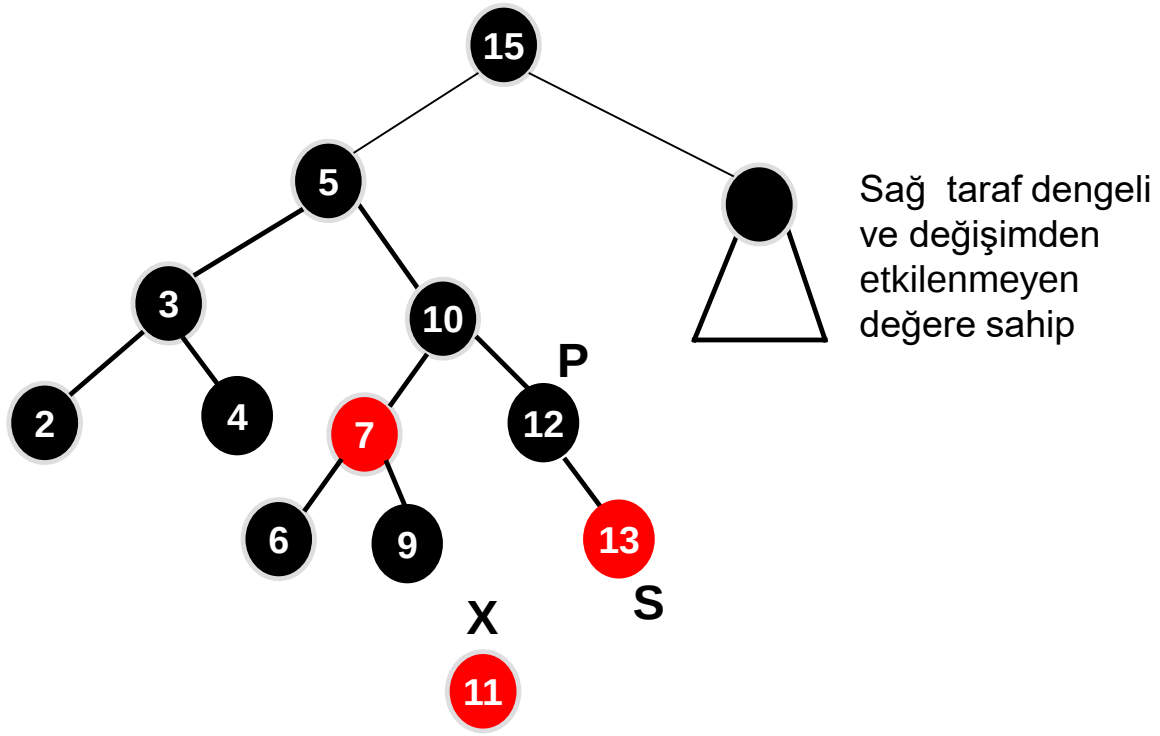
Örnek

- X'i 11 taşı. X siyah olduğundan silinemez. X ve S siyah çocuklara sahip (Adım 2-A1). X,P ve S' yi yeniden renklendir.



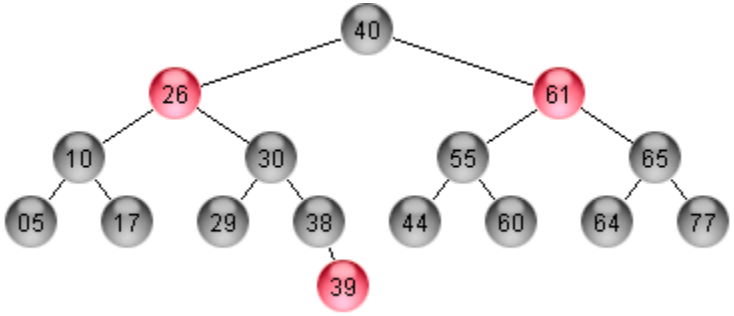
Örnek

- Adım 3: X'i sil.
- Adım 4: Kökü siyah yap

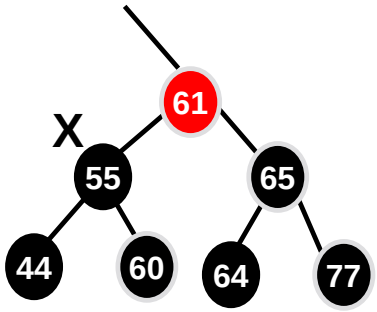


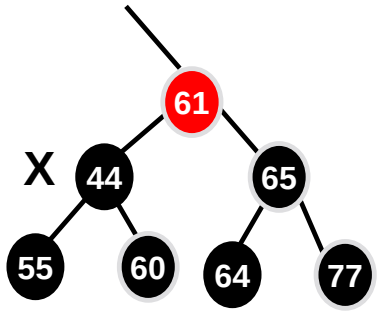
Örnek

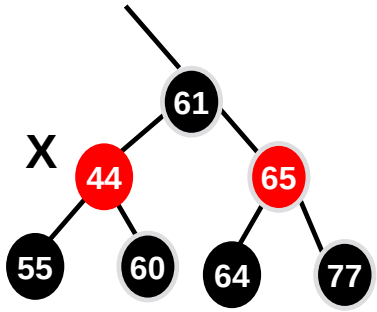
- Örnek: Aşağıda verilen red-black ağacından 55 nolu düğümü siliniz.



- Silinecek 55 nolu düğüm ile yerine gelecek düğümü (44) yer değiştir.
- Adım1: Kök düğümün çocukları kırmızı, kökü kırmızı yap X=kök. X'i uygun çocuğa taşı X(61). Yeni durum Adım 2-A1'i uygula.

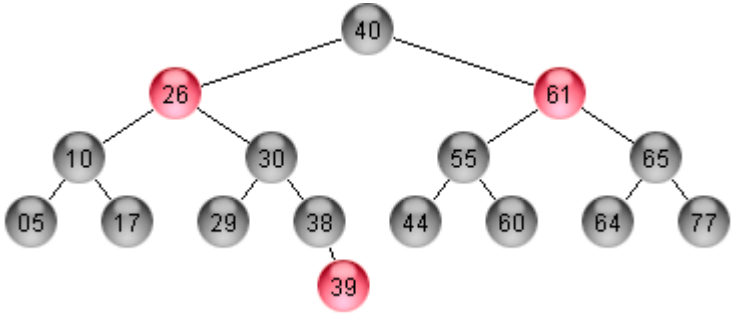


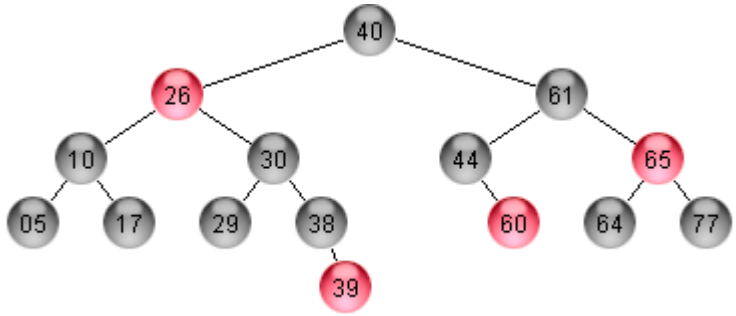




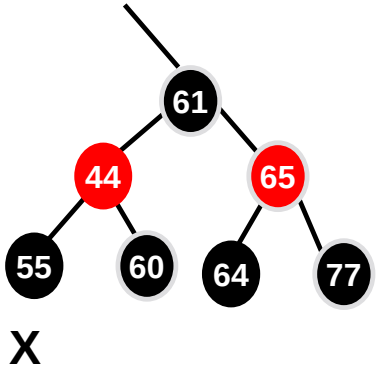
Örnek

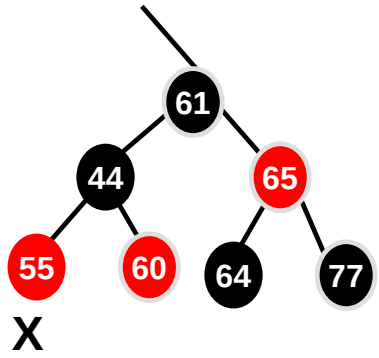
- Örnek: Aşağıda verilen red-black ağacından 55 nolu düğümü siliniz.

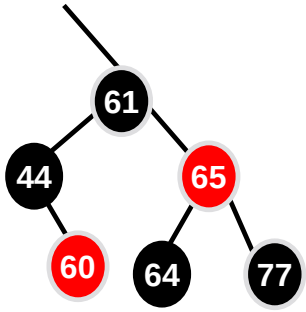




- X'i taşı X(55). Yeni durum Adım 2-A1'i uygula
- Adım 3 (X'i sil) ve Adım 4 uygula

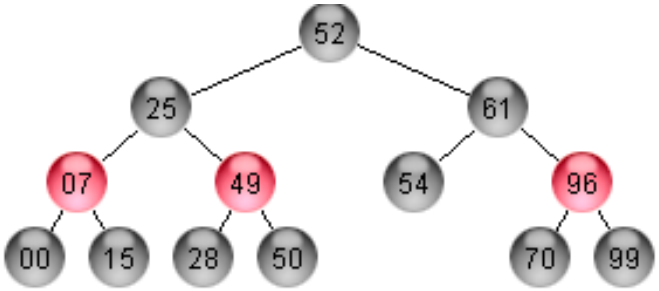


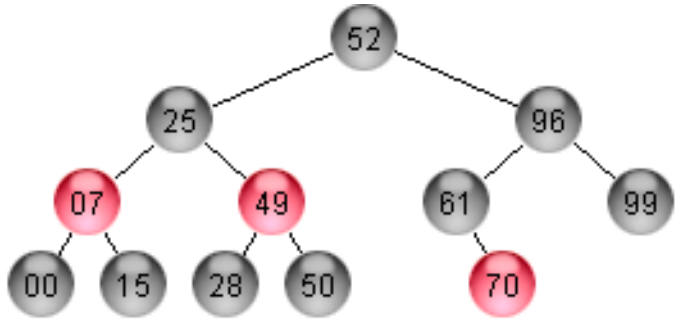




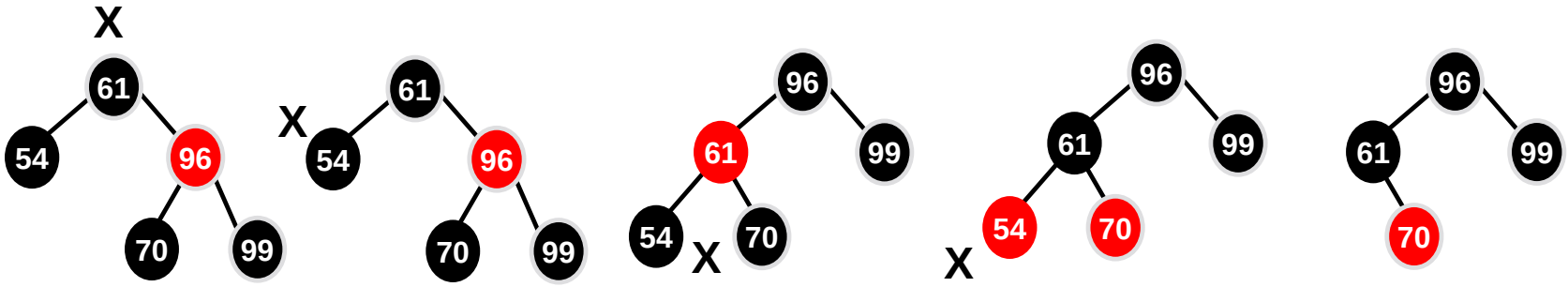
Örnek

- Örnek: Aşağıda verilen red-black ağacından 54 nolu düğümü siliniz.



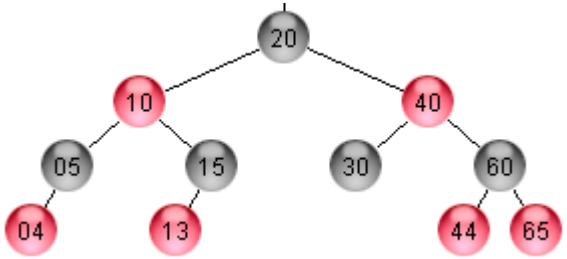


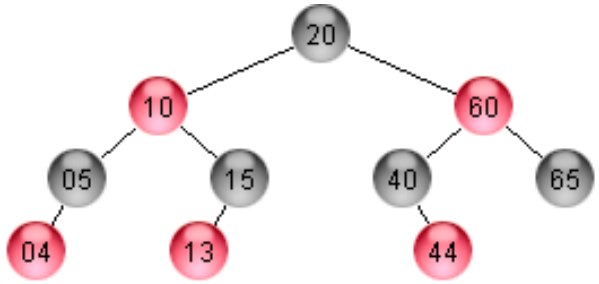
- Adım 1'i uygula X'i taşı (X(61)). Yeni durum Adım 2-B2'yi uygula (X(54))
- Yeni durum Adım 2-A1'i uygula. Adım 3 ve 4'ü uygula



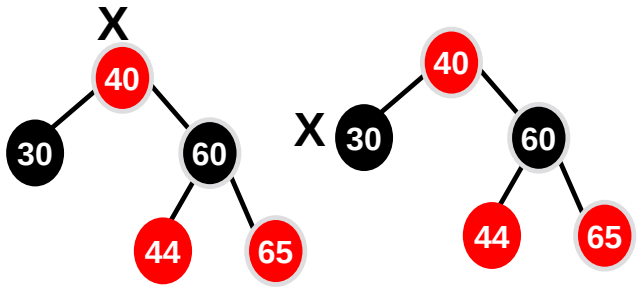
Örnek

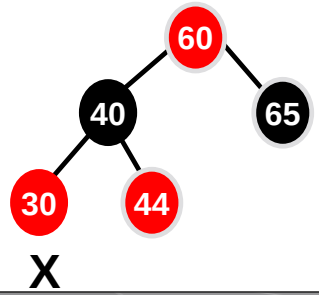
- Örnek: Aşağıda verilen red-black ağacından 30 nolu düğümü siliniz.

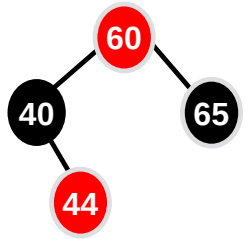




- Adım1'i uygula X=kök. X'i ,'i uygun çocuğa taşı X(30). Yeni durum Adım 2-A3'ü uygula. Adım 3 ve 4'ü uygula.

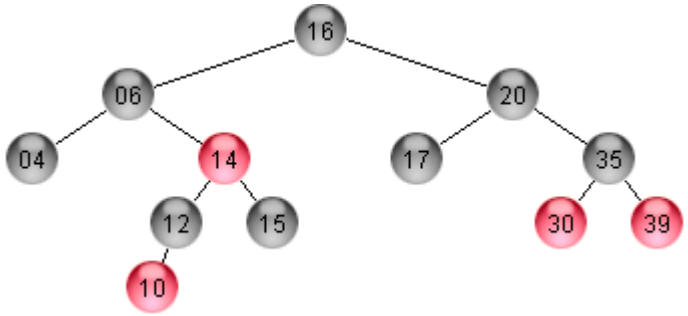




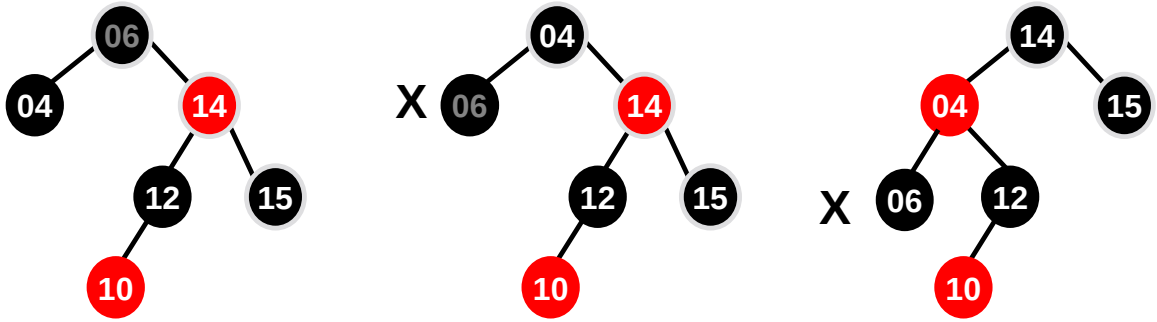


Örnek

- Örnek: Aşağıda verilen red-black ağacından 6 nolu düğümü siliniz.

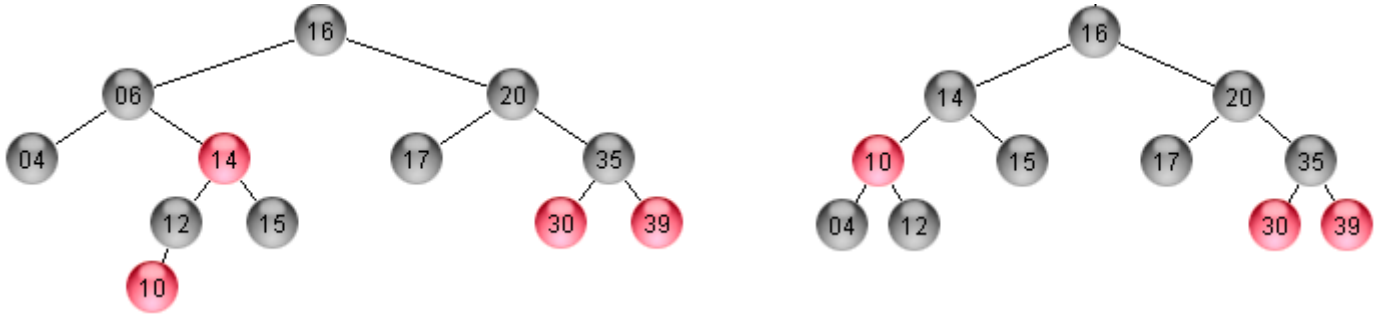


- 4 ile 6 yı yer değiştir.
- Adım 1' uygula
- X'i uygun çocuğa taşı X(4). Yeni durum Adım 2-B
- X'i uygun çocuğa taşı X(6). Yeni durum Adım 2-B2'yi uygula.

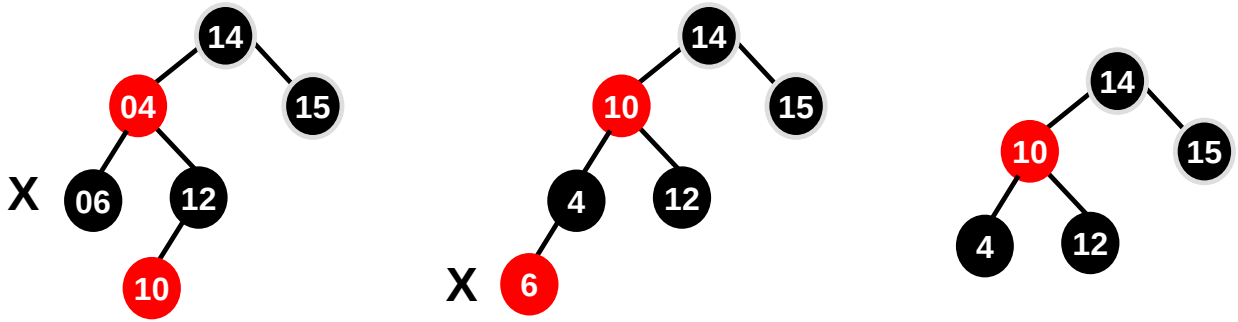


Örnek

- Örnek: Aşağıda verilen red-black ağacından 6 nolu düğümü siliniz.

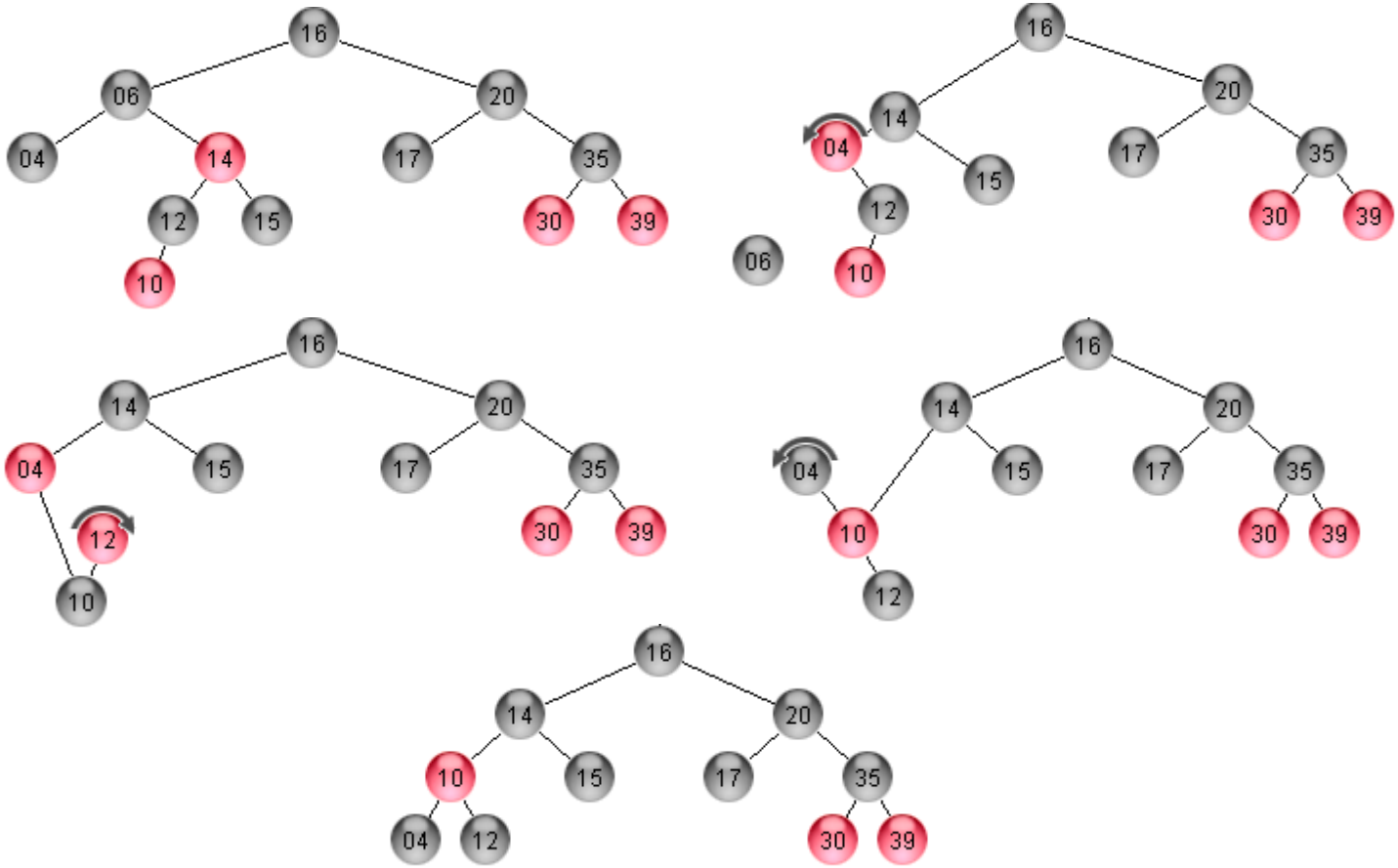


- X'i uygun çocuğa taşı X(6). Yeni durum Adım 2-B2'yi uygula.
- Yeni durum Adım 2-A2'i uygula. Adım 3 ve 4'ü uygula



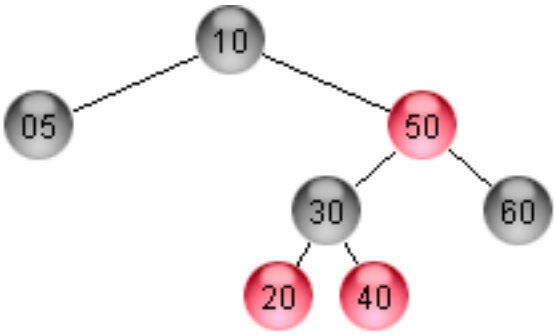
Örnek

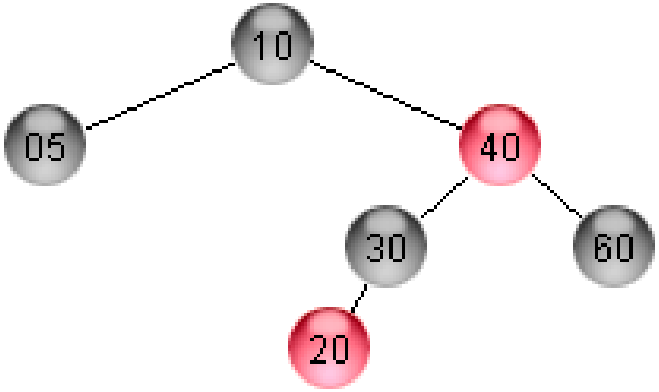
- Örnek: Aşağıda verilen red-black ağacından 6 nolu düğümü siliniz.



Red-Black Tree: Deletion

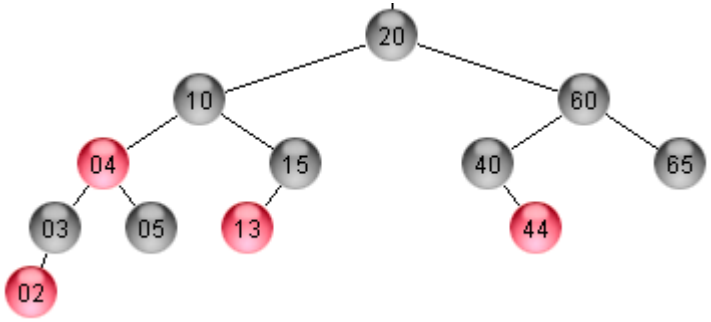
- Silinecek düğüm kırmızı, siyah çocukları var ise renk değişmez. (50 silindi)

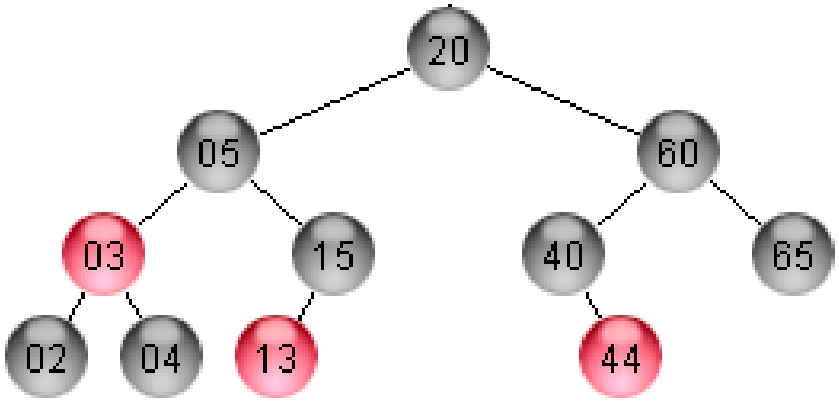




Red-Black Tree: Deletion

- Örnek: 10 değerini siliniz.

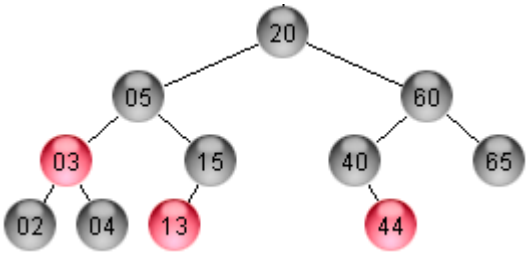


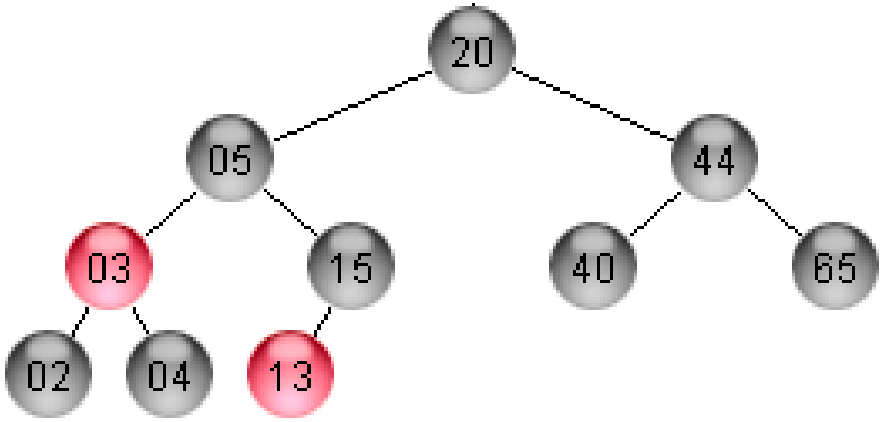


- 4B, 3R, 2B, 4 Sağ

Red-Black Tree: Deletion

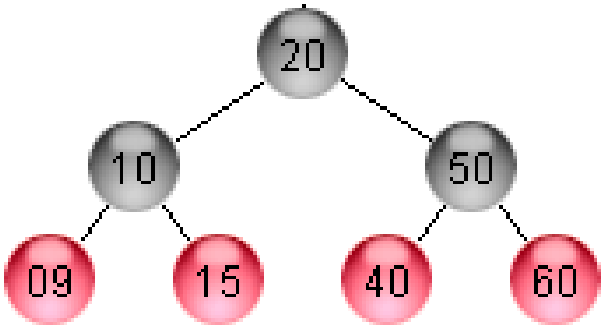
- Örnek:60 değerini siliniz

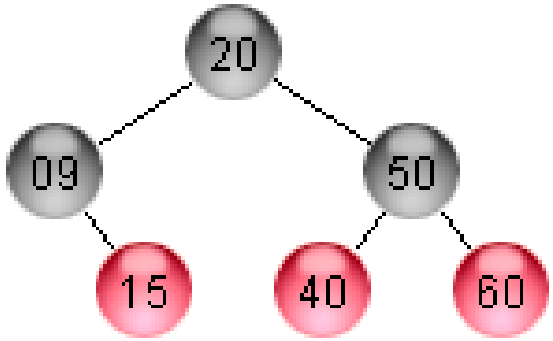


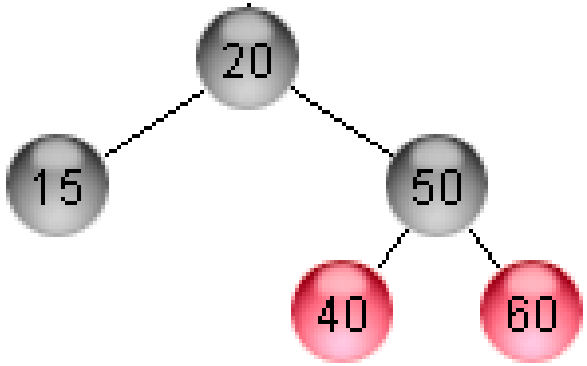


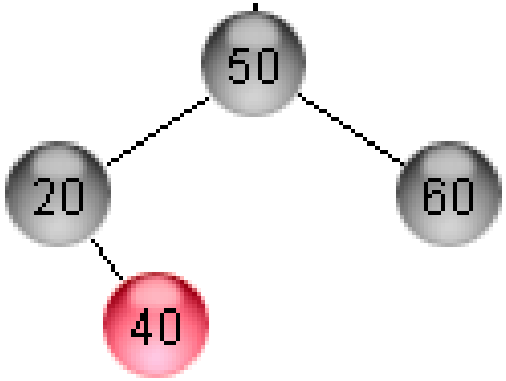
Red-Black Tree: Deletion

- Örnek
- Sırasıyla 10, 9,15, silindi.



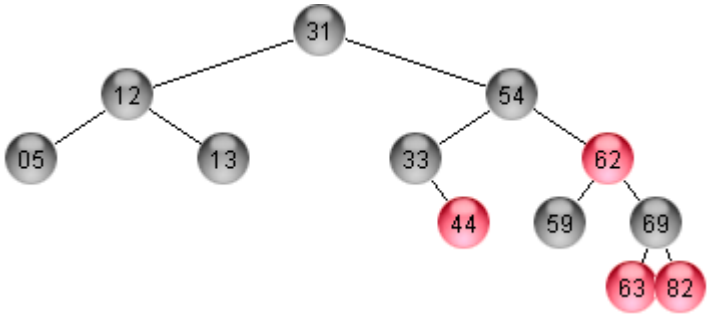


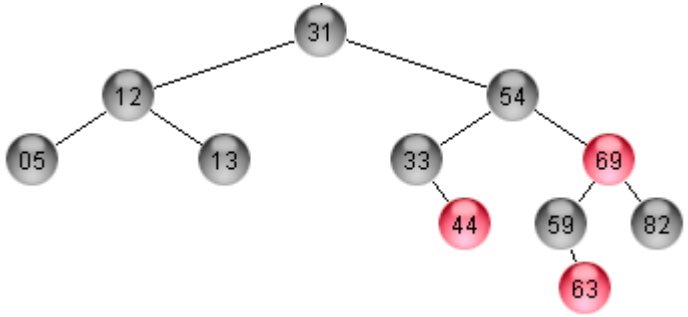




Red-Black tree Silme

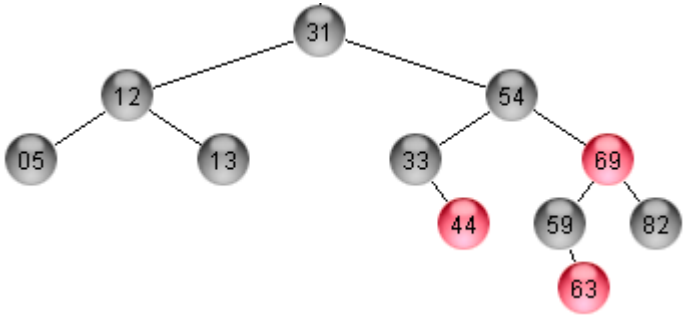
- Örnek: 62 silinecek

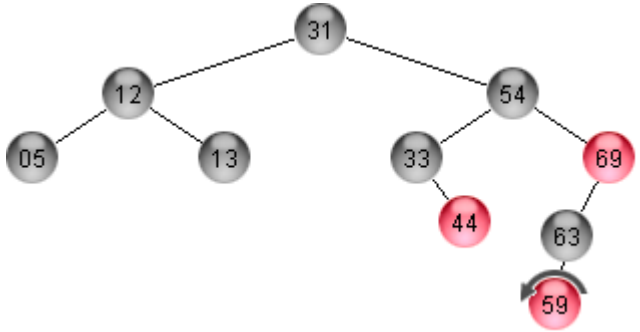


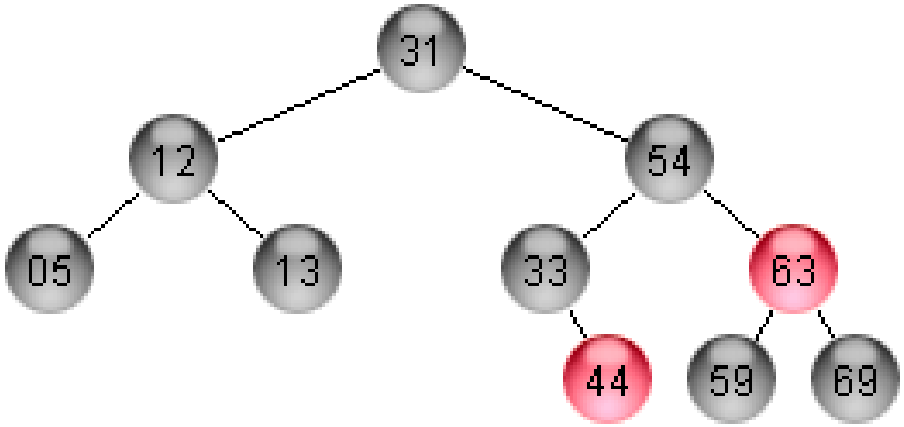


Red-Black tree Silme

- Örnek: 82 silinecek

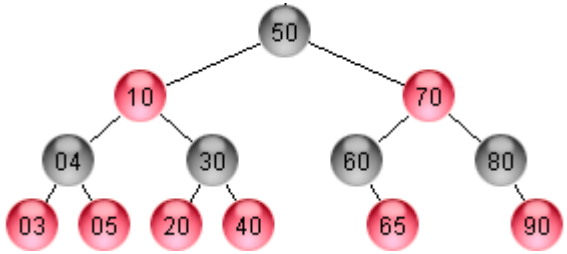




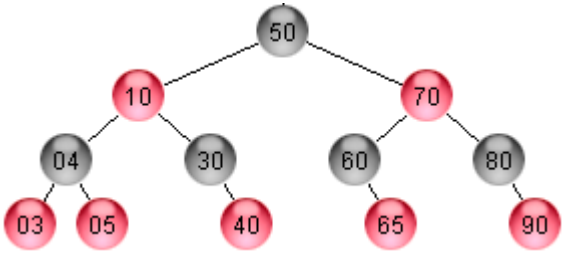


Red-Black tree Silme

- (Standart olması açısından silinen düğüm yaprak değil ise genelde soldaki en büyük düğümü alacağız.)

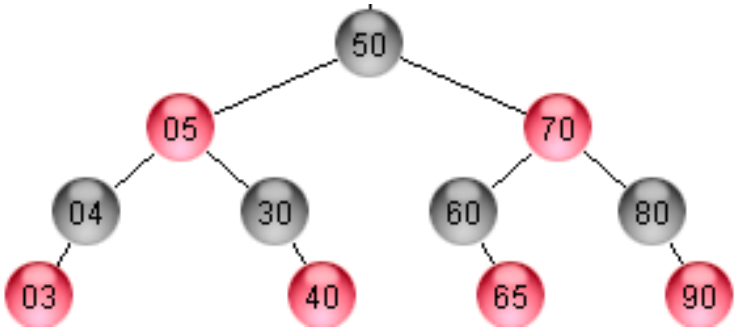


- Sil:20,10, 70, 30,50
- 20 silindi

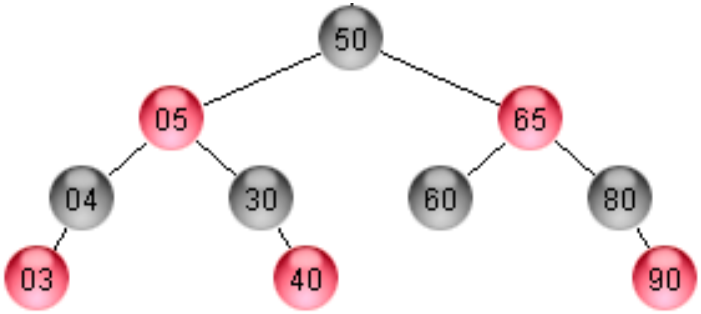


Red-Black tree Silme

- Sil:10,70,30,50,
- 10 silindi

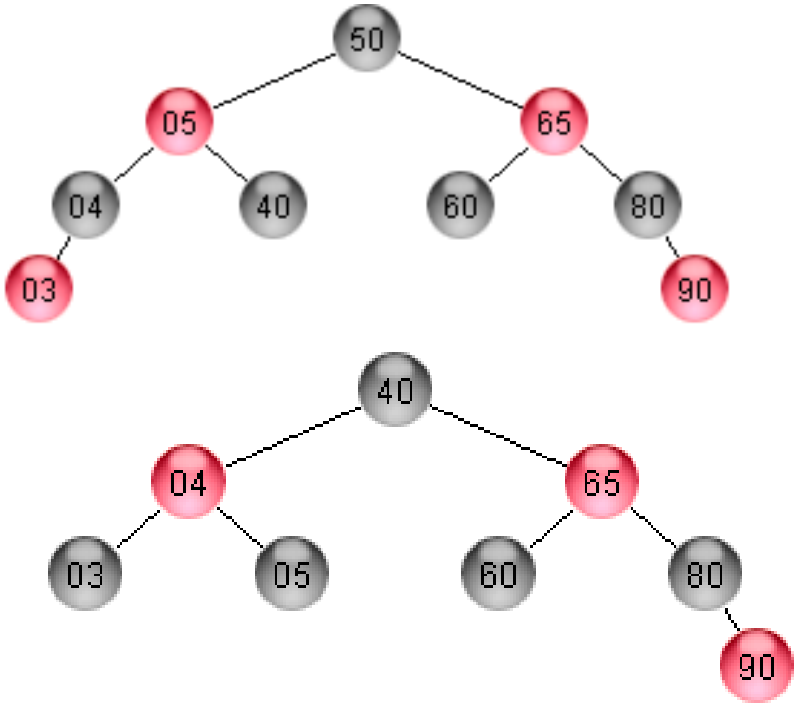


- 70 silindi



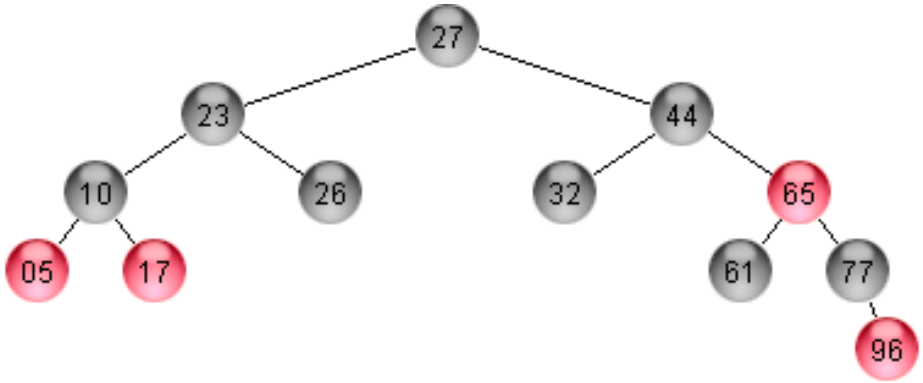
Red-Black tree Silme

- Sil:30,50,
- 30 silindi.
- 50 silindi.



Red-Black tree Silme

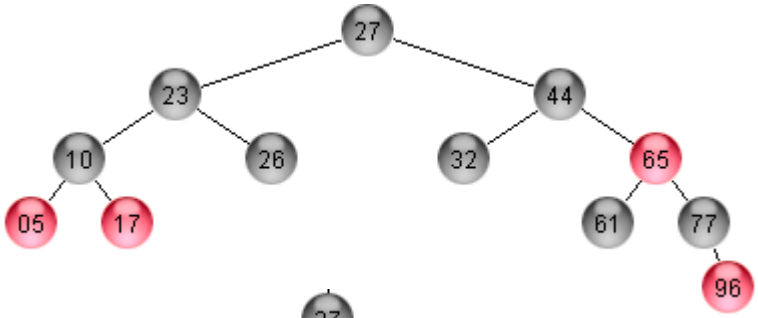
- Aşağıda verilen ağaçta sırasıyla 96, 44, 23, 27 düğümlerini silerek aşağıdaki soruları cevaplayınız.

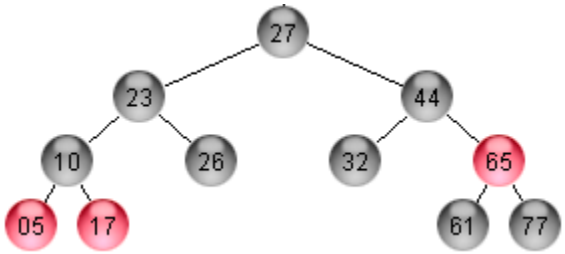


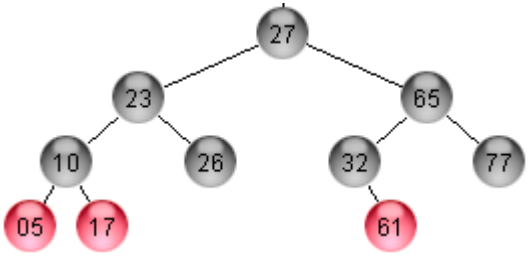
- (Doğru değer dışında yazılan değerler olursa sonuç yanlış olarak kabul edilecektir.)
- A) Hangi düğümler kırmızı renge sahiptir?
- B) Kök düğümün değeri nedir?

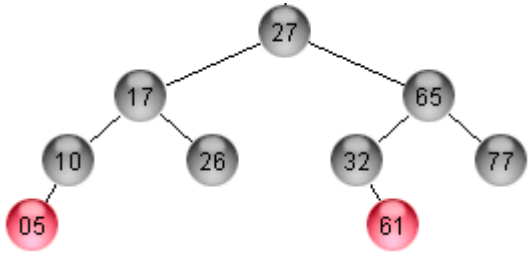
Red-Black tree Silme

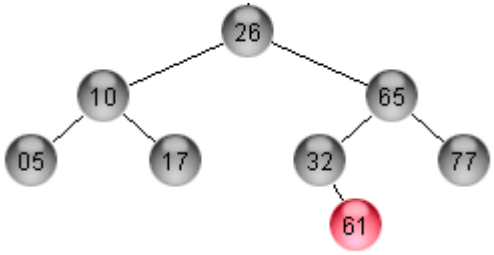
- 96, 44, 23, 27







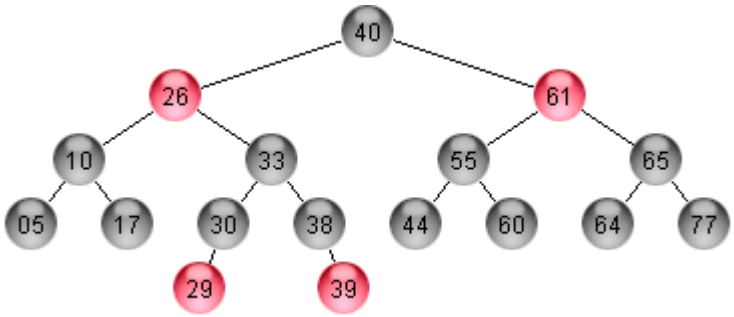




A) 61, B) 26

Red-Black tree Silme

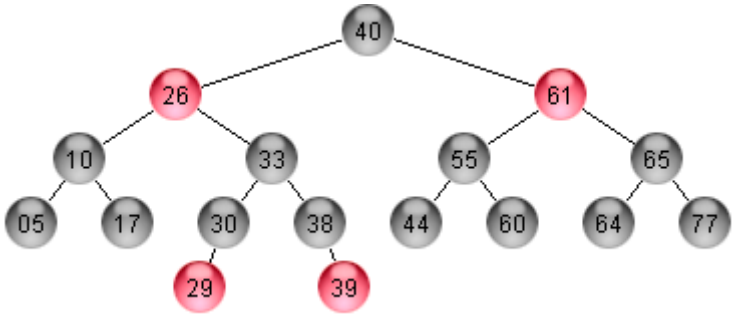
- Aşağıda verilen ağaçta sırasıyla 33, 55, 40, 10 düğümlerini silerek aşağıdaki soruları cevaplayınız.

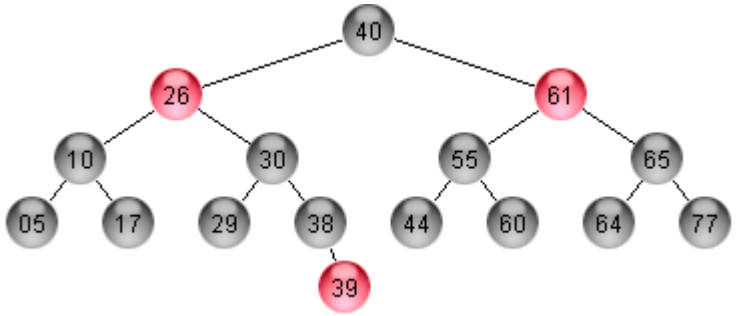


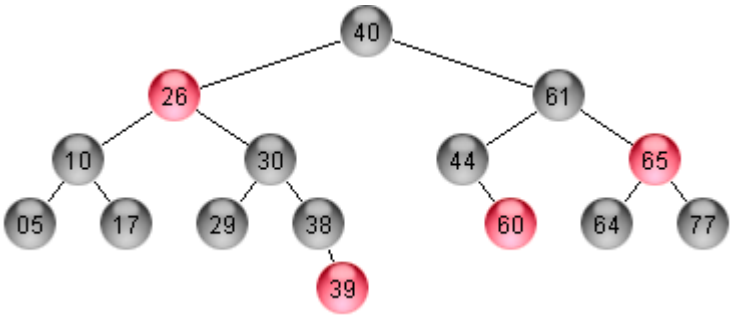
- (Doğru değer dışında yazılan değerler olursa sonuç yanlış olarak kabul edilecektir.)
- A) Kaç adet Siyah renge sahip düğüm vardır?
- B) Yaprakta bulunan hangi düğümler kırmızı renge sahiptir? Kök düğümün değeri nedir?
- C) Ağacın son hali AVL ağacı olarak dengede midir?
- Dengede değil ise dengeleyiniz.

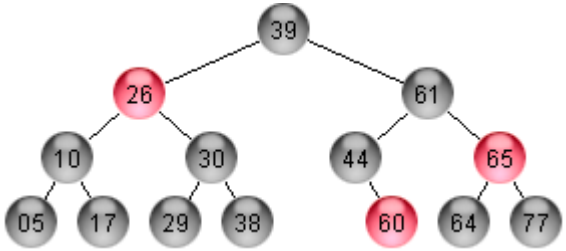
Red-Black tree Silme

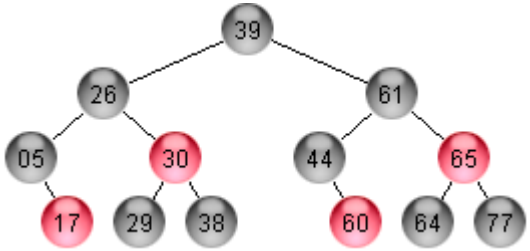
- 33, 55, 40, 10







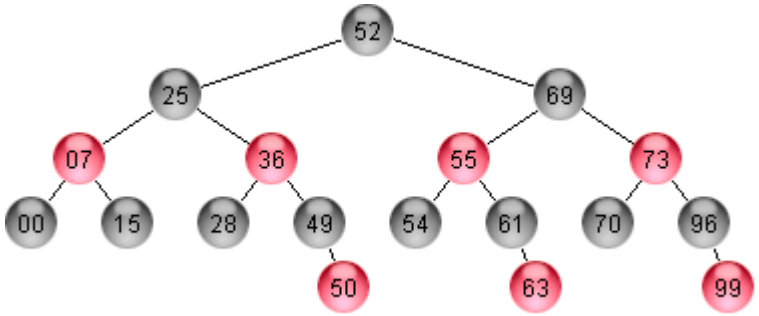


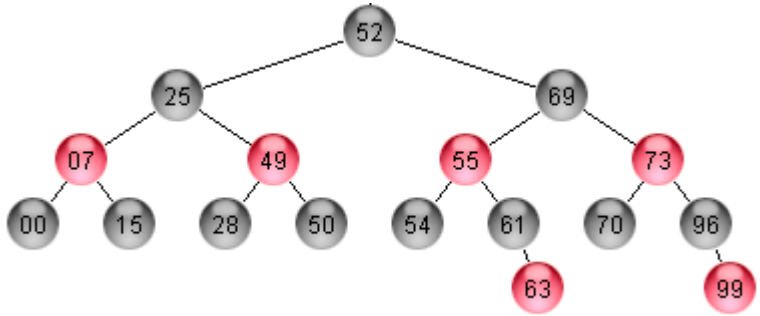


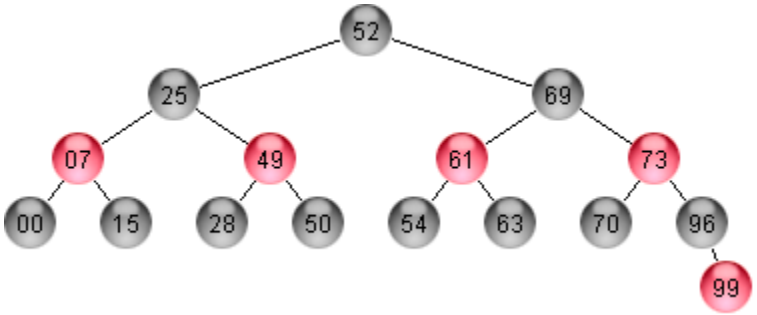
A) 9 adet, B) 17,60 C) Evet

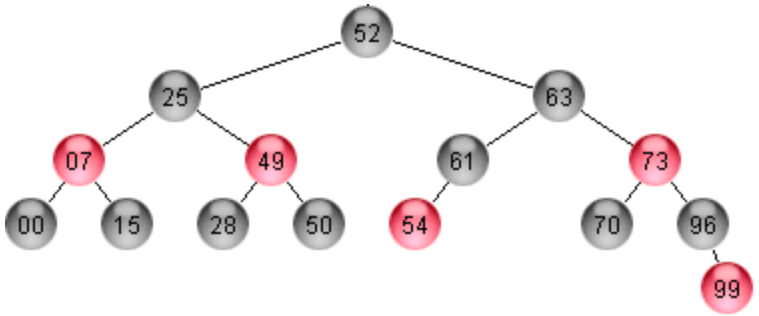
Red-Black tree Silme

- Silinecek değerler :36,55,69









Red-Black tree Silme

- Silinecek değerler: 73,63,54