

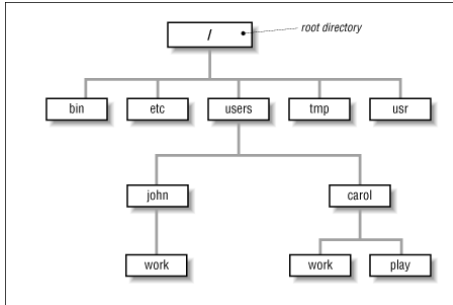
Anahat

- 1 Ağaçlar
- 2 İkili Ağaçlar
Dolaşma/Traversal
- 3 İkili Arama Ağaçları
Arama, Ekleme ve Silme
- 4 İfade Ağaçları

Ağaçlar

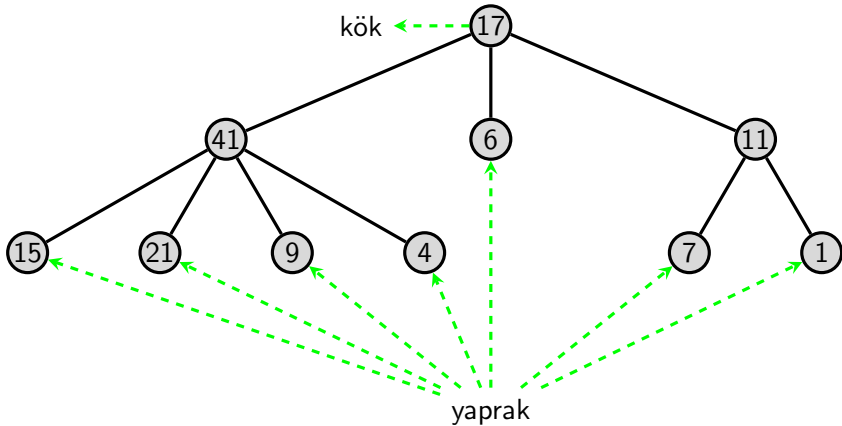
Tanım

Ağaçlar, düğümlerden ve düğümleri birbirine bağlayan kenarlardan oluşan; herhangi iki düğümü birbirine bağlayan sadece bir yolun bulunduğu veri yapılarıdır.



- ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡

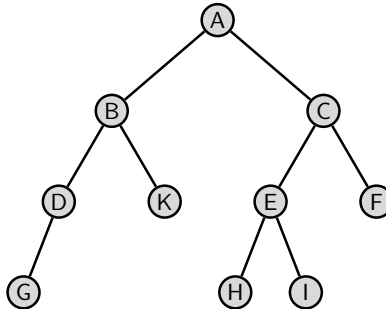
Örnek ağaç



İkili Ağaçlar

Tanım

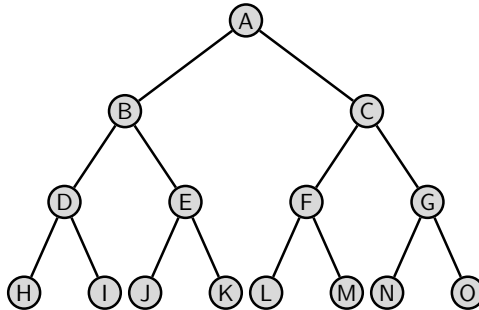
Her bir düğümünde en fazla iki çocuğu olan ağaçlara ikili ağaçlar denir. Bir düğümün sol ve sağ çocuğu olabilir.



Dolu İkili Ağaçlar/Full Binary Tree

Tanım

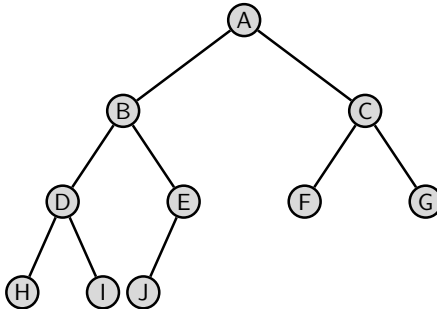
Tüm yaprakları aynı seviyede olan ikili ağaçlardır.



Bütün İkili Ağaçlar/Complete Binary Tree

Tanım

Dolu ikili ağaca yaprak değerleri soldan itibaren eklendiğinde oluşan ikili ağaç.

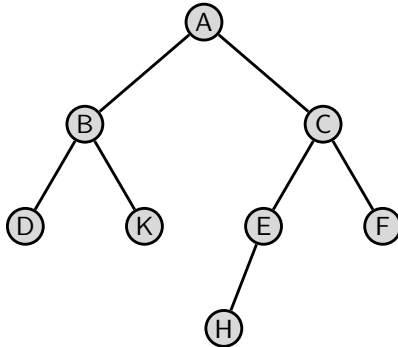


Ağaçlarda dolaşma/Traversal

İkili ağaçlarda üç farklı dolaşma şekli vardır:

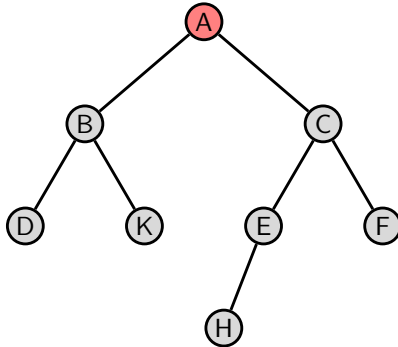
- **Önce değer(Preorder):** Önce değer elde edilir, sonra sırasıyla sol ve sağ alt ağaç önce değer olarak dolaşılır.
- **Ortada değer(Inorder):** Önce sol alt ağaç ortada değer olarak dolaşılır, sonra değer elde edilir, son olarak sağ alt ağaç ortada değer olarak dolaşılır.
- **Sonra değer(Postorder):** Önce sırasıyla sol ve sağ alt ağaçlar sonra değer şeklinde gezilir, sonra değer elde edilir.

Preorder dolaşma



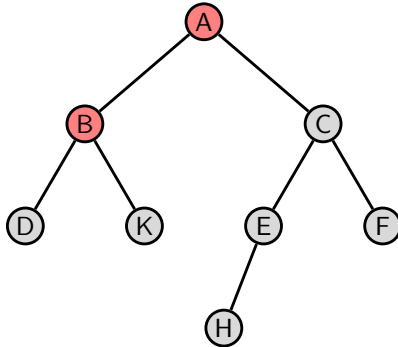
Preorder dolaşma 1

Dolaşma: A



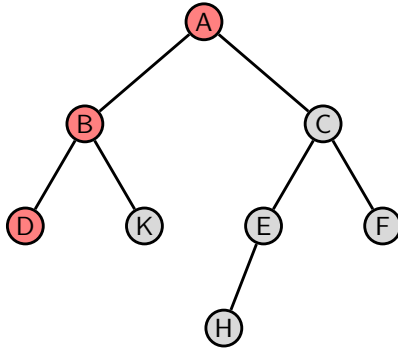
Preorder dolaşma 2

Dolaşma: A, B



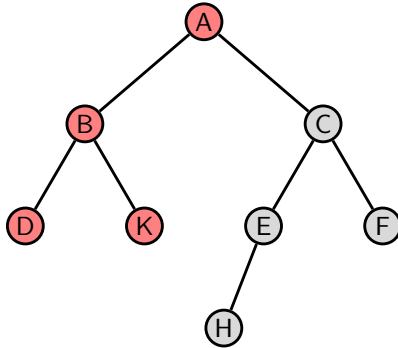
Preorder dolaşma 3

Dolaşma: A, B, D



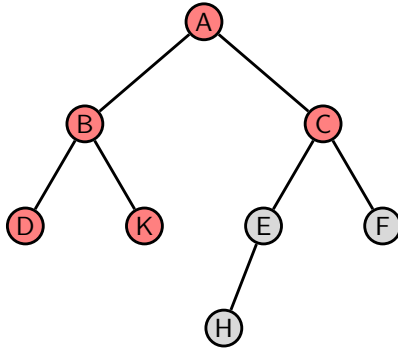
Preorder dolaşma 4

Dolaşma: A, B, D, K



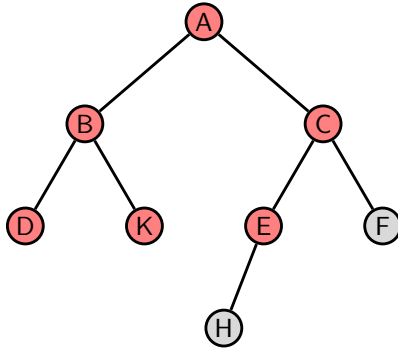
Preorder dolaşma 5

Dolaşma: A, B, D, K, C



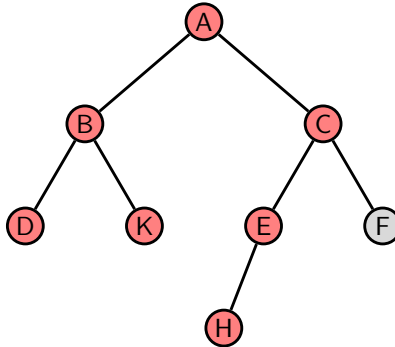
Preorder dolaşma 6

Dolaşma: A, B, D, K, C, E



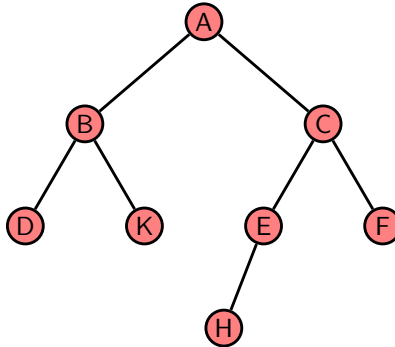
Preorder dolaşma 7

Dolaşma: A, B, D, K, C, E, H

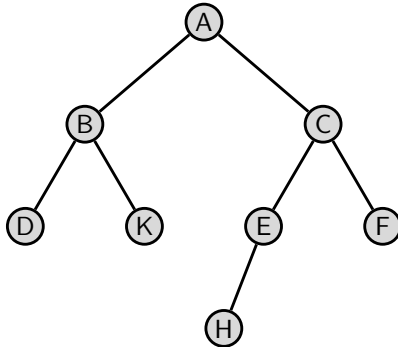


Preorder dolaşma 8

Dolaşma: A, B, D, K, C, E, H, F

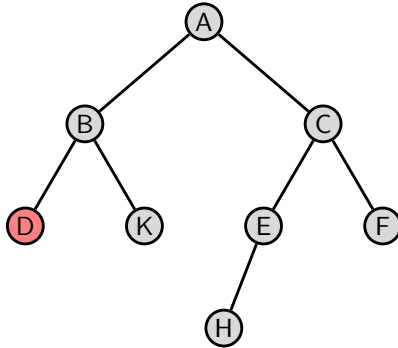


Inorder dolaşma



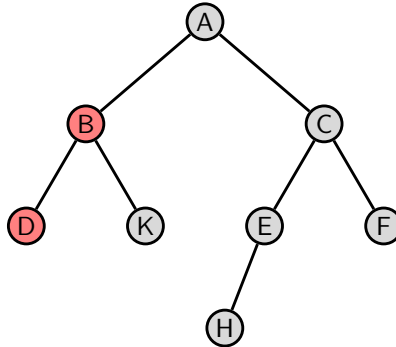
Inorder dolaşma 1

Dolaşma: D



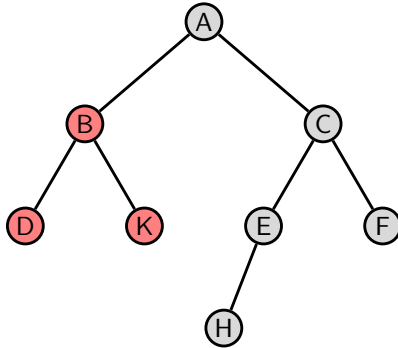
Inorder dolaşma 2

Dolaşma: D, B



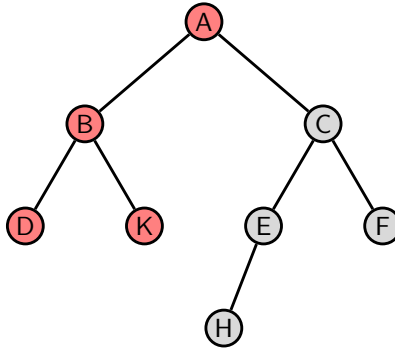
Inorder dolaşma 3

Dolaşma: D, B, K



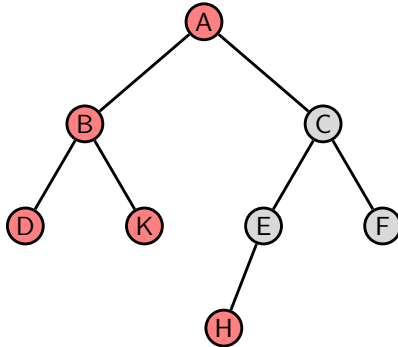
Inorder dolaşma 4

Dolaşma: D, B, K, A



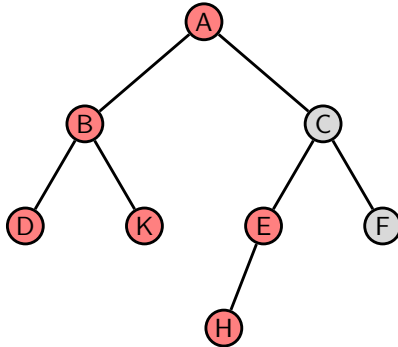
Inorder dolaşma 5

Dolaşma: D, B, K, A, H



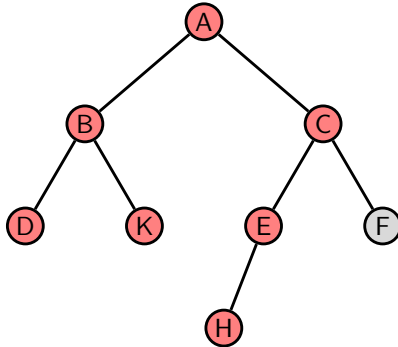
Inorder dolaşma 6

Dolaşma: D, B, K, A, H, E



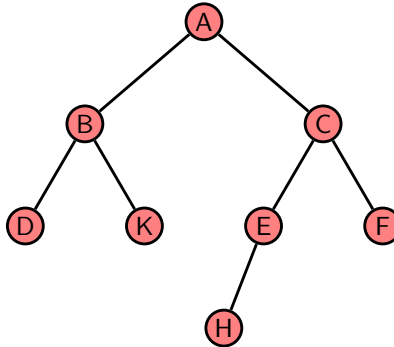
Inorder dolaşma 7

Dolaşma: D, B, K, A, H, E, C

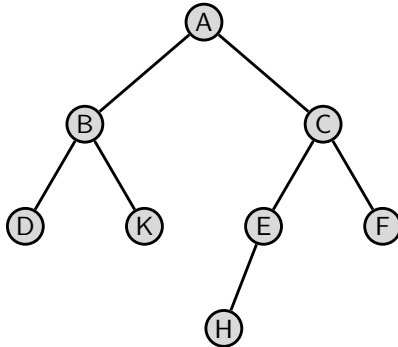


Inorder dolaşma 8

Dolaşma: D, B, K, A, H, E, C, F

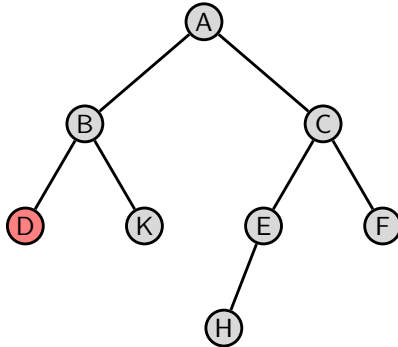


Postorder dolaşma



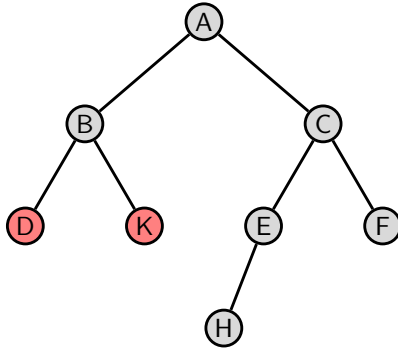
Postorder dolaşma 1

Dolaşma: D



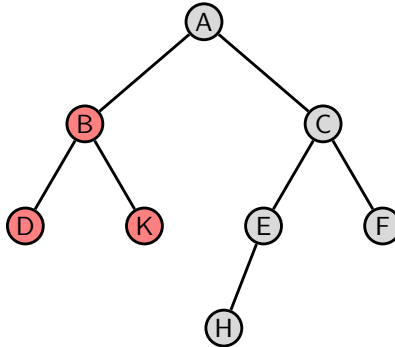
Postorder dolaşma 2

Dolaşma: D, K



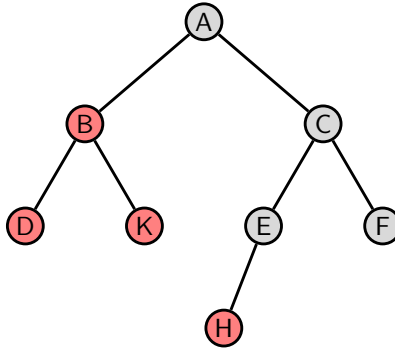
Postorder dolaşma 3

Dolaşma: D, K, B



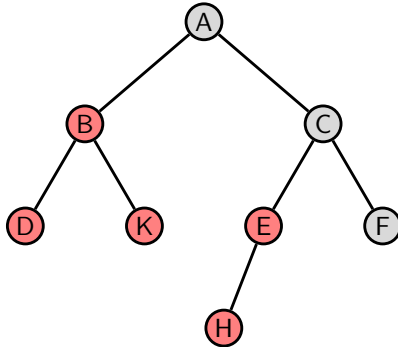
Postorder dolaşma 4

Dolaşma: D, K, B, H



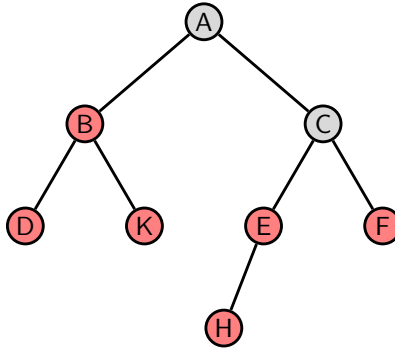
Postorder dolaşma 5

Dolaşma: D, K, B, H, E



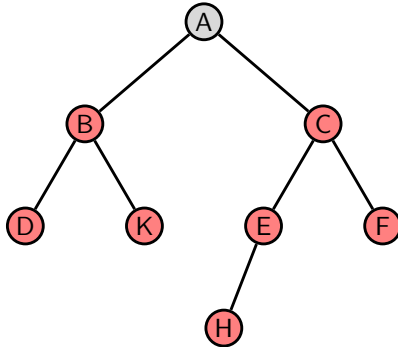
Postorder dolaşma 6

Dolaşma: D, K, B, H, E, F



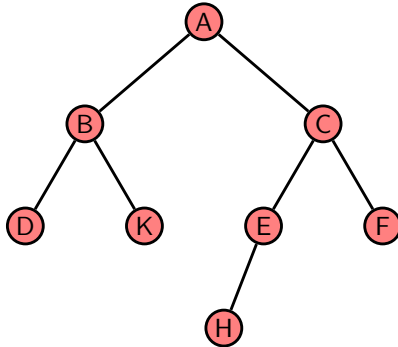
Postorder dolaşma 7

Dolaşma: D, K, B, H, E, F, C



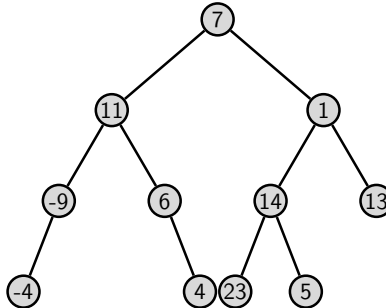
Postorder dolaşma 8

Dolaşma: D, K, B, H, E, F, C, A



Soru

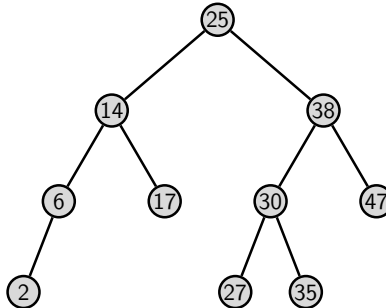
Aşağıdaki ikili ağacın preorder, inorder ve postorder dolaşimleri nasıl olur?



İkili Arama Ağaçları

Tanım

Kökün solundaki tüm anahtar değerleri kökten küçük, kökün sağındaki her bir anahtar değeri kökten büyük ve bu iki özelliği tüm düğümlerde geçerli olan ikili ağaç yapısıdır.



Arama İşlemi

Function Ara(*düğüm*, *eleman*):

if *düğüm null ise then*

return *null*

else

if *düğüm.değer==eleman then*

return *düğüm*

else if *eleman<düğüm.değer then*

return Ara(*düğüm.sol*, *eleman*)

else

return Ara(*düğüm.sağ*, *eleman*)

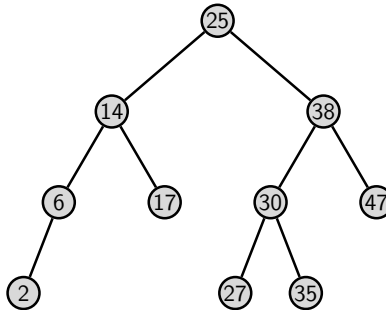
end

end

End Function

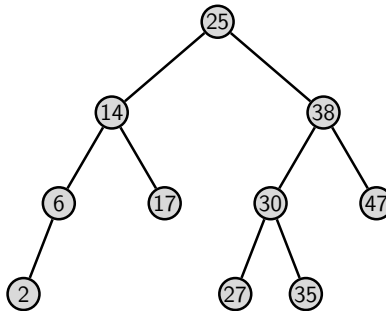
Arama işlemi

35 ve 15 değerlerini aşağıdaki ağaçta arayalım:



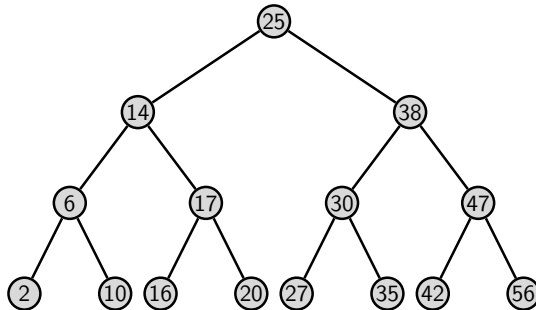
Arama işlemi

35 ve 15 değerlerini aşağıdaki ağaçta arayalım:

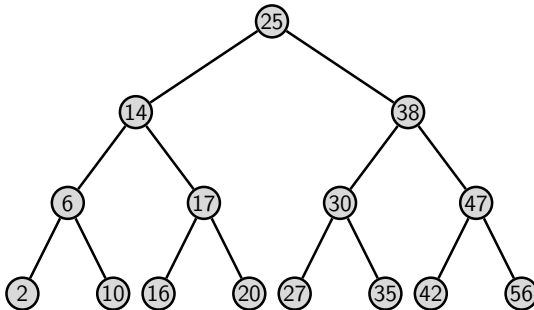


- Arama işlemi bu ağaçta en iyi ve en kötü ihtimalle kaç adımda biter?
- Karşılaşılabilecek en iyi ve en kötü ağaçlar nasıldır?

Arama en iyi durum

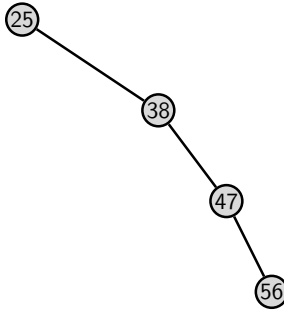


Arama en iyi durum

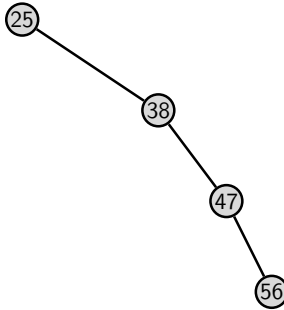


N elemanlı bir dengeli ağaçta en fazla kaç adımda aranılan değer bulunabilir?

Arama en kötü durum



Arama en kötü durum



N elemanlı bir ikili ağaçta en fazla kaç adımda aranılan değer bulunabilir?

Ekleme İşlemi

Function Ekle(*düğüm*, *eleman*):

```
    if düğüm.değer==eleman then
        | HATA veya ekleme yapma
    else if eleman<düğüm.değer then
        | if düğüm.sol null ise then
        | | düğüm.sol=yeni düğüm
        | end
        | else
        | | Ekle(düğüm.sağ, eleman)
        | end
    else
        | if düğüm.sağ null ise then
        | | düğüm.sağ=yeni düğüm
        | end
        | else
        | | Ekle(düğüm.sol, eleman)
        | end
    end
```

End Function

Ekleme işlemi

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8 değerlerini sırasıyla ikili arama ağacına ekleyin.

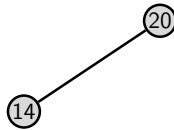
Ekleme işlemi: 20

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8

20

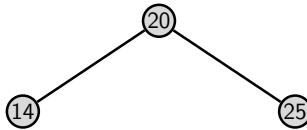
Ekleme işlemi: 14

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8



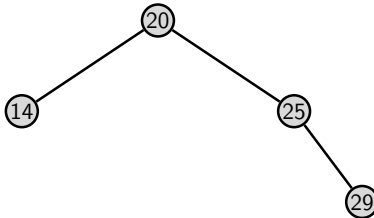
Ekleme işlemi: 25

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8



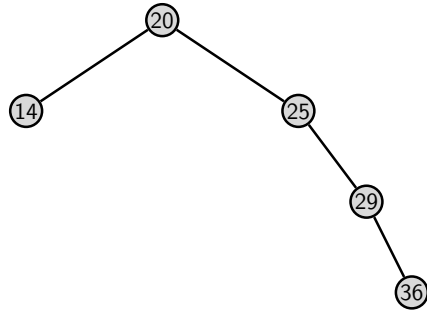
Ekleme işlemi: 29

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8



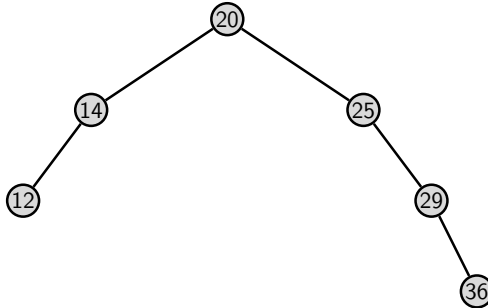
Ekleme işlemi: 36

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8



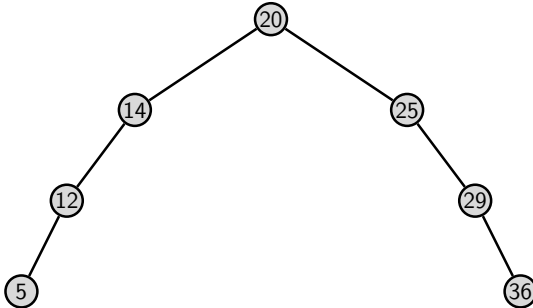
Ekleme işlemi: 12

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8



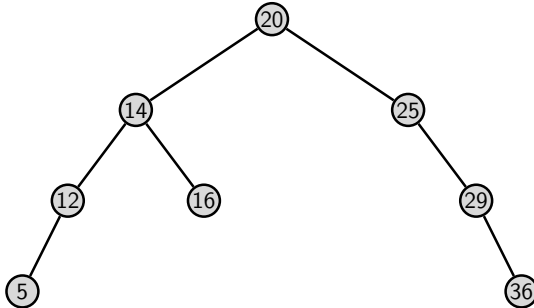
Ekleme işlemi: 5

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8



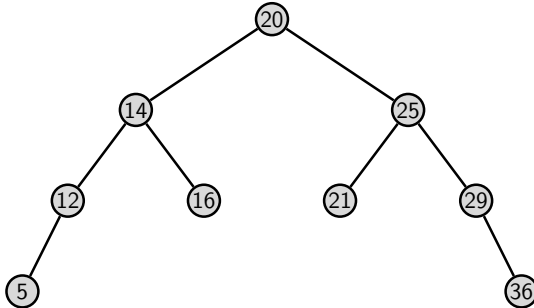
Ekleme işlemi: 16

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8



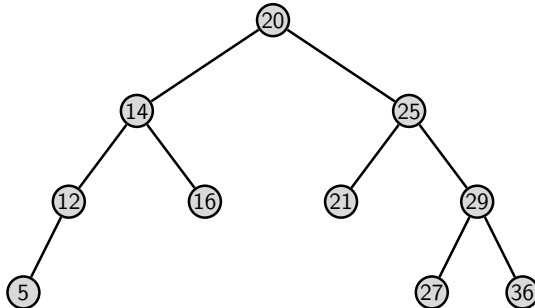
Ekleme işlemi: 21

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8



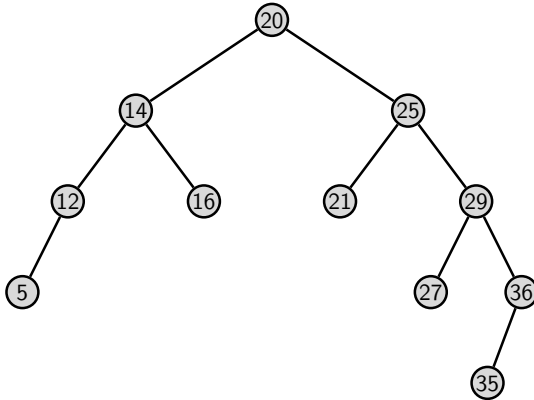
Ekleme işlemi: 27

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8



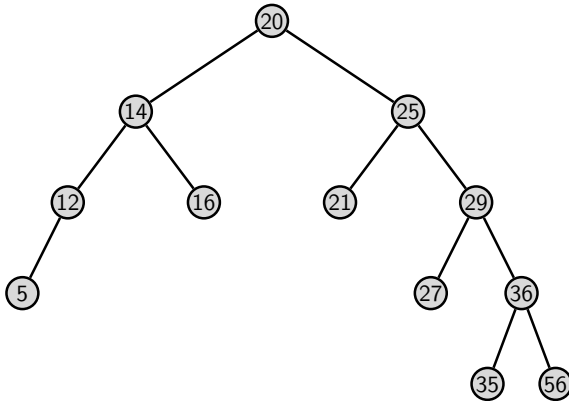
Ekleme işlemi: 35

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8



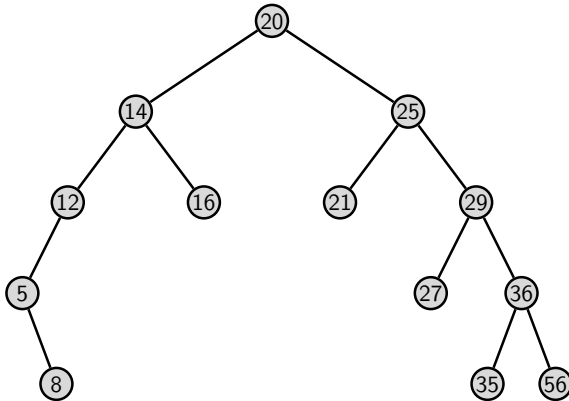
Ekleme işlemi: 56

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8



Ekleme işlemi: 8

20, 14, 25, 29, 36, 12, 5, 16, 21, 27, 35, 56, 8

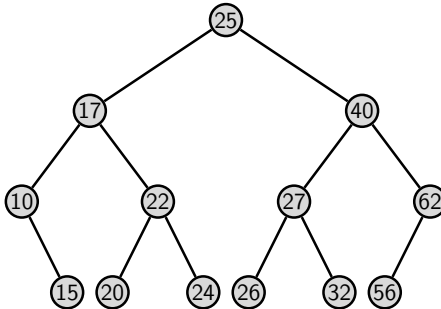


Ekleme işlemi

25, 17, 40, 22, 27, 62, 26, 32, 20, 56, 10, 24, 15 değerlerini sırasıyla ikili arama ağacına ekleyin.

Ekleme işlemi

25, 17, 40, 22, 27, 62, 26, 32, 20, 56, 10, 24, 15 değerlerini sırasıyla ikili arama ağacına ekleyin.



Silme işlemi

Silme için 3 farklı durum söz konusudur:

- Silinecek eleman yaprak ise
- Silinecek elemanın sadece bir çocuğu varsa
- Silinecek elemanın iki çocuğu da varsa

Silme işlemi

Silme için 3 farklı durum söz konusudur:

- Silinecek eleman yaprak ise
- Silinecek elemanın sadece bir çocuğu varsa
- Silinecek elemanın iki çocuğu da varsa

Ardıl/Successor

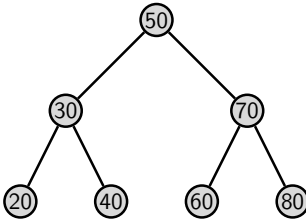
Ağaçta verilen değerden büyük değerlerin en küçüğüdür.

Öncel/Predecessor

Ağaçta verilen değerden küçük değerlerin en büyüğüdür.

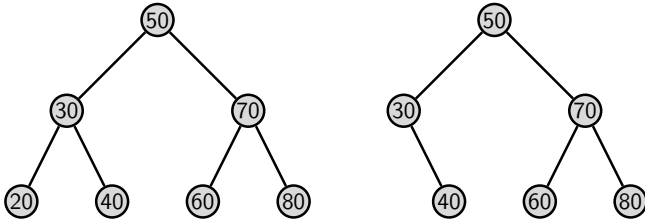
Silinecek eleman yaprak ise

Silinecek eleman yaprak ise ebeveyni ile bağlantısını kopartmak yeterlidir.
Delete 20



Silinecek eleman yaprak ise

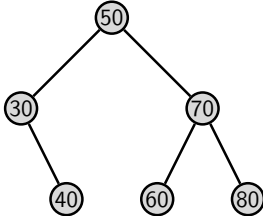
Silinecek eleman yaprak ise ebeveyni ile bağlantısını kopartmak yeterlidir.
Delete 20



Silinecek elemanın sadece bir çocuğu varsa

Silinecek elemanın sadece bir çocuğu varsa çocuk düğümü ilgili düğümün yerine kopyala ve düğümü sil.

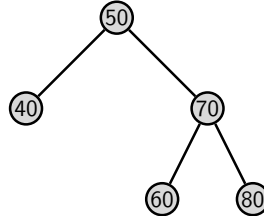
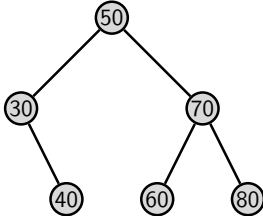
Delete 30



Silinecek elemanın sadece bir çocuğu varsa

Silinecek elemanın sadece bir çocuğu varsa çocuk düğümü ilgili düğümün yerine kopyala ve düğümü sil.

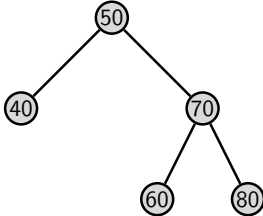
Delete 30



Silinecek elemanın iki çocuğu da varsa

Silinecek elemanın iki çocuğu da varsa düğümün ardılı(successor) bulunur ve silinir, silinen düğüm ilgili düğümün yerine yerleştirilir.

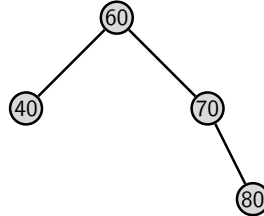
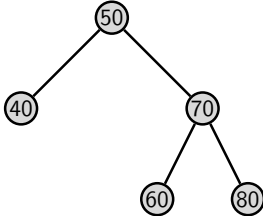
Delete 50



Silinecek elemanın iki çocuğu da varsa

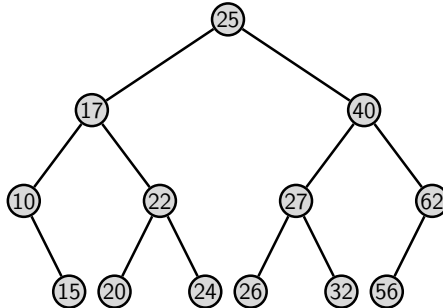
Silinecek elemanın iki çocuğu da varsa düğümün ardılı(successor) bulunur ve silinir, silinen düğüm ilgili düğümün yerine yerleştirilir.

Delete 50



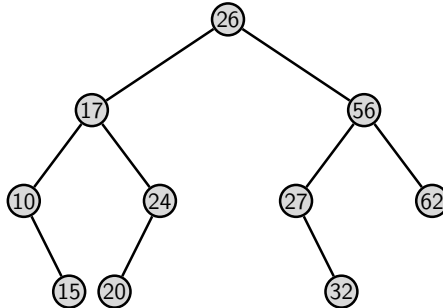
Silme işlemi

22, 25, 40 değerlerini sırasıyla aşağıdaki ikili arama ağacından silin.



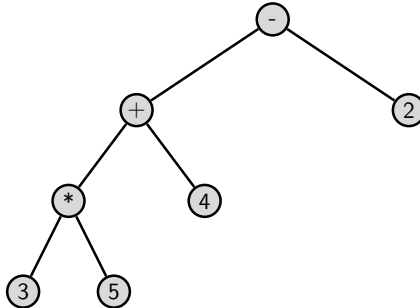
Silme işlemi

22, 25, 40 değerleri silindikten sonra.



İfade ağaçları

$3 * 5 + 4 - 2$ ifadesinin ifade ağacında gösterimi

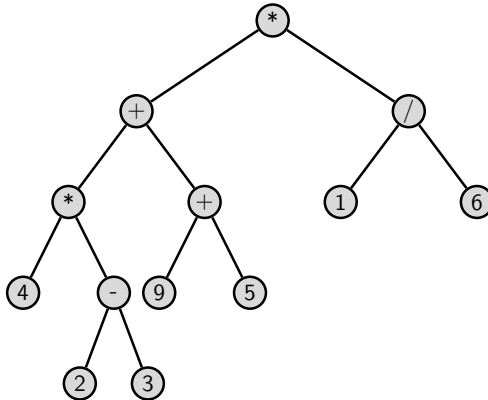


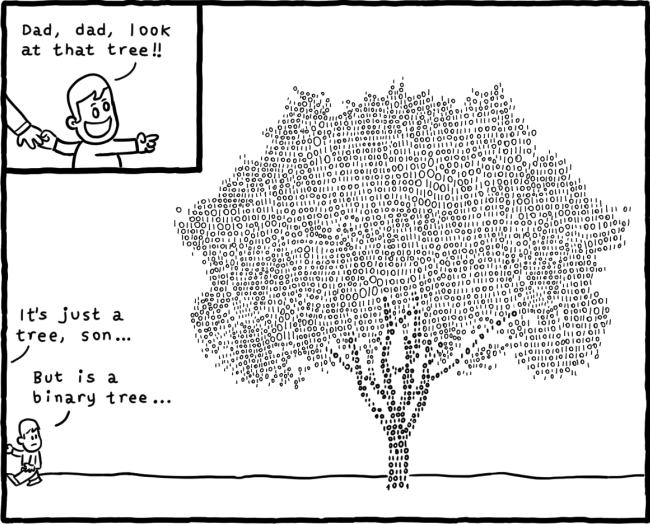
İfade ağaçları

$4 * (2 - 3) + (9 + 5) * (1/6)$ ifadesinin ifade ağacında gösterimini gerçekleştirin

İfade ağaçları

$4 * (2 - 3) + (9 + 5) * (1/6)$ ifadesinin ifade ağacında gösterimini gerçekleştirin





Daniel Stori {turnoff.us}