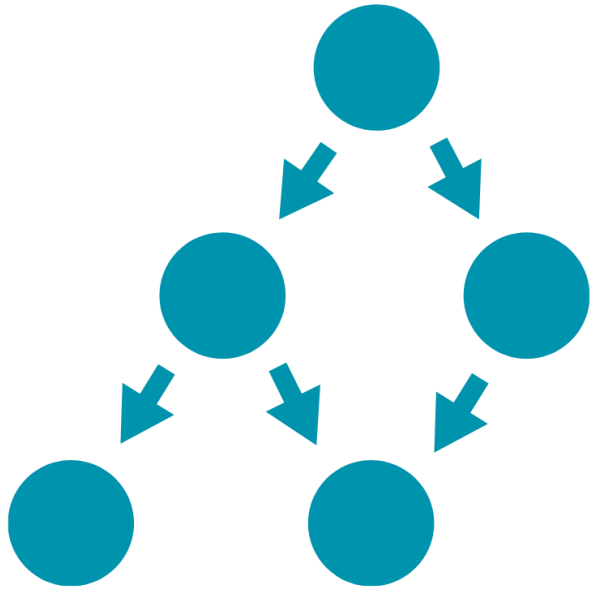
BTK AKADEMİ

ZAFER CÖMERT
Öğretim Üyesi



VERİ YAPILARILARI VE ALGORİTMALAR

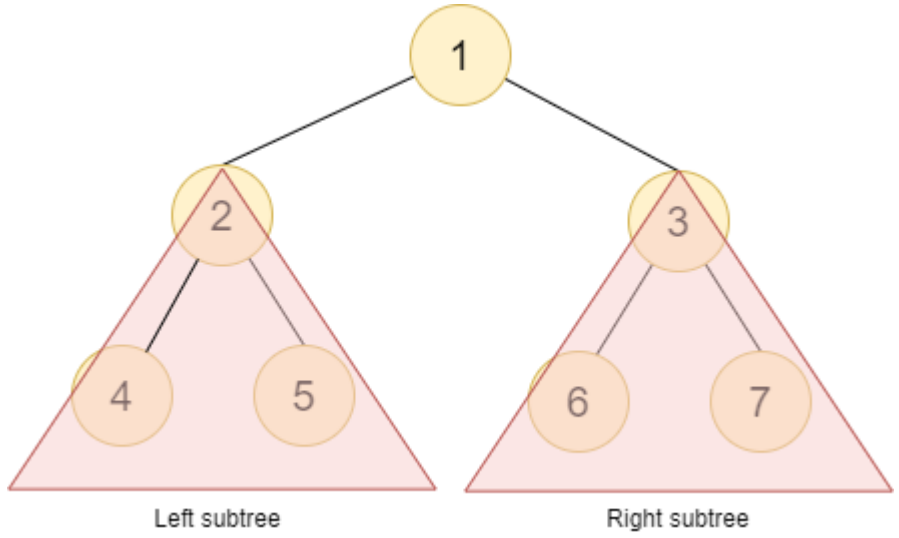
Binary Trees

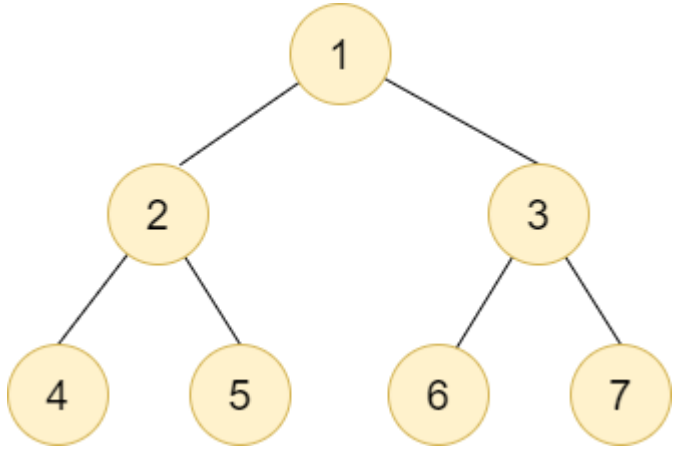
Giriş

1. İkili ağaçlar
2. İkili ağaç türleri
3. İkili ağaçların özellikleri
4. İkili ağacın yapısı
5. İşlevler
    1. Temel işlevler
    2. Yardımcı işlevler
6. İkili ağaç uygulamaları
7. İkili ağaçta gezinme

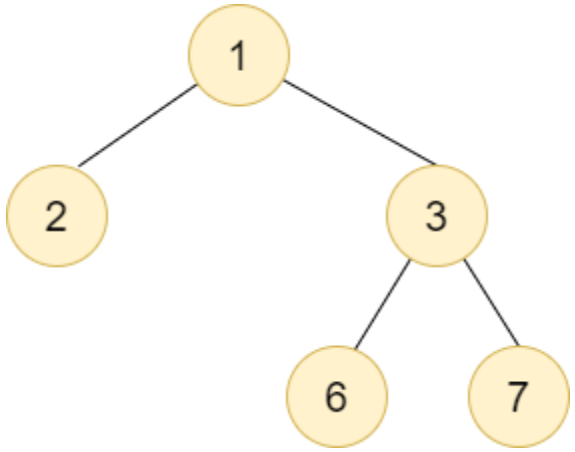
İkili ağaçlar

- Bir ağaç, hiç çocuğu yoksa, tek çocuğu var ise ya da iki çocuğu var ise ikili ağaç olarak adlandırılır. Boş bir ağaç geçerli bir ikili ağaçtır.



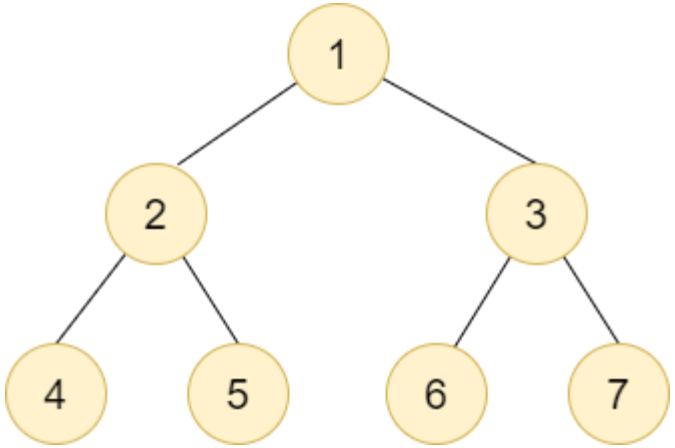


İkili ağaçlar



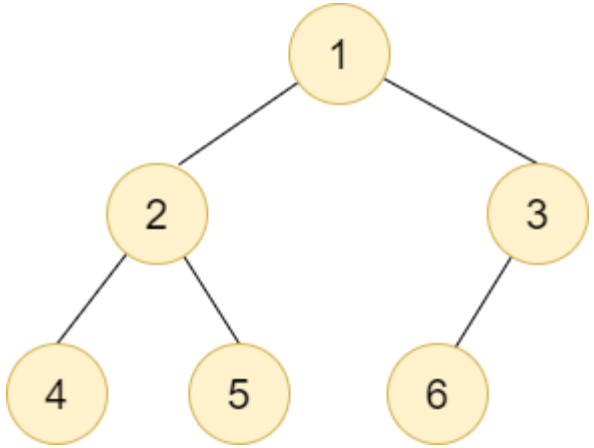
Strict binary tree

İki çocuğu var ya da hiç yok.



Full binary tree

Her düğümün iki çocuğu var.

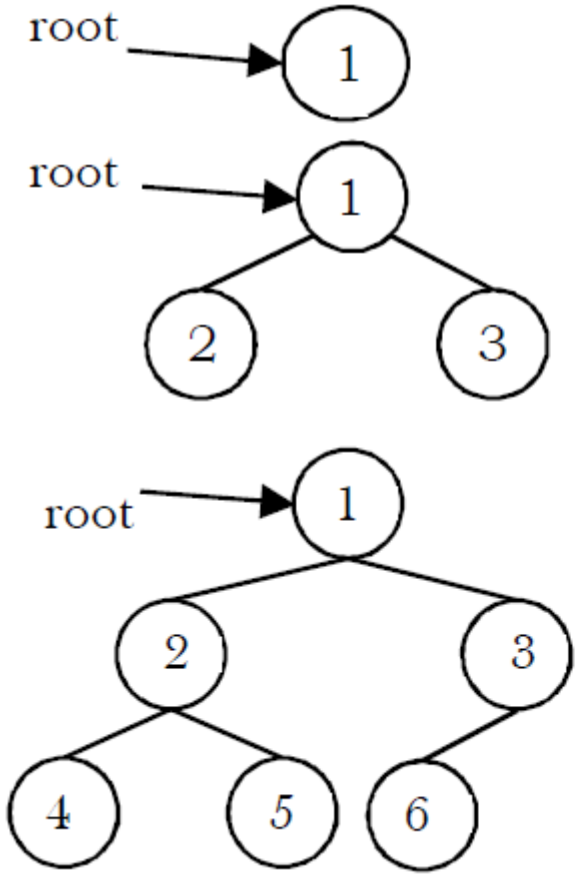


Complete binary tree

Düğümlerin doldurulmasına sol yapraktan başlanır.

BTK AKADEMİ

İkili ağaçların özellikleri



Height	Number of nodes at level h
$h = 0$	$2^0 = 1$
$h = 1$	$2^1 = 2$
$h = 2$	$2^2 = 4$

İkili ağaçların özellikleri

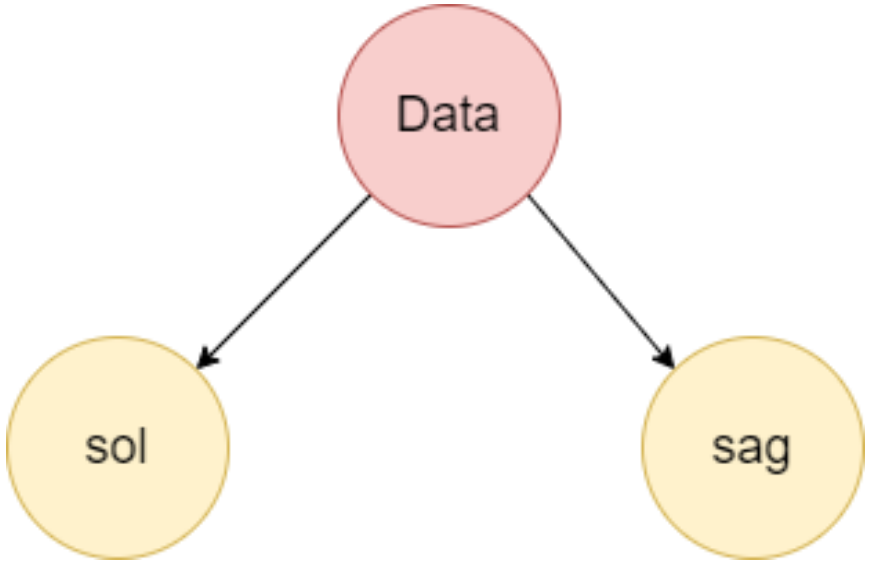
- Full iki ağaçtaki eleman sayısı: $2^{h+1} - 1$
  $[2^0 + 2^1 + \cdots + 2^h = 2^{h+1} - 1]$

- Tam ikili ağaçtaki eleman sayısı minimum 2^h ve maksimum $2^{h+1} - 1$ aralığında olur.

- Bir full ikili ağaçta sol düğümlerin sayısı 2^h olur.

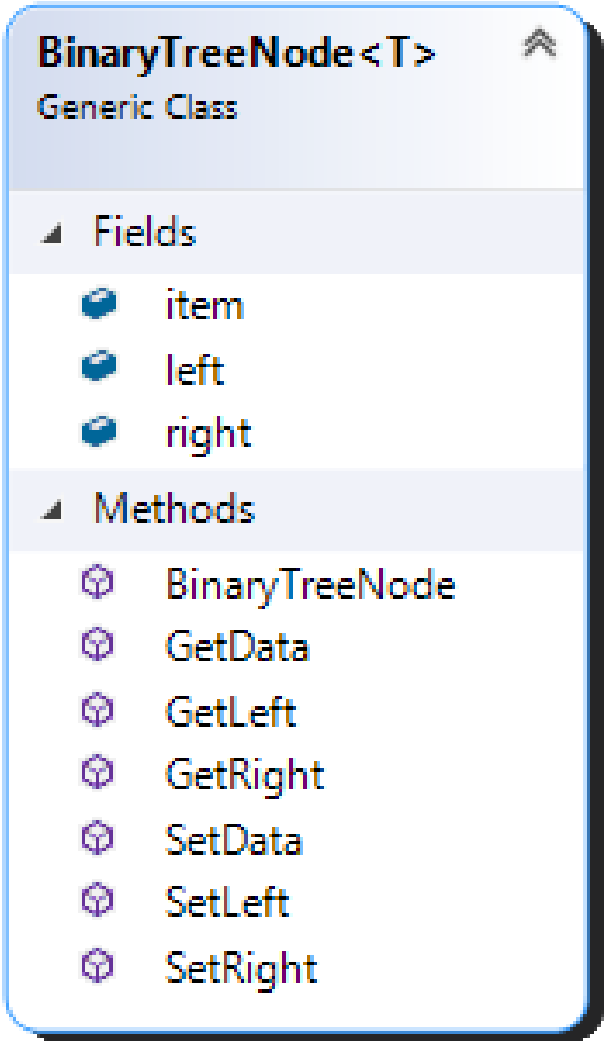
İkili ağacın yapısı

- İkili ağaçlarda düğüm yapısı kullanılır.
- Bu düğüm içerisinde veriyi tutacak bir alan yer alır.
- Ayrıca sol ve sağı gösterecek şekilde iki adet işaretçi de yine düğüm içeriğinde tanımlanır.



Düğüm tasarımı

- Yandaki şekilde Generic veri taşıyacak şekilde modellenmiş bir ağaç düğüm yapısına yer verilmiştir.
- Düğüm tasarımları değişken olabilir.
- Düğümler üzerinde daha farklı alan ve özellikler tanımlanabilir.



İkili Ağaçlarda İşlevler

- **Temel işlevler**
  - Ağaca ekleme yapma
  - Ağaçta silme
  - Ağaçta arama yapma
  - Ağaçta dolaşma
- **Yardımcı işlevler**
  - Ağacın boyutunu bulma
  - Ağacın yüksekliğini bulma
  - Ağacın seviyesini bulma
  - Verilen bir çift düğüm için en az ortak ata bulma (`least common ancestor`)

İkili Ağaçların Uygulamaları

Application of Binary Trees

- İfade ağaçları (**`Expression trees`**) derleyicilerde kullanılır.
- **`Huffman coding`** algoritması sıkıştırma için kullanılır.
- **`Binary Search Tree (BST)`**, ikili arama ağacı **`O(nlogn)`** karmaşıklığında arama yapmak üzere kullanılır.
- **`Priority Queue (PQ)`**, minimum ya da maksimum elemanları arayıp, silmek üzere desteklenen bir veri yapısı dahilinde kullanılır.

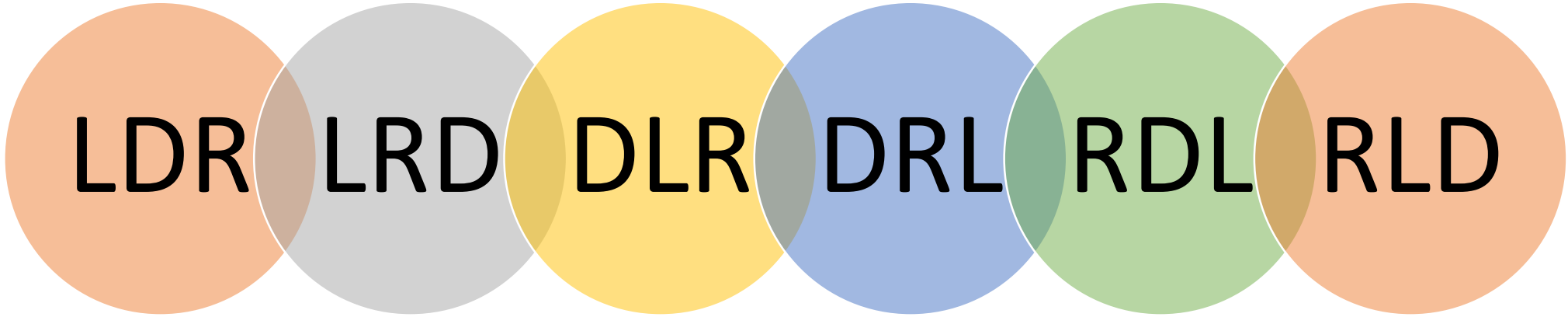
İkili Ağaçlarda Gezinme

Binary Tree Traversal

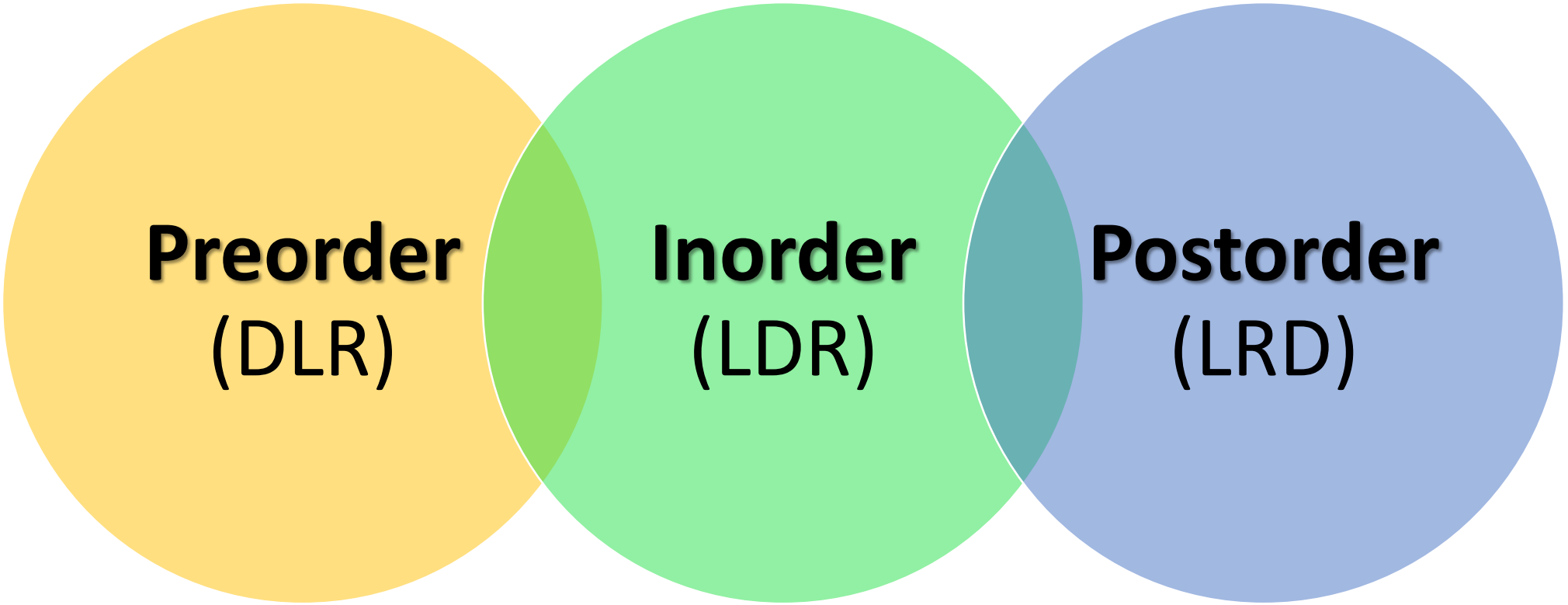
İkili Ağaçlarda Gezinme

- Ağaçlar üzerindeki elemanları işleyebilmek için onlara erişmek gerekir.
- Bu nedenle ağaçta gezinmeyi sağlayacak mekanizmalara ihtiyaç duyulur.
- Bir ağaç üzerindeki tüm düğümleri gezme/dolaşma `tree traversal` olarak ifade edilir.

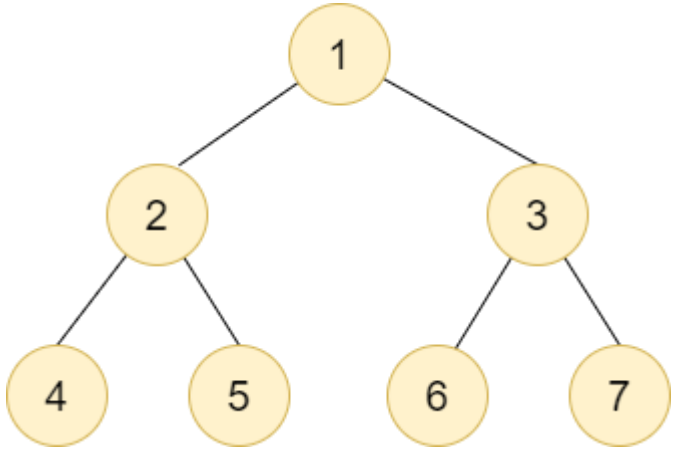
İkili Ağaçlarda Gezinme



İkili Ağaçlarda Gezinme



İkili Ağaçlarda Gezinme : PreOrder



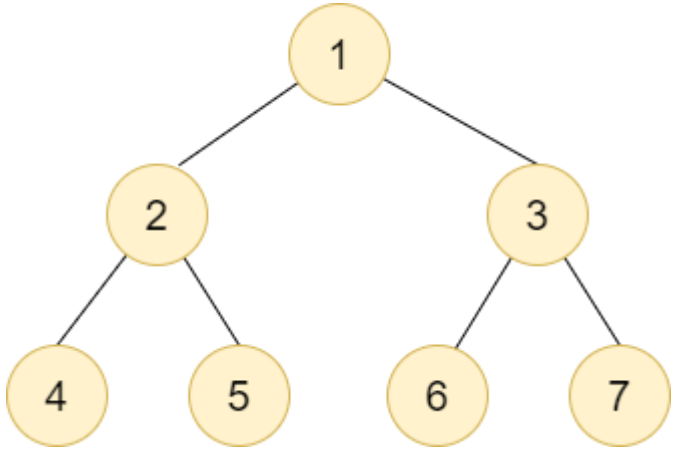
- PreOrder: 1 2 4 5 3 6 7
- Zaman ve Bellek Maliyeti : $O(n)$

İkili Ağaçlarda Gezinme : PreOrder

```
public void PreOrder(BinaryTreeNode<T> root)
{
    if (root!=null)
    {
        Console.WriteLine(root.item);
        PreOrder(root.left);
        PreOrder(root.right);
    }
}
```

- PreOrder: 1 2 4 5 3 6 7
- ArrayList ve Stack kullanımı ile iteratif bir şekilde de tasarlanabilir.

İkili Ağaçlarda Gezinme : InOrder



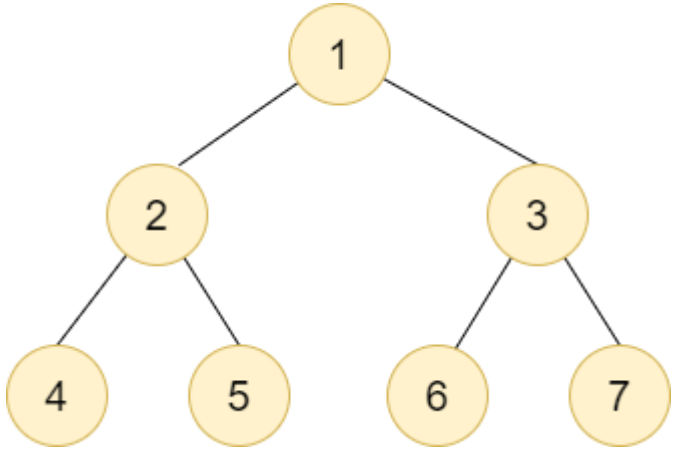
- InOrder: 4 2 5 1 6 3 7
- Zaman ve Bellek Maliyeti: $O(n)$

İkili Ağaçlarda Gezinme : InOrder

```
public void InOrder(BinaryTreeNode<T> root)
{
    if (root != null)
    {
        InOrder(root.left);
        Console.WriteLine(root.item);
        InOrder(root.right);
    }
}
```

- InOrder: 4 2 5 1 6 3 7
- ArrayList ve Stack kullanımı ile iteratif bir şekilde de tasarlanabilir.

İkili Ağaçlarda Gezinme : PostOrder



- PostOrder: 4 5 2 6 7 3 1
- Zaman ve Bellek Maliyeti: $O(n)$

İkili Ağaçlarda Gezinme : PostOrder

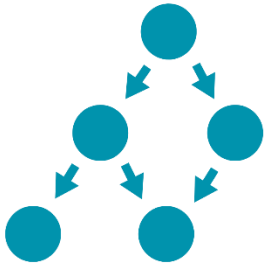
```
public void PostOrder(BinaryTreeNode<T> root)
{
    if (root != null)
    {
        PostOrder(root.left);
        PostOrder(root.right);
        Console.WriteLine(root.item);
    }
}
```

- PostOrder: 4 5 2 6 7 3 1
- ArrayList ve Stack kullanımı ile iteratif bir şekilde de tasarlanabilir.

İkili Ağaçlarda Gezinme : PostOrder

```
public void PostOrder(BinaryTreeNode<T> root)
{
    if (root != null)
    {
        PostOrder(root.left);
        PostOrder(root.right);
        Console.WriteLine(root.item);
    }
}
```

- PostOrder: 4 5 2 6 7 3 1
- ArrayList ve Stack kullanımı ile iteratif bir şekilde de tasarlanabilir.



Veri Yapıları ve Algoritmalar

ZAFER CÖMERT
Öğretim Üyesi

BTK AKADEMİ