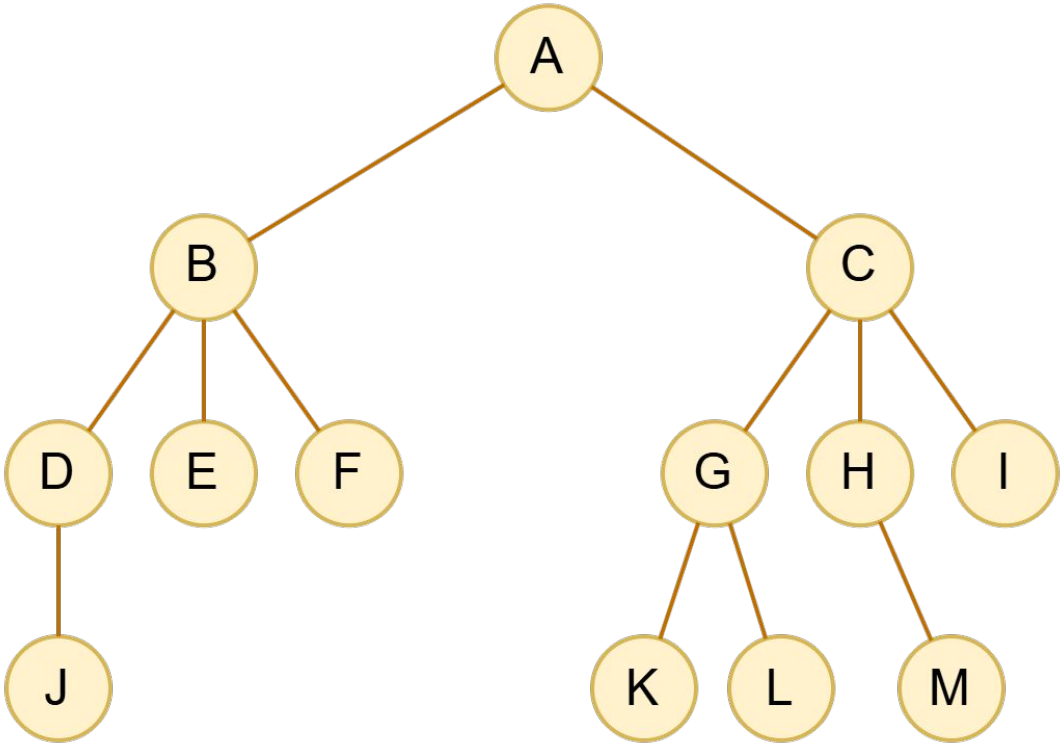
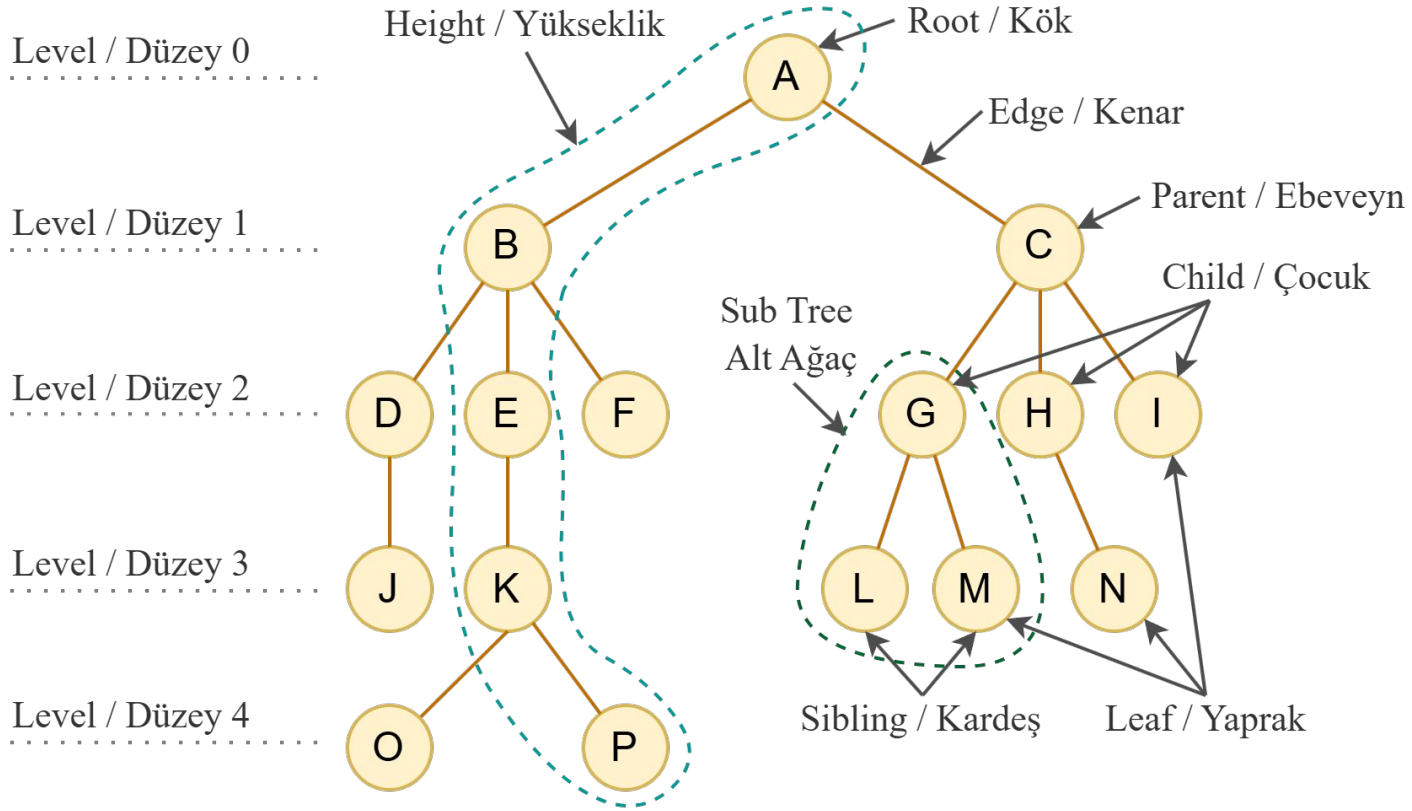
Ağaç (Tree) Veri Yapısı

Dr. Hakan Temiz

Ağaçlar



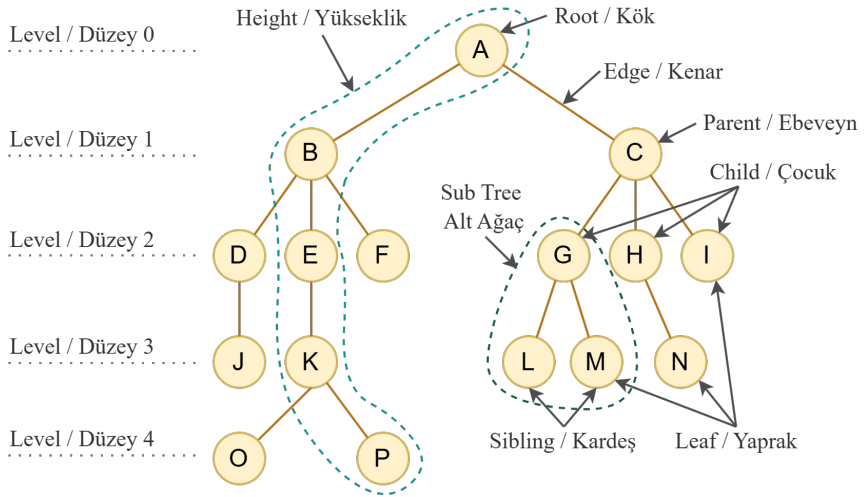
Ağaçlar - Tanımlar



Ağaçlar - Tanımlar

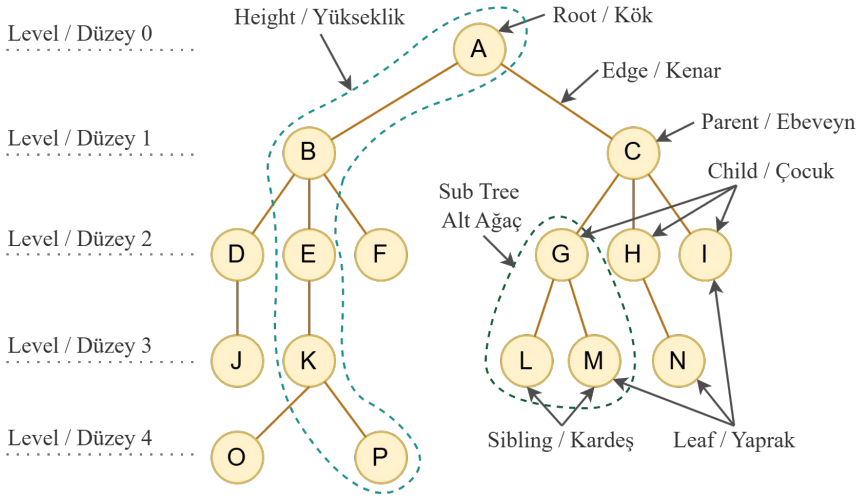
#2

- **Düğüm (Node):** Daire içine alınmış her harf bir düğümü temsil eder. Düğüm, verileri depolayan herhangi bir veri yapısıdır.
- **Kök (Root):** Ağaçtaki tüm düğümlerin bağlandığı ilk düğüme kök denir. Her ağaçta daima benzersiz bir kök düğüm vardır. Örnekte A kök düğümüdür.
- **Alt Ağaç (Sub Tree):** Ana ağacın belirli bir bölümüdür. Düğümleri başka bir ağacın soyundan gelen bir ağaçtır. G, L, M düğümleri alt ağaçtır.
- **Kenar (Edge):** Herhangi iki düğüm arasındaki bağlantıya kenar denir. Ağaçtaki toplam kenar sayısı, ağaçtaki toplam düğüm sayısından en fazla bir eksiktir.



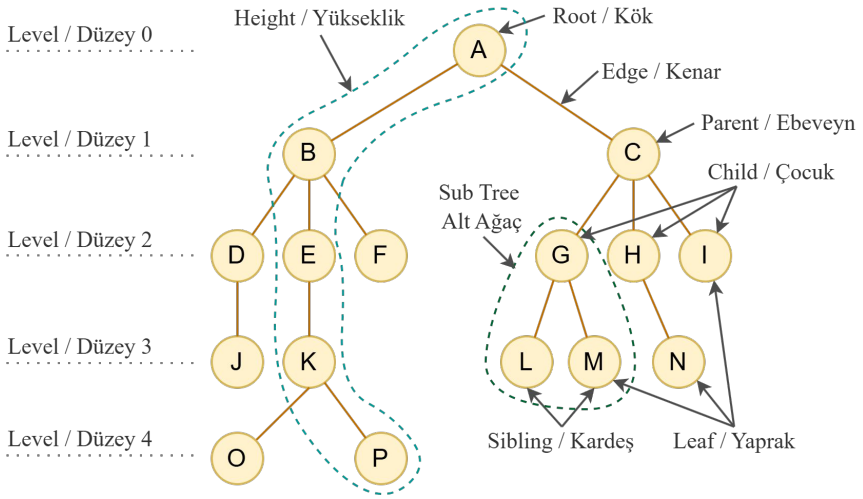
Ağaçlar - Tanımlar

- **Ebeveyn / Üst (Parent):** Bir alt ağacın en üst düğümü o alt ağacın ebeveynidir/üstüdür. Örneğin, C düğümü G, H ve I düğümlerinin ve ayrıca G düğümü L ve M düğümlerinin ebeveynidir.
- **Ata (Ancestor) ve Torun (Descendant):** Bir düğüm, kendisine bağlı tüm alt düğümlerin atasıdır. Bu düğüme bağlı tüm alt düğümler bu düğümün torunudur. Bir düğümden itibaren ağacın köküne uzanan yol üzerindeki tüm düğümler o düğümün atasıdır. Kök düğüm diğer tüm düğümlerin atası ve onlar da torunudur.
- **Çocuk (Child):** Ebeveyne bağlı alt düğümler çocuk düğümdür. B ve C düğümleri A düğümünün çocukları iken; G, H ve I düğümleri C düğümünün çocuklarıdır.



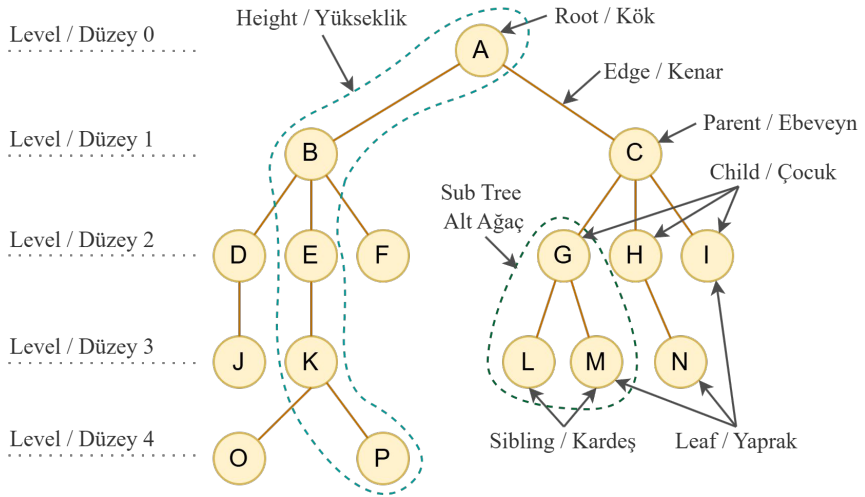
Ağaçlar - Tanımlar

- **Kardeş (Sibling):** Aynı ebeveyne sahip tüm düğümler kardeştir. Örneğin, B ve C düğümleri kardeştir ve benzer şekilde D, E ve F düğümleri de kardeştir.
- **Yaprak (Leaf):** Çocuğu olmayan düğüme yaprak denir. Yaprak ağacın son düğümüdür ve derecesi her zaman sıfır (0)'dır. F, I, J, L, M, N, O ve P yaprak düğümdür.
- **Dış (external) ve İç (internal) Düğüm:** Yaprak düğümlerin hepsi dış düğüm; geriye kalanlar iç düğüm olarak anılır.
- **Derece (Degree):** Bir düğümün toplam çocuk sayısına o düğümün derecesi denir. Yalnızca bir düğümden oluşan bir ağacın derecesi 0'dır. A düğümünün derecesi 2; B düğümünün derecesi 3 ve H düğümünün derecesi 1'dir.



Ağaçlar - Tanımlar

- **Seviye / Düzey (Level):** Ağacın kök düğümü seviye 0 olarak kabul edilir. Kök düğümün çocukları seviye 1'dir ve seviye 1'deki düğümlerin çocukları seviye 2 olarak kabul edilir. Örnekte, O ve P 'nin seviyesi 4'dür.
- **Yükseklik (Height):** Ağacın en uzun yolundaki düğümlerin toplam sayısı ağacın yüksekliğidir. Örneğin, önceki örnek ağaçta, ağacın yüksekliği 5'dir. Çünkü A-B-E-K-O veya A-B-E-K-P yolları en uzun yollardır ve ikisi de toplam 5 düğüme sahiptir.
- **Derinlik (Depth):** Bir düğümün derinliği, ağacın kökünden o düğüme kadar olan kenar sayısıdır. Örnekte, M düğümünün derinliği 3'dür.

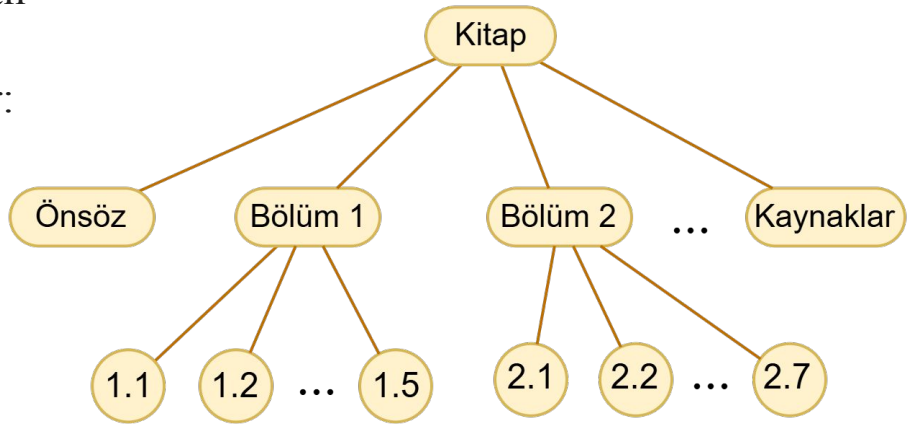


Ağaçlar - Türler

- Sıralı Ağaçlar
- Çarpık (Dejenere) Ağaçlar
- İkili Ağaçlar (Binary Trees)
    - Tam İkili Ağaç (Full Binary Tree)
    - Tam Dolu İkili Ağaç (Complete Binary Tree)
    - Mükemmel İkili Ağaç (Perfect Binary Tree)
    - Dengeli İkili Ağaç
    - AVL Ağacı
    - Kırmızı Siyah Ağaç
    - Kümeleme (Heap) Ağacı

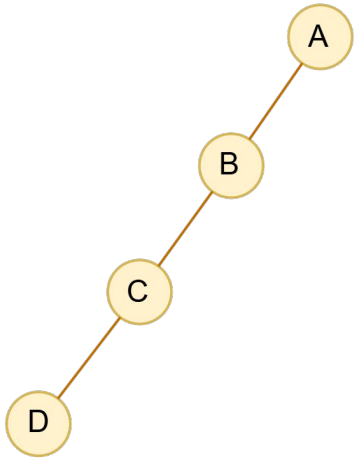
Ağaçlar - Türler - Sıralı Ağaçlar

- Sıralı ağaçlarda, her düğümün çocukları arasında anlamlı bir doğrusal sıra vardır. Örneğin, çocuklar birinci, ikinci, üçüncü vb. veya tam tersi olarak sıralı olabilir.
- Sıralama ağaçlarına, aile soy ağacı örnek verilebilir: aile soy ağacında, çocuklar, doğumlarına göre bir sıralamaya konulabilir.
- Diğer bir örnek, bir kitabın bölüm yapısının ağaç şeklinde temsilidir. Bölümler, Önsöz, Bölüm 1, Bölüm 2, ..., şeklinde devam eder.
- Bölümler de kendi içlerinde, örneğin Bölüm 1, Bölüm 1.1, Bölüm 1.2, ..., şeklinde alt bölümlerden meydana gelir. İşte, bu yapı sıralı bir ağaç yapısına bir örnektir.

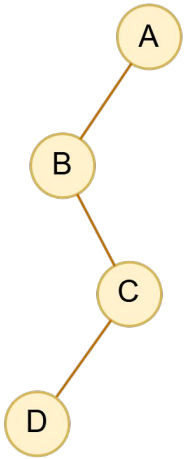


Ağaçlar - Türler - Çarpık (Dejenere) Ağaçlar

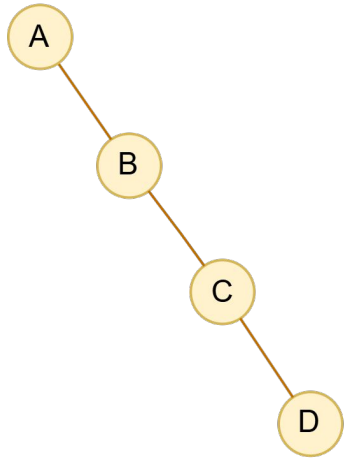
- En sondaki yaprak düğümler hariç, tüm düğümleri sadece ve sadece 1 çocuk düğüme sahip olan ağaçlara **çarpık (dejenere) ağaç** denir.
- Eğik ağaç sadece sol çocuklara sahip ise **sola çarpık ağaç**;
- Sağ çocuklara sahip ise **sağa çarpık ağaç** adı verilir.



Sola Çarpık Ağaç



Çarpık Ağaç



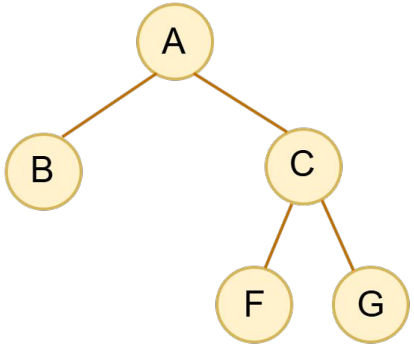
Sağa Çarpık Ağaç

Ağaçlar - Türler - İkili Ağaçlar

- Bir ağacın her düğümü ya hiç çocuğa sahip değilse veya en fazla 2 düğüme sahipse, bu ağaca ikili ağaç denir.
- Bu bağlamda, boş bir ağaç da geçerli bir ikili ağaçtır.
- İkili ağacı, bir kök ve kökün sol ve sağ alt ağaçları olarak adlandırılan iki ayrık ikili ağaçtan oluşan bir ağaç olarak değerlendirilebilir.
- İkili ağaçlarda, genellikle, sol çocuk sıralamada sağ çocuktan önce gelir.

Ağaçlar - Türler - Tam İkili Ağaç (Full Binary Tree)

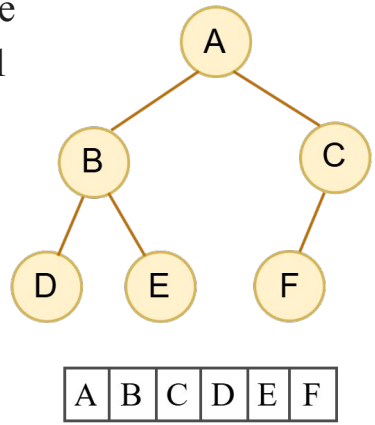
- Her düğümün tam olarak iki çocuğu varsa veya hiç çocuğu yoksa, bu ağaca tam ikili ağaç denir. Dolu ikili ağaçta yalnızca bir alt düğümü olan hiçbir düğüm bulunmaz.
- Aşağıdaki örnekte, seviye seviye düğümler yazıldığında A B C - - F G dizisi elde edilir. Dizideki boşluklar, B 'nin alt düğümlerinin olmayışından kaynaklanır.
- Tam ikili ağaçta, n adet iç düğüm varsa yaprak sayısı n + 1 olur ve buna tam ikili ağaç teoremi denir.



A	B	C	-	-	F	G

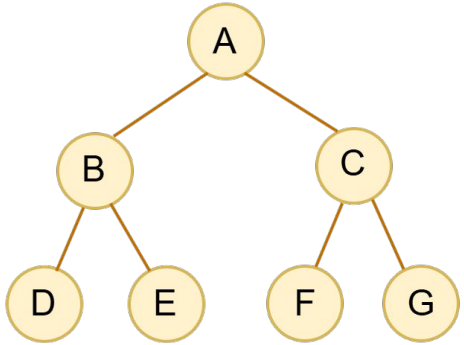
Ağaçlar - Türler - Tam Dolu İkili Ağaç (Complete Binary Tree)

- Yüksekliği h olan bir tam ikili ağaçta, düğümlere kökten başlayarak seviye seviye numaralandırma yapıldığında (kök düğümün 1) 1'den ağaçtaki düğüm sayısına kadar tam dolu bir dizi elde edilir. Gezinirken None referanslar (olmayan düğümler) için de numaralandırma yapılır. Tam dolu ikili bir ağacın tüm yaprak düğümleri h veya h – 1 yüksekliğinde olur ve dizide hiçbir eksik sayı yoktur.
- Tam dolu ikili ağacın bazı özellikleri şunlardır:
  - Tüm yaprakları aynı derinlikte olan tam dolu ikili ağaca düzgün ikili ağaç denir.
  - Son seviye dışındaki tüm düzeyler tamamen doludur.
  - d derinliğindeki düğüm sayısı 2^d 'dir.
  - n düğümlü bir tam dolu ikili ağacın yüksekliği $\log(n+1)$ 'dir.
- Yandaki örnekte, seviye seviye düğümler yazıldığında herhangi bir boşluk olmadan A B C D E F G dizisi elde edilir. Tam dolu ikili ağaçlar tam dolu bir dizi meydana gelir.



Ağaçlar - Türler - Mükemmel İkili Ağaç (Perfect Binary Tree)

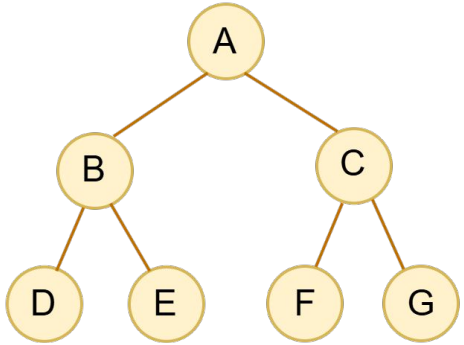
- Her düğümün tam olarak iki çocuğu varsa ve tüm yaprak düğümler aynı derinlikte ise bu ağaca mükemmel ikili ağaç denir.
- Mükemmel ikili ağacın tüm yaprak düğümleri maksimum derinliktedir.
- Yaprak olmayan düğümlerin daima iki çocuğa sahip olmaları ağacın simetrik olmasını sağlar.
- Bu ağaçlar, dizi veri yapısı kullanılarak temsil edilebilir. Dizi ile temsil edilmeleri durumunda, i dizinindeki bir düğümün sol çocuğu 2i + 1 ve sağ çocuğu 2i + 2 indeksinde bulunur.
- Bu basit erişim hesabı, ağacın düğümlerine erişmeyi ve dolayısıyla ağacı dolaşmayı kolaylaştırır.



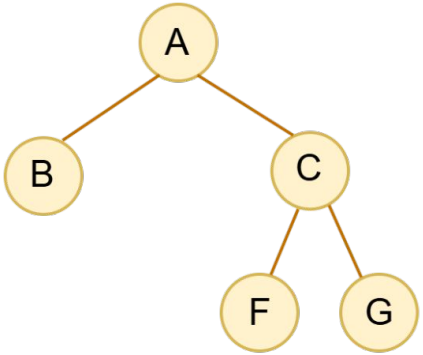
Ağaçlar - Türler - Mükemmel İkili Ağaç (Perfect Binary Tree)

Mükemmel ikili ağacın özellikleri şunlardır:

- Yaprak düğümlerinin derecesi 0'dır.
- Yüksekliği h olan bir mükemmel ağacın toplam düğüm sayısı, ağacın her düğümü dolu olduğundan, $2^0 + 2^1 + \ldots + 2^h = 2^{h+1} - 1$ olur.
- Bir düğümün ortalama derinliği $\Theta(\ln(n))$ olur.
- Yaprak düğümü sayısı = yaprak olmayan düğüm sayısı + 1'dir.
- Ağacın yüksekliği: N sayıda düğüme sahip mükemmel bir ikili ağacın yüksekliği $\log(N+1) - 1 = \Theta(\ln(n))$ olur.

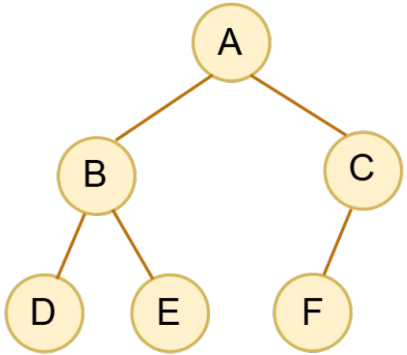


Ağaçlar - Türler - Tam, Tam Dolu ve Mükemmel İkili Ağaç



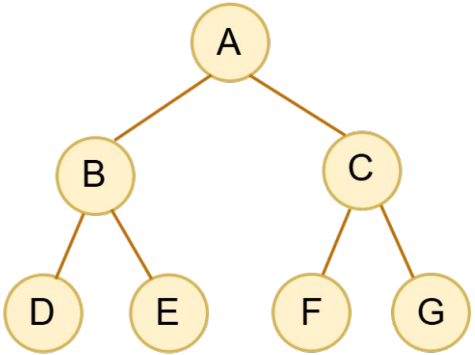
A	B	C	-	-	F	G

Tam İkili Ağaç



A	B	C	D	E	F

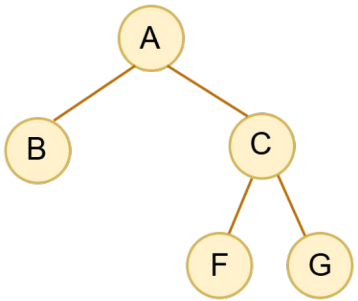
Tam Dolu İkili Ağaç



Mükemmel İkili Ağaç

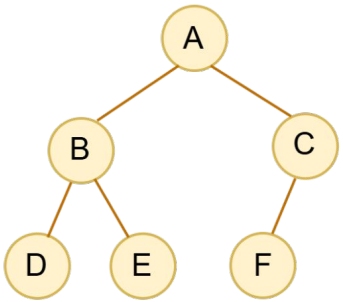
Ağaçlar - Türler - Tam, Tam Dolu ve Mükemmel İkili Ağaç #1

- Mükemmel ikili ağaçta tüm yapraklar aynı derinliktedir; son seviye hariç hiç yaprak düğüm yoktur.
- Tam ikili ağaç son seviyeden önce de boş bir yaprak (B) düğüme sahiptir.
- Tam dolu ikili ağacın tüm yaprak düğümleri en alt seviyede olmakla beraber, son seviye hariç, tüm düğümler iki çocuğa sahip değildir (örnekte C).



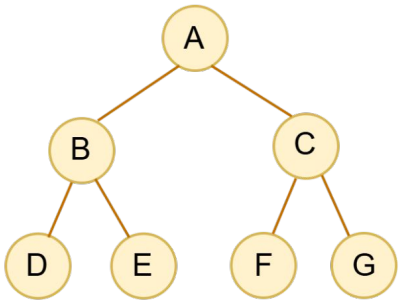
A	B	C	-	-	F	G

Tam İkili Ağaç



A	B	C	D	E	F

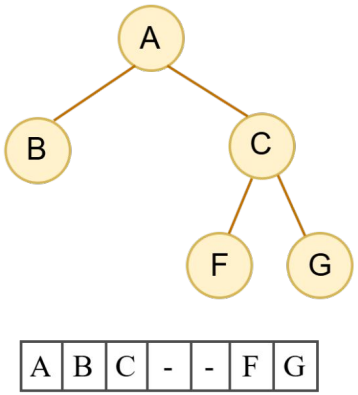
Tam Dolu İkili Ağaç



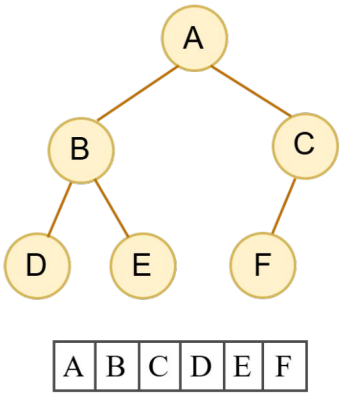
Mükemmel İkili Ağaç

Ağaçlar - Türler - Tam, Tam Dolu ve Mükemmel İkili Ağaç #2

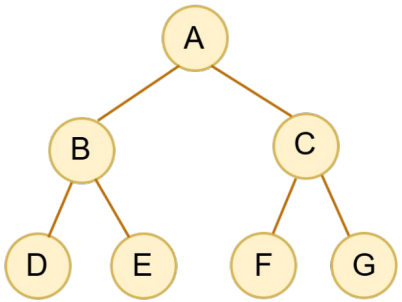
- Seviye seviye tüm düğümler sıralandığında, tam ikili ağaç A B C - - F G dizisini oluştururken; tam dolu ikili ağaç A B C D E F G dizisini oluşturur.
- Tam ikili ağaç dizisinde ara indislerde boş (None - Null) değerler bulunabilir. Bu durum tam dolu ikili ağaçlar için söz konusu değildir.



Tam İkili Ağaç



Tam Dolu İkili Ağaç

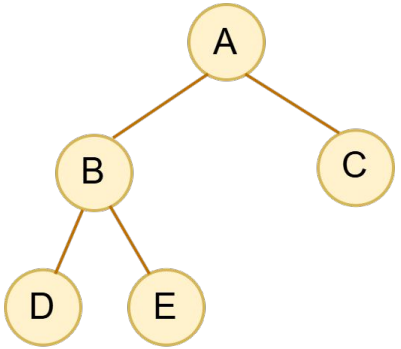


Mükemmel İkili Ağaç

Ağaçlar - Türler - Ağaç Türünü Belirleme

Örnek #1

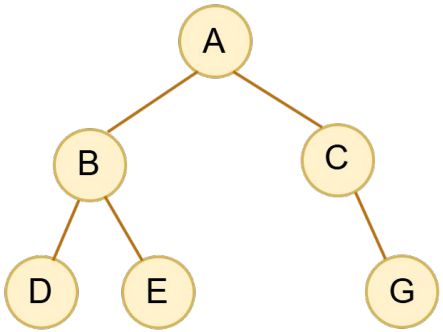
- Bu ağaçta, derecesi 1 olan hiçbir düğüm bulunmamaktadır.
- Her düğüm kesinlikle 2 veya 0 çocuğa sahiptir.
- Sıralama yapıldığında A B C D E dizi oluşur.
- Bu nedenle hem tam ikili ağaç hem de tam dolu ikili ağaçtır.
- Son seviye hariç, tüm düğümleri iki elemana sahip olmadığından (C 'nin yok) mükemmel ikili ağaç değildir.



Ağaçlar - Türler - Ağaç Türünü Belirleme

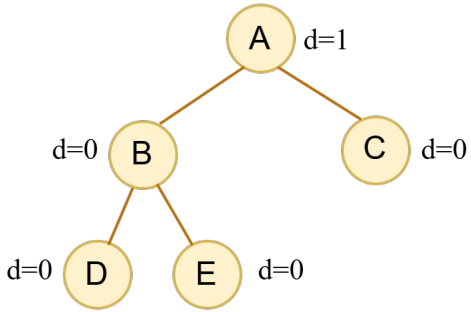
Örnek #2

- Verilen ağaçta, C düğümünün derecesi 1’dir.
- Dolayısıyla tam ikili ağaç değildir.
- Bir dizi üzerinde sıralanırsa A B C D E - G sıralaması oluşur.
- Arada boşluk oluştuğundan tam dolu ikili bir ağaç da değildir.
- Tüm yaprak düğümler son düzeyde olmasına rağmen, son seviye hariç, her düğümün iki çocuğu olmadığından, mükemmel ikili ağaç da değildir.

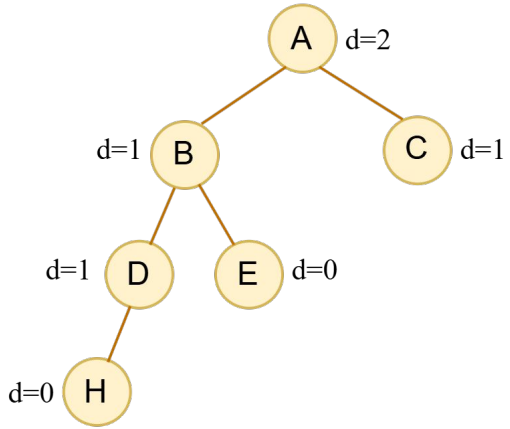


Ağaçlar - Türler - Dengeli İkili Ağaç #1

- n düğümlü bir ağacın yüksekliği O(logn) ise, ağaç dengelidir denir.
- Bu ağaç, gelişmesini düğümlerine homojen biçimde yansıtır.
- Dengeli ağaçta her düğümünün sol ve sağ alt ağaçları yükseklikleri arasındaki fark 0 veya 1'dir.
- Boş bir ağaç aynı zamanda dengeli bir ağaçtır.



Dengeli İkili Ağaç

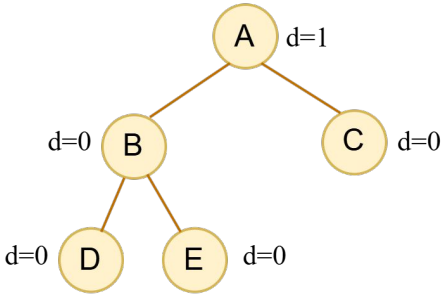


Dengesiz İkili Ağaç

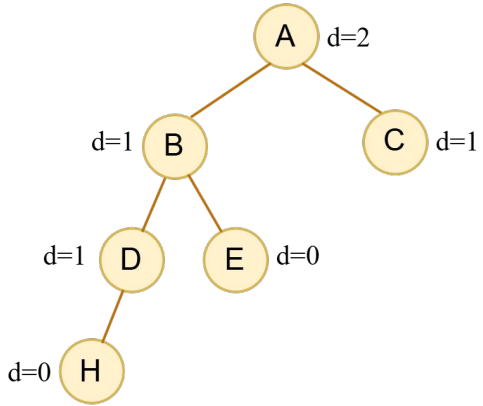
derinlik (d) = | sol çocuğun yüksekliği - sağ çocuğun yüksekliği |

Ağaçlar - Türler - Dengeli İkili Ağaç #2

- Dengeli ikili arama ağaçları arama, ekleme ve silme için O(log n) süre sağladıkları için performans açısından elverişlidir.
- Bir İkili ağacın dengeli olması için şu koşulları sağlamalıdır:
  - Herhangi bir düğümdeki sol ve sağ alt ağaçların yüksekliklerinin mutlak farkı 1'den küçük olmalıdır.
  - Her düğüm için sol ve sağ alt ağacı dengeli bir ikili ağaç olmalıdır.



Dengeli İkili Ağaç

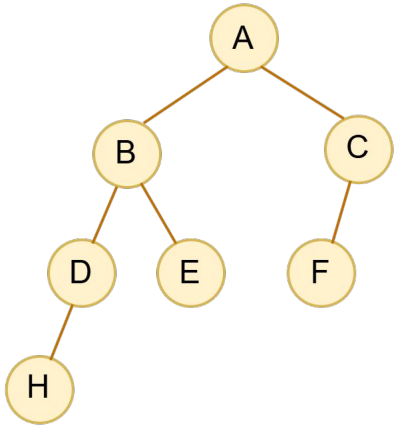


Dengesiz İkili Ağaç

derinlik (d) = | sol çocuğun yüksekliği - sağ çocuğun yüksekliği |

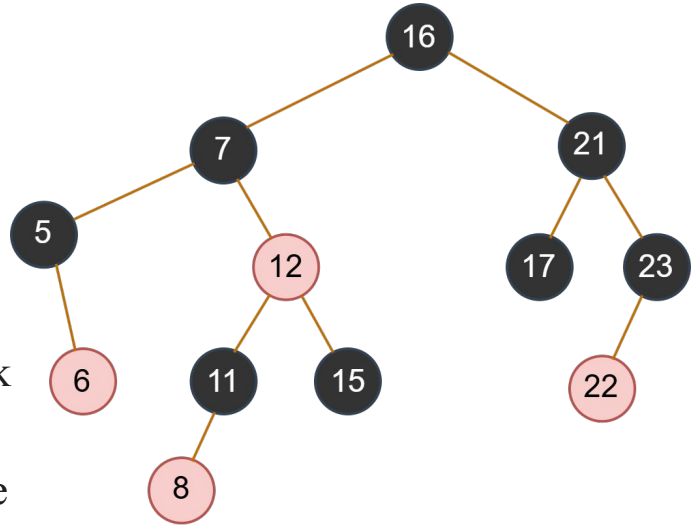
Ağaçlar - Türler - AVL Ağacı

- AVL ağacı, sol ve sağ alt ağaçların yükseklikleri arasındaki farkın tüm düğümler için 1'den fazla olamayan ve bu şekilde kendini (yüksekliği) dengeleyen bir ikili arama ağacıdır.
- İsmini mucitlerinin (Adel'son-Vel'skii ve Landis) isimlerinin baş harflerinden almıştır.
- AVL'nin amacı yüksekliği mümkün olduğunca düşük tutmaktır.
- Yükseklik-denge özelliği gereği, bir AVL ağacının alt ağacı da bir AVL'dir.
- Yandaki ağaç AVL'dir çünkü her düğüm için sol ve sağ alt ağaçların yükseklikleri arasındaki fark 1'den küçük veya eşittir.
- Bir AVL ağaçta ekleme, silme ve arama gibi işlemlerin karmaşıklığı $O(\log_2 n)$'dir.



Ağaçlar - Türler - Kırmızı Siyah Ağaç

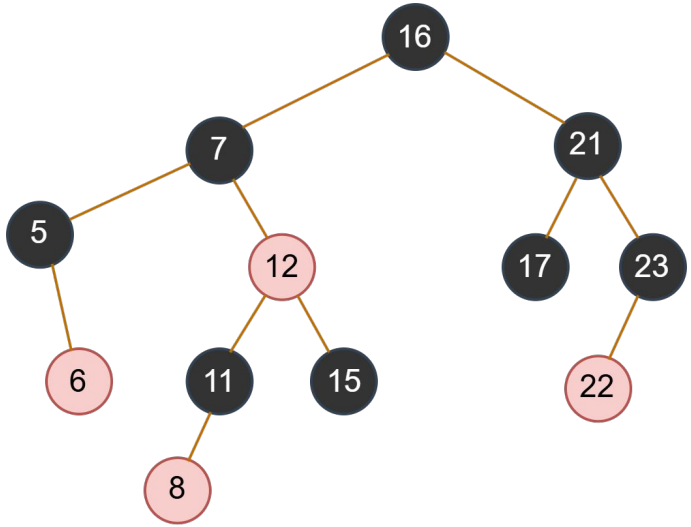
- Düğümlerinin kırmızı ve siyah renk vb. ek bir özniteliğe sahip olduğu ve kendi kendini dengeleyen bir ikili arama ağacıdır.
- Düğümler genellikle siyah ve kırmızı olarak (denge gözetilerek) nitelendirilir.
- Düğümlerin renklerle kısıtlanması, kökten herhangi bir yaprağa kadar olan yolun, diğer düğümlerin yol uzunluğunun iki katından fazla olmamasını ve böylece ağacın yaklaşık olarak dengeli kalmasını sağlar.
- Bu ağaçlar, eklemeler ve silmeler sırasında dengeyi koruyarak verimli veri alma ve işleme sağlar.
- AVL ağaçlarındaki, öğe ekleme, çıkarma işlemleri neticesinde ağacı yeniden yapılandırma vb. dezavantajları yoktur.
- Ekleme, silme ve arama gibi işlemler için logaritmik zaman karmaşıklığına sahiptir.



Ağaçlar - Türler - Kırmızı Siyah Ağaç #2

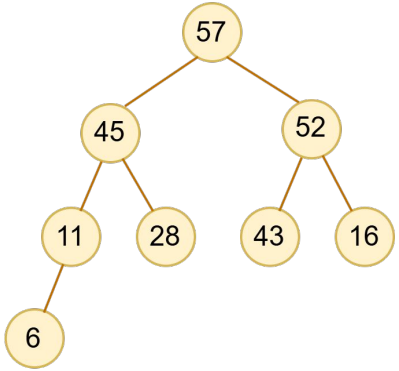
Kırmızı siyah ağaçlar şu özelliklere sahiptir:

- Tüm düğümler kırmızı ya da siyahtır.
- Kök düğüm siyahtır.
- Tüm yapraklar (None/Boş olanlar) siyah kabul edilir.
- Kırmızı bir düğümün çocukları (varsa) siyahtır.
- Sıfır veya bir çocuğu olan tüm düğümler, siyah atalarının sayısı olarak tanımlanan aynı siyah derinliğe sahiptir.

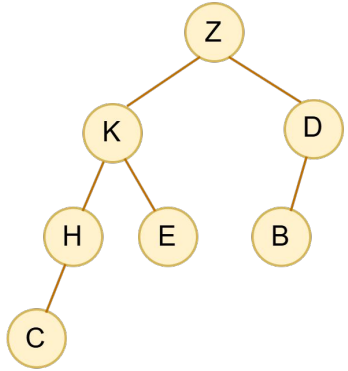


Ağaçlar - Türler - Kümeleme (Heap) Ağacı #1

- Kümeleme ağacı, büyük değerlerin üst seviyeye; küçük olanların da alt seviyeye yerleştirilmesi prensibine göre inşa edilen ağaç türüdür.
- Bir düğümün değeri, çocuklarının değerlerinden daima büyük veya eşittir.
- Bu bağlamda, kök seviyesinde en büyük değer bulunurken, en alt seviye olan h seviyesinde en küçük değerler bulunur.
- Yerleşim, küçükten büyüğe de olabilmekle beraber aksi belirtilmedikçe küçükten büyüğe kabul edilir.
- Sıralama sayısal değerlere göre olabileceği gibi, alfanümerik karakterlere göre de yapılabilir.



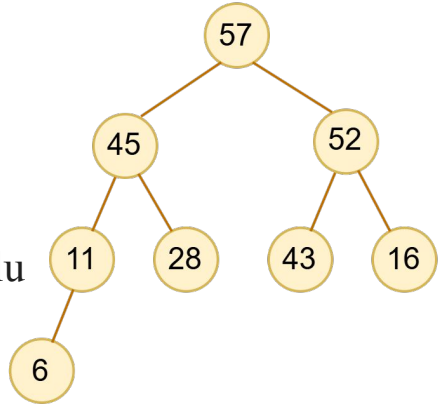
Sayısal Kümeleme Ağacı



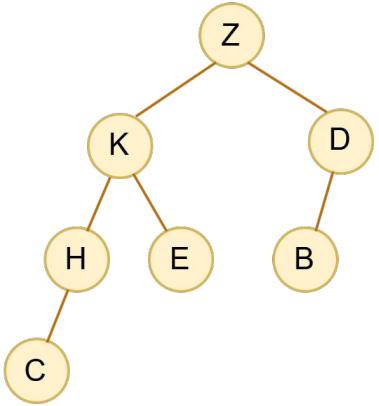
Alfabetik Kümeleme Ağacı

Ağaçlar - Türler - Kümeleme (Heap) Ağacı #2

- Bir ağacın kümeleme ağacı olması için,
  - Herhangi iki yaprak düğümün derinlikleri farkı en fazla 1 olmalıdır.
  - Tüm düğümler için, düğümün değeri çocuklarının değerlerinden büyük olmalıdır.
  - En alt düzey hariç, tüm düzeyler düğümlerle dolu olmalıdır.
  - Tüm düğümler için, en alt seviyedeki boşluklar daima ağacın sağ tarafında kalmalıdır.
- Kümeleme ağaçları uygulamalarına, öncelik kuyrukları (priority queues) ve oldukça verimli sıralama sağlayan kümeleme sıralaması (heap sort) gibi algoritmaların gerçekleştirimi örnek verilebilir.



Sayısal Kümeleme Ağacı



Alfabetik Kümeleme Ağacı

Ağaçlar - Türler - İkili Ağaçlar - Gezinme

Ağaçlarda gezinme, ağaçtaki tüm düğümleri dolaşma eylemidir. Doğrusal olmayan bir veri yapısı olduğundan, ağacı gezmenin çok farklı yolları bulunabilir. Gezinmenin iki ana türü vardır:

- **Genişlik Öncelikli Gezinme (Breath First Traversal)**

  Genişlik öncelikli gezinme yalnızca tek bir şekilde yapılır: seviye seviye gezinme. Bu gezinmede, en üst seviyeden başlanarak, bir seviyedeki düğümler soldan sağa doğru gezildikten sonra bir alt seviyey geçilir ve bu süreç alt seviyelerde de aynı şekilde gerçekleştirilir.

- **Derinlik Öncelikli Gezinme (Depth First Traversal)**

  Derinlik öncelikli gezinmede, daima öncelikle düğümlerin alt düğümleri sonrasında aynı seviyedeki düğümler ziyaret edilir. Bu gezinmenin üç türü vardır:

  - **Kök Önce (Pre-order):** Bu gezinmede, önce ebeveyn (Kök), sonra Sol ve son olarak Sağ çocuk ziyaret edilir.
  - **Kök Ortada (InOrder) Gezinme:** Bu gezinmede, önce Sol çocuk, sonra ebeveyn (Kök) sonra da Sağ çocuk ziyaret edilir.
  - **Kök Sonra (PostOrder) Gezinme:** Bu gezinmede, önce sol çocuk, sonra sağ çocuk ve sonra da ebeveyn (kök) ziyaret edilir..

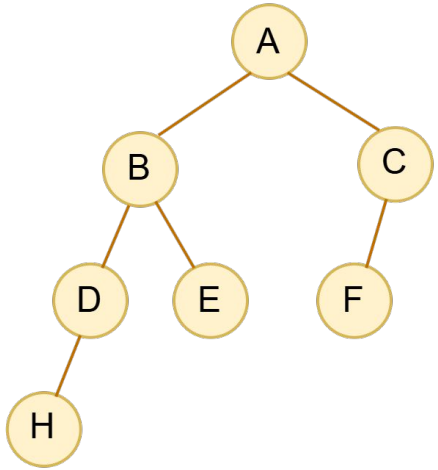
Ağaçlar - Türler - İkili Ağaçlar - Gezinme - Örnek #1

- **Genişlik Öncelikli Gezinme:** Seviye seviye gezilir.
  A-B-C-D-E-F-H
- **Kök Önce (Pre-order):** Önce ebeveyn (Kök), sonra Sol ve son olarak Sağ çocuk ziyaret edilir.

  A-B-D-H-E-C-F
- **Kök Ortada (InOrder) Gezinme:** Önce Sol çocuk, sonra ebeveyn (Kök) sonra da Sağ çocuk ziyaret edilir.

  H-D-B-E-A-F-C
- **Kök Sonra (PostOrder) Gezinme:** Bu gezinmede, önce sol çocuk, sonra sağ çocuk ve sonra da ebeveyn (kök) ziyaret edilir.

  H-D-E-B-F-C-A



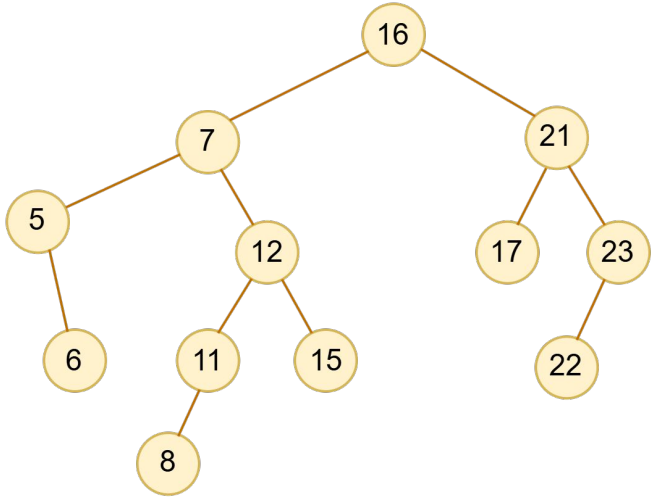
Ağaçlar - Türler - İkili Ağaçlar - Gezinme - Örnek #2

- **Genişlik Öncelikli Gezinme:** Seviye seviye gezilir.
  16-7-21-5-12-17-23-6-11-15-22-8
- **Kök Önce (Pre-order):** Önce ebeveyn (Kök), sonra Sol ve son olarak Sağ çocuk ziyaret edilir.

  16-7-5-6-12-11-8-15-21-17-23-22
- **Kök Ortada (InOrder) Gezinme:** Önce Sol çocuk, sonra ebeveyn (Kök) sonra da Sağ çocuk ziyaret edilir.

  5-6-7-8-11-12-15-16-17-21-22-23
- **Kök Sonra (PostOrder) Gezinme:** Bu gezinmede, önce sol çocuk, sonra sağ çocuk ve sonra da ebeveyn (kök) ziyaret edilir.

  6-5-8-11-15-12-7-17-22-23-21-16



Ağaçlar - Türler - İkili Ağaçlar - Kök Önce Gezinme

```
class Dugum:
    def __init__(self, data):
        self.data = data
        self.sol = None
        self.sag = None

def kok_once_gez(kok):
    if kok is None:
        return

    # Bu düğümün verisi
    print(kok.data, end=' ')

     Sol alt ağacı Özyineli gez
    kok_once_gez(kok.sol)

    # Sağ alt ağacı Özyineli gez
    kok_once_gez(kok.sag)
```

```
kok= Dugum(1)
kok.sol = Dugum(2)
kok.sag = Dugum(3)
kok.sol.sol = Dugum(4)
kok.sol.sag = Dugum(5)
kok_once_gez(kok)
```

```
1 2 4 5 3
```

Ağaçlar - Türler - İkili Ağaçlar - Kök Ortada Gezinme

```
class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

def inorderTraversal(root):
    if root is None:
        return

    # Sol alt ağacı Özyineli gez
    inorderTraversal(root.left)

    # Bu düğümün verisi
    print(root.data, end=' ')

    # Sağ alt ağacı Özyineli gez
    inorderTraversal(root.right)
```

```
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
inorderTraversal(root)
```

```
4 2 5 1 3
```

Ağaçlar - Türler - İkili Ağaçlar - Kök Sonra Gezinme

```
class Dugum:
    def __init__(self, data):
        self.data = data
        self.sol = None
        self.sag = None

def kok_sonra_gez(kok):
    if kok is None:
        return

    # Sol alt ağacı Özyineli gez
    kok_sonra_gez(kok.sol)

    # Sağ alt ağacı Özyineli gez
    kok_sonra_gez(kok.sag)

    # Bu düğümün verisi
    print(kok.data, end=' ')
```

```
kok= Dugum(1)
kok.sol = Dugum(2)
kok.sag = Dugum(3)
kok.sol.sol = Dugum(4)
kok.sol.sag = Dugum(5)
kok_sonra_gez(kok)
```

```
4 5 2 3 1
```

Ağaçlar - Türler - İkili Ağaçlar - Genişlik Öncelikli Gezinme

```
from collections import deque

class Dugum:# Ağaç düğümü

    def __init__(self, data):
        self.data = data
        self.sol = None
        self.sag = None

# genişlik öncelikli gez
def genislik_oncelikli_gez(kok):
    if not kok:
        return

    kuyruk= deque([kok])
    while kuyruk:
        dugum= kuyruk.popleft()
        print(dugum.data, end=" ")
        if dugum.sol:
            kuyruk.append(dugum.sol)
        if dugum.sag:
            kuyruk.append(dugum.sag)

kok= Dugum(1)
kok.sol = Dugum(2)
kok.sag = Dugum(3)
kok.sol.sol = Dugum(4)
kok.sol.sag = Dugum(5)
kok.sag.sag = Dugum(6)
genislik_oncelikli_gez(kok)
```

```
1 2 3 4 5 6
```

Ağaçlar - İkili Arama Ağaçları (Binary Search Trees)

Gezinmeler Ne İşe Yarar?

- **Preorder (Kök Önce):**

  Ağacın kopyasını elde etme, Düğümleri sayma Yaprakları sayma, Matematiksel ifadeler için prefix gösterimini elde etme.

- **Postorder (Kök Sonra):**

  İkili ağacı silme, Fonksiyonel dil derleyicilerinde, Hesap makinesi programlarında, Postfix matematiksel ifadelerin oluşturulmasında.

- **Inorder (Kök Ortada):**

  İkili Arama Ağacında (Binary Search Tree) ağaçtaki bilgiyi sıralı halde yazdırmada.

Ağaçlar - İkili Arama Ağaçları (Binary Search Trees)

- İkili arama ağaçları, veri sıralama, arama ve alma için verimli bir yol sağlayan yapılardır.
- Bu ağaçlarda, sol düğümün değeri ebeveynin değerinden; ebeveynin değeri de sağ düğümün değerinden küçüktür.
- Tekrarlara izin verilmez.
- Bir ebeveynin solundaki tüm alt düğümlerin değerleri ebeveynin değerlerinden küçüktür.
- Bunun tam tersi ise sağ alt torunlar için geçerlidir.

Ağaçlar - İkili Arama Ağaçları (Binary Search Trees) - Arama

Arama

- Arama işlemi, kök düğüm ile başlar.
- Aranan değer kök düğümün değeri ise arama sona erer.
- Aranan, kök düğümden küçükse arama sol çocuk üzerinden; büyük ise, sağ düğümden devam eder.
- Bu karşılaştırma ve ilerleme prosedürü alt seviyelerde de aynı şekilde devam eder.
- Arama işlemi her seferinde bir ebeveynin bir yanından devam ettiği için oldukça hızlı arama imkanı sağlar. Örneğin 1 milyon veri arasından arananın bulunması en fazla $\log_2(10^6) \approx 20$ yineleme gerektirir.
- Bir bağlantılı listede ise arama 1 milyon yineleme bile gerektirebilir.

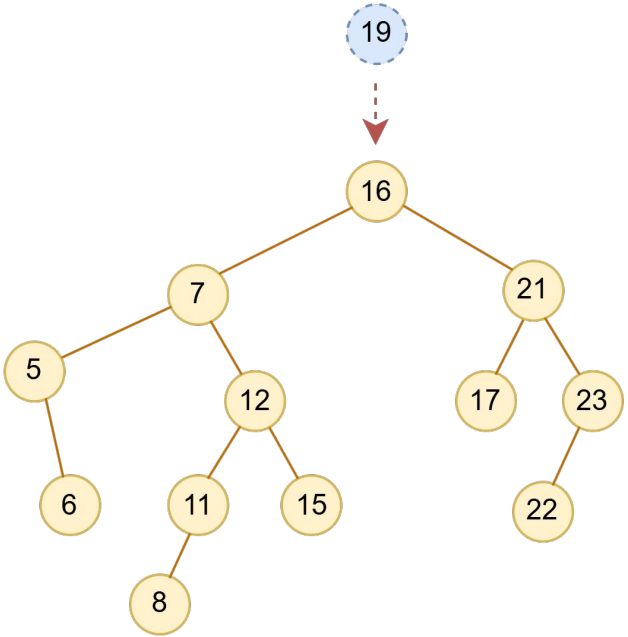
Ağaçlar - İkili Arama Ağaçları (Binary Search Trees) - Ekleme

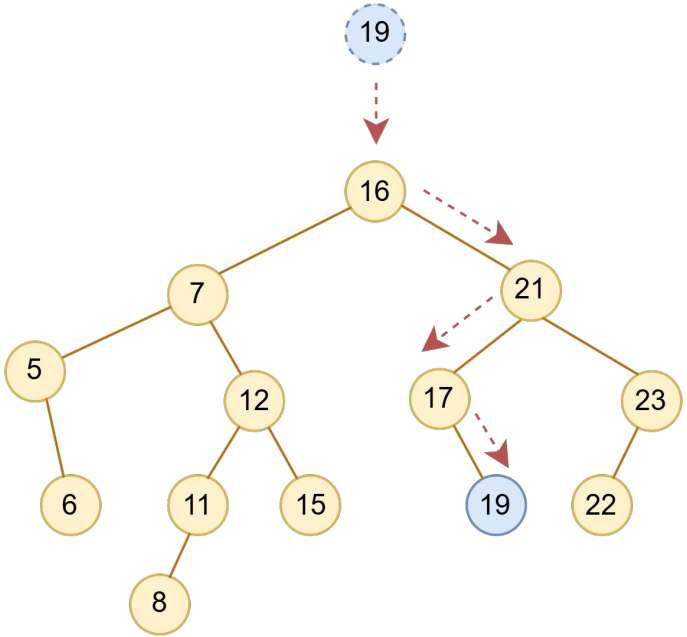
Ekleme

- İkili ağaçlarda ekleme işlemi, arama işlemine benzer.
- Yeni düğümün geleceği konum, arama işlemindeki gibi kök düğümden başlanarak ve her seferinde eklenen düğümün değeri ile mevcut düğümlerin değerleri kıyaslanarak alt düğümlere doğru devam edilir.
- Gidilebilecek alt düğüm kalmadığında, yeni düğümün konumu belirlenmiş olur.
- Yeni düğümün değeri ağaçta var ise ekleme işlemi birşey yapılmadan sona erer.

Ağaçlar - İkili Arama Ağaçları - Ekleme - Örnek

19, kök düğümün değeri 16’dan büyük olduğu için kök düğümün sağ çocuğu ile devam edilir. 21’den küçük olduğu için 17 düğümüne gidilir. 17 düğümü bir yaprak olduğundan ve 19 değeri 17’den büyük olduğundan, 19 düğümü 17’nin sağ çocuğu yapılır ve ekleme işlemi sona erer.



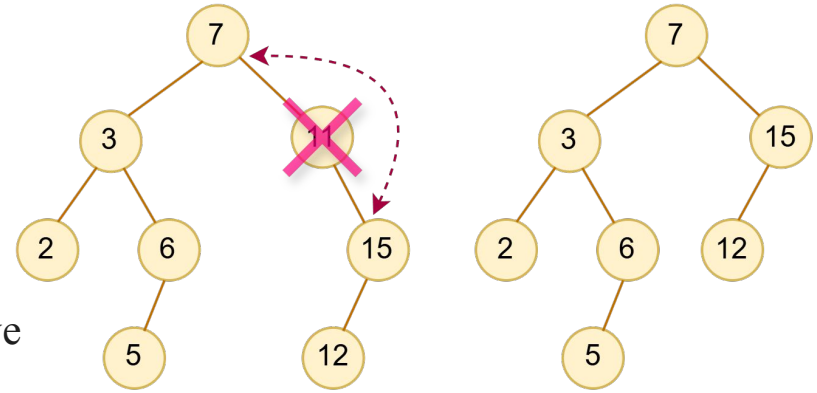


Ağaçlar - İkili Arama Ağaçları - Silme

Silme

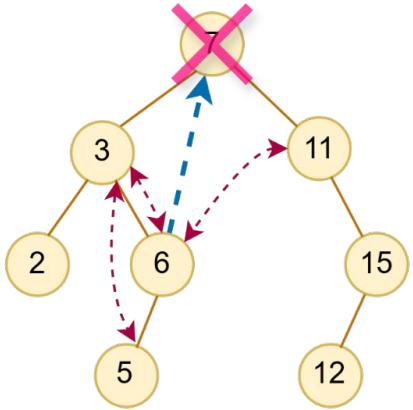
Silme işlemi, arama ve ekleme işlemlerinden biraz daha karmaşıktır. Silinecek düğümün varlığı/yokluğu ve çocuk sayısına göre, silme işlemi farklı müdahaleler gerektirir. Bunlar şunlardır:

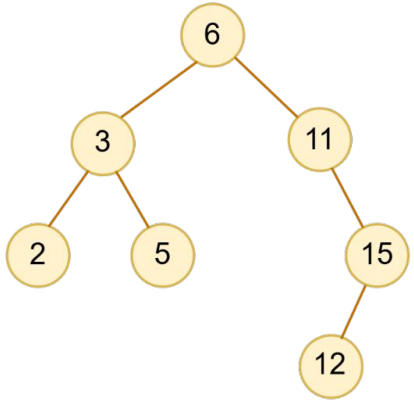
- Silinecek düğüm ağaçta yoktur. Birşey yapmaya gerek yoktur.
- Silinecek düğüm bir yapraktır. Doğrudan silinir. İlave bir işleme gerek yoktur.
- Silinecek düğüm yalnızca bir çocuğa sahiptir. Bu durumda, çocuk düğüm, silinen düğümün ebeveynine bağlanır. Yani, silinen düğümün yerini alır.



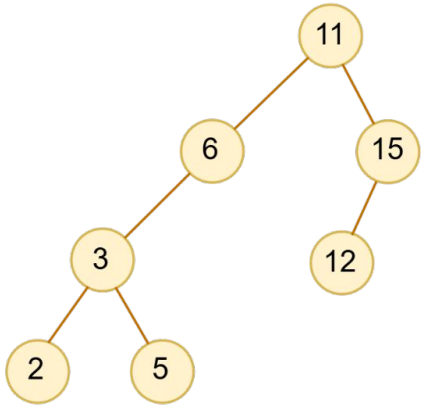
Ağaçlar - İkili Arama Ağaçları - Silme

Silinecek düğümün iki çocuğu vardır. Bu durumda, düğüm silinince iki alt ağaç meydana gelir. Silinecek düğümün solundaki alt ağacın en büyük düğümü silinenin yerini alır ve sağ alt ağacın en küçük düğümüne bağlandıktan sonra düğüm silinir. Bağlama neticesinde ağın dengeli olması gözetilir. Ağın bir tarafa doğru fazla gitmesi neticesinde dengesizleşmesini önlemek amacıyla, sol ağacın en büyük düğümü, sağ alt ağacın en küçük düğümünün soluna da bağlanabilir.





Sağ ağaç, Sol ağacın en büyük düğümünün sağına bağlı (dengeli)

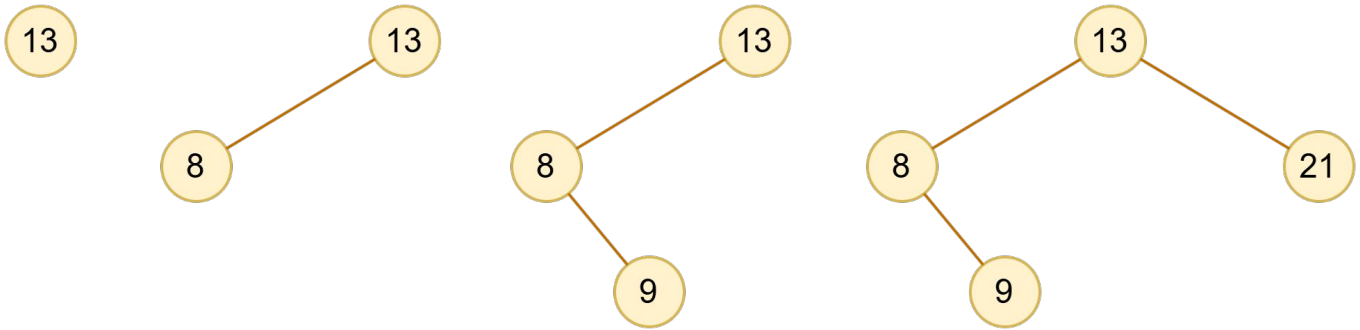


Sol ağacın en büyük düğümü, Sağ ağacın en küçük düğümünün soluna bağlı (dengesiz)

Ağaçlar - İkili Ağaçlar - Ağaç Oluşturma #1

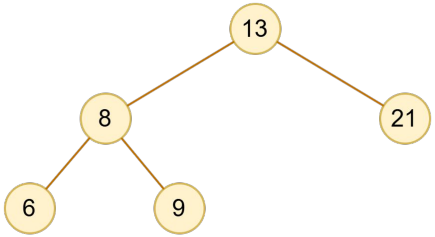
İkili bir ağacı oluşturmak mümkündür. Ağacı inşa ederken, bir sonraki elemanın değerine bakılarak, mevcut düğüm değerinden büyükse sola; aksi halde sağa doğru ilerlenerek, bir yaprağa ulaşıncaya kadar devam edilir.

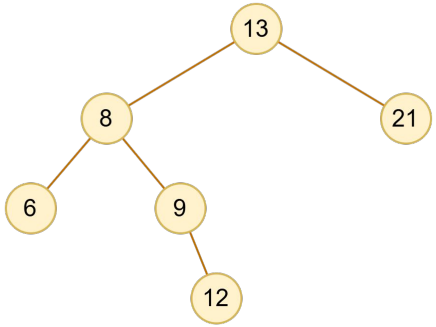
Şu değerlerden bir ağaç inşa edelim: 13, 8, 9, 21, 6, 12, 7, 19, 44, 47, 33

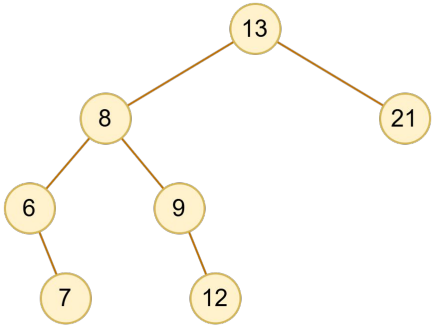


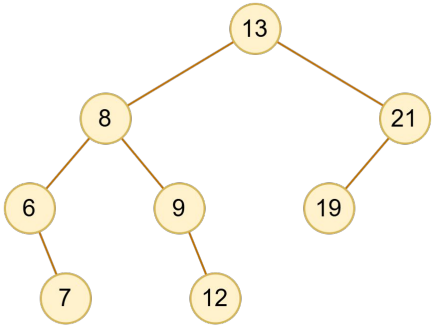
Ağaçlar - İkili Ağaçlar - Ağaç Oluşturma #2

Şu değerlerden bir ağaç inşa edelim: 13, 8, 9, 21, 6, 12, 7, 19, 44, 47, 33



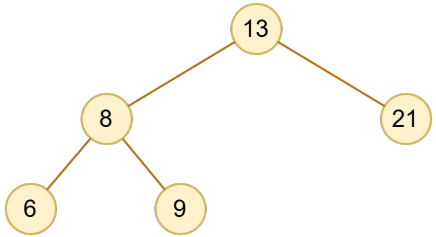


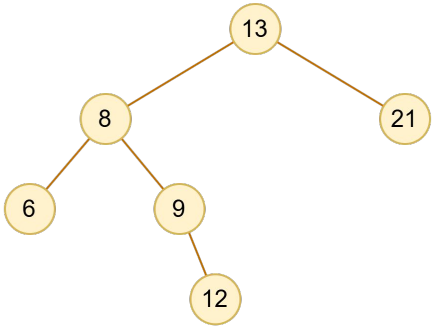


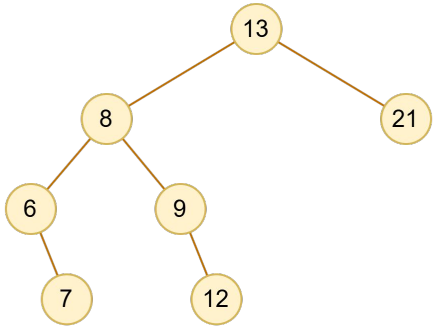


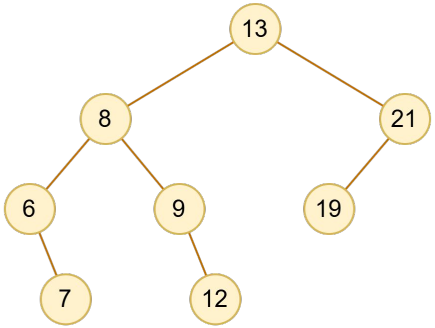
Ağaçlar - İkili Ağaçlar - Ağaç Oluşturma #3

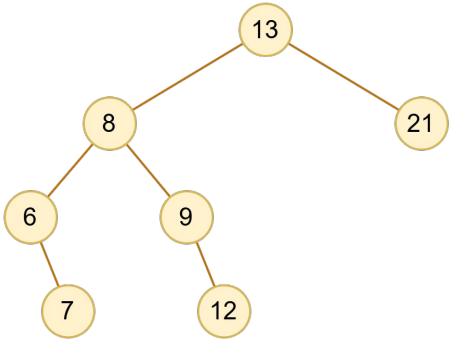
Şu değerlerden bir ağaç inşa edelim: 13, 8, 9, 21, 6, 12, 7, 19, 44, 47, 33

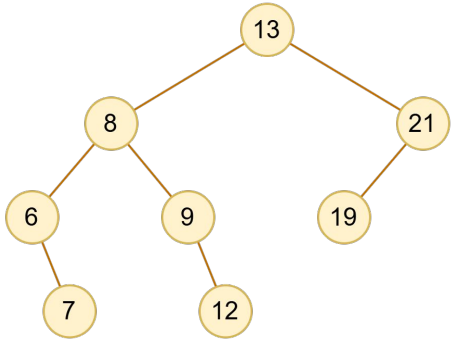






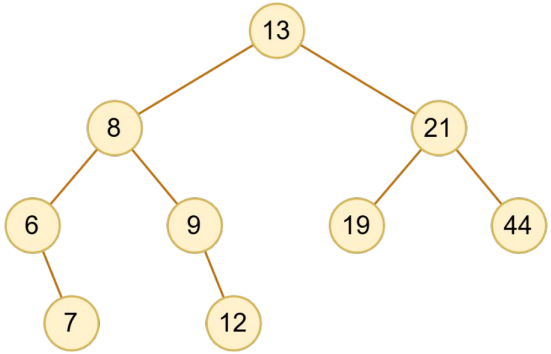


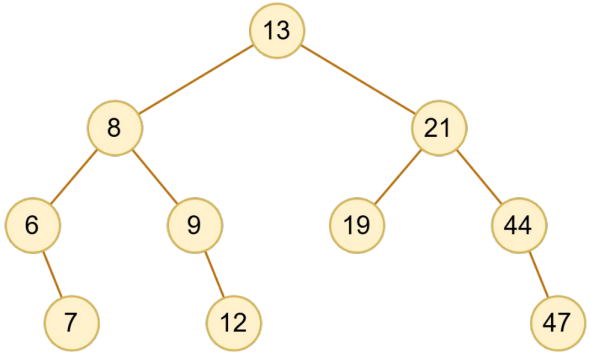


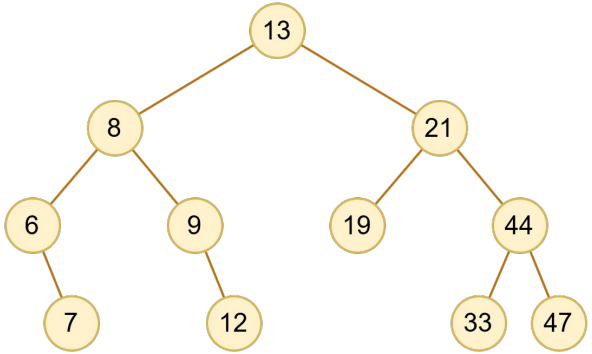


Ağaçlar - İkili Ağaçlar - Ağaç Oluşturma #4

Şu değerlerden bir ağaç inşa edelim: 13, 8, 9, 21, 6, 12, 7, 19, 44, 47, 33



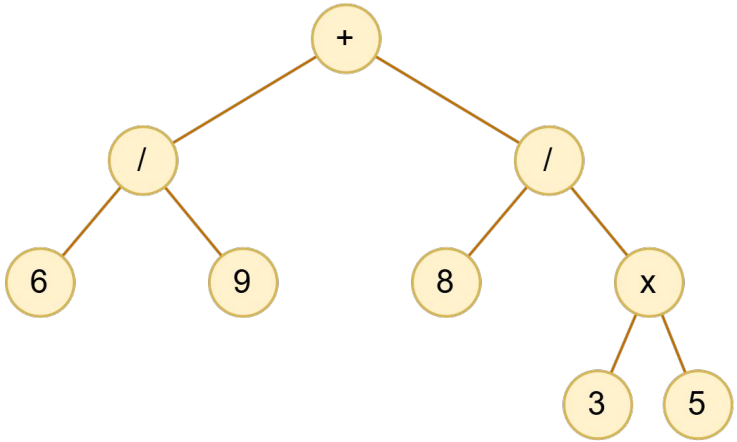




Ağaçlar - İkili Ağaçlar - İfadelerin Gerçekleştirimi

İkili ağaçlar, matematiksel, programlama kodu vb. ifadelerin yazılması ve ayrıştırılması için de kullanılmaktadır. Şekilde, bir ifadenin ikili ağaç yapısında temsil edilmesi gösterilmiştir.

Yaprak olmayan düğümler işlemleri ve operatörleri ifade ederken, yapraklar üzerinde işlem yapılacak olan operandları ifade etmektedir. Şimdi bu ifadeyi Kök Önce, Kök Ortada ve Kök Sonra yöntemleri ile ayrıştıralım:



Kök Önce : + / 6 9 / x 3 5 8

Kök Ortada : 6 / 9 8 / 3 X 5

Kök Sonra : 6 9 / 8 3 5 X / +