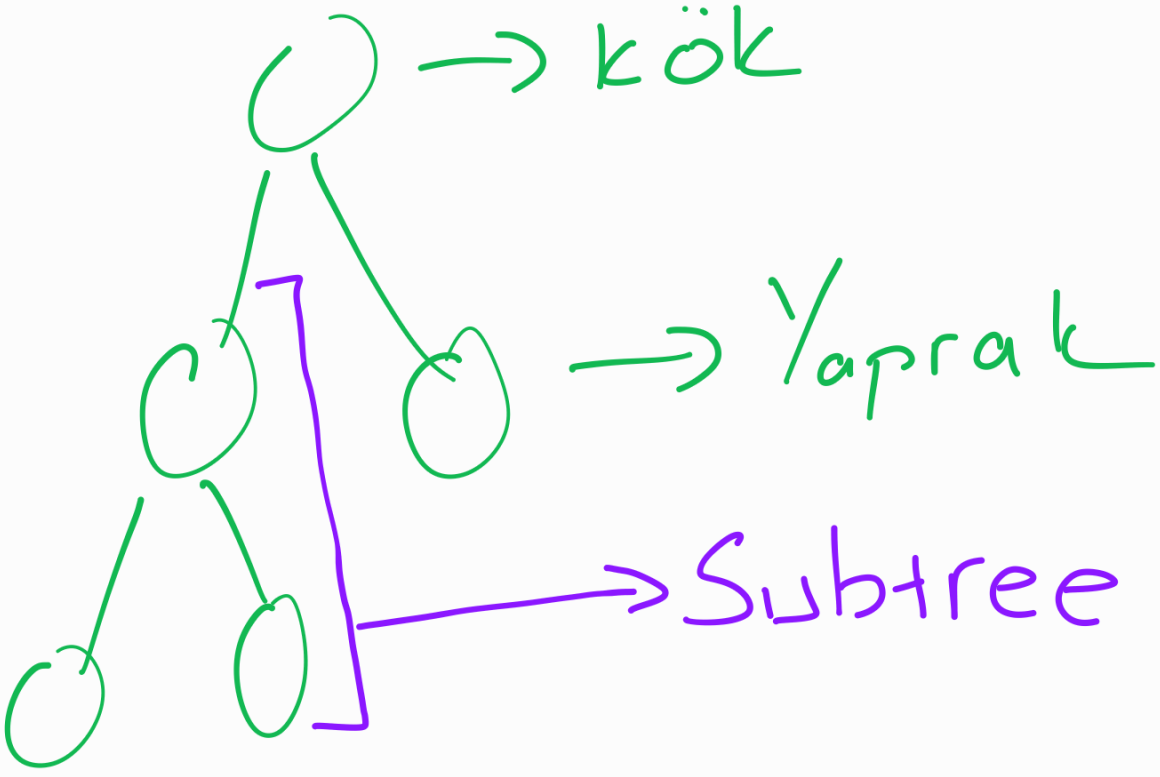
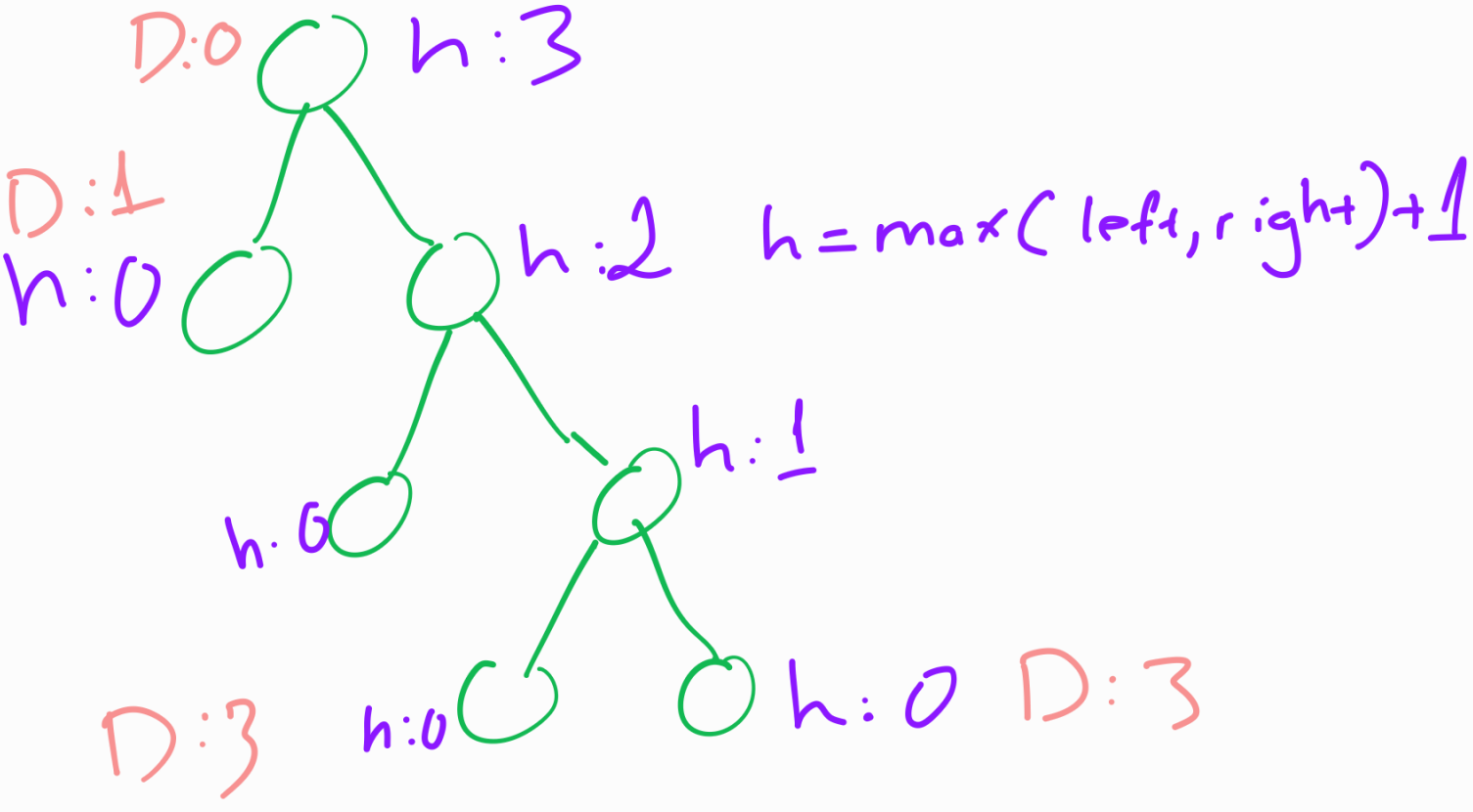




Depth : Reverse of height

Recursive

preorder gezinti

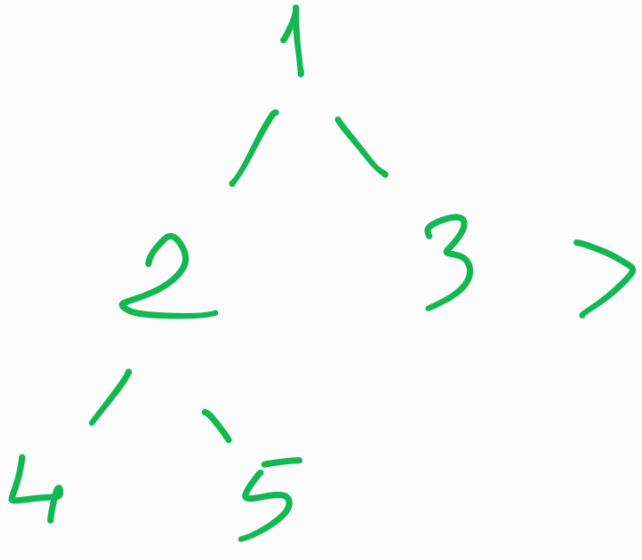
```java
// Preorder traversal
void preorder(Node node) {
    if (node == null) {
        return;
    }

    // 1. Düğümün değerini yazdır
    System.out.print(node.value + " ");
    // 2. Sol alt ağacı gez
    preorder(node.left);
    // 3. Sağ alt ağacı gez
    preorder(node.right);
}
```

> Nod yaz

> lefte git

> Righta git



> 1 2 4 5 3 şeklinde çıktı alır.

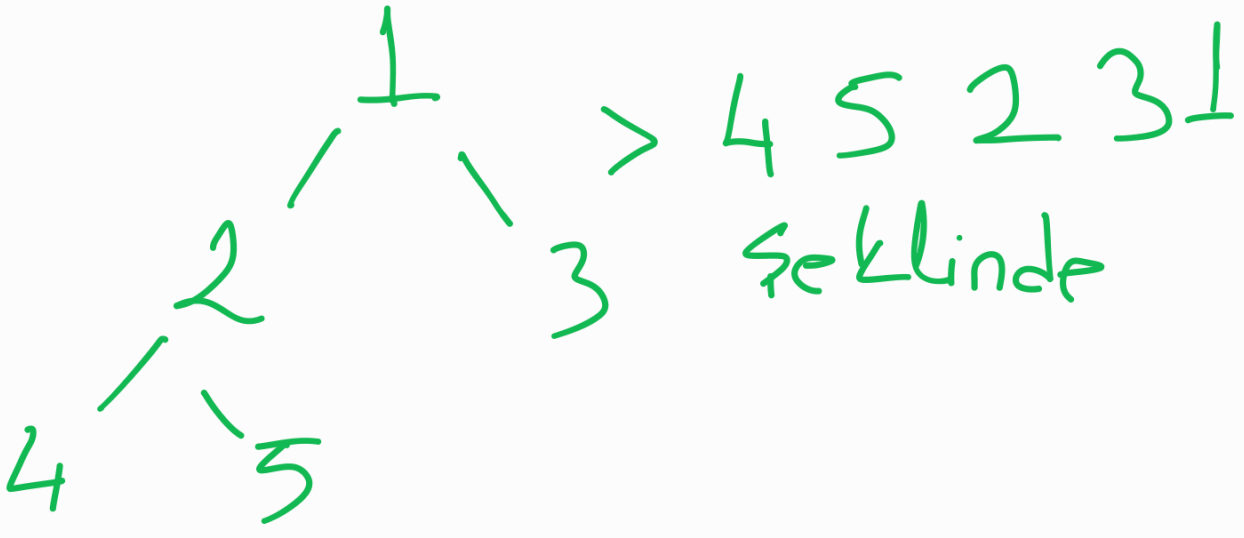
```java
// Postorder traversal
void postorder(Node node) {
    if (node == null) {
        return;
    }

    // 1. Sol alt ağacı gez
    postorder(node.left);
    // 2. Sağ alt ağacı gez
    postorder(node.right);
    // 3. Düğümün değerini yazdır
    System.out.print(node.value + " ");
}
```

> Sola git

> Sağa bak

> node yaz



> 4 5 2 3 1 şeklinde

Node Yükseklik Bulma kodu

```
    // Belirli bir düğümün yüksekliğini hesapla
    int getNodeHeight(Node node) {
        if (node == null) {
            return -1; // Boş düğümün yüksekliği -1 olarak
kabul edilir.
        }

        // Sol ve sağ alt ağaçların yüksekliklerini hesapla
        int leftHeight = getNodeHeight(node.left);
        int rightHeight = getNodeHeight(node.right);

        // Maksimum yüksekliği döndür
        return Math.max(leftHeight, rightHeight) + 1;
    }
```

Recursive olarak alta in Null yiyince return almaya başla

BST için sıralı yazdırma

En lefte git (küçük)
yaz
bi Sağa bak (bir büyük)

```
void inorder(Node node) {
    if (node == null) {
        return;
    }

    // Sol alt ağacı gez
    inorder(node.left);
    // Düğümü yazdır
    System.out.print(node.value + " ");
    // Sağ alt ağacı gez
    inorder(node.right);
}
```