

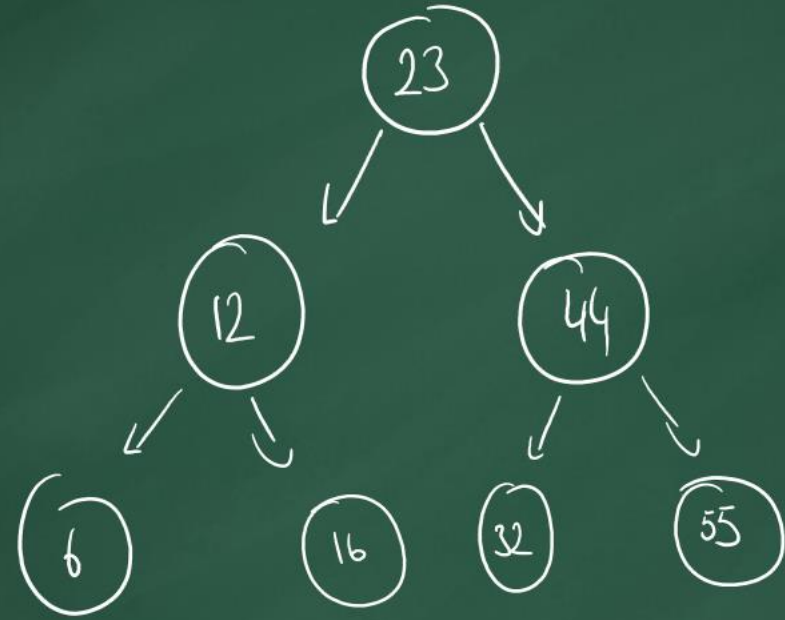
VERİ YAPILARILARI VE ALGORİTMALAR

BST Problems

# Giriş

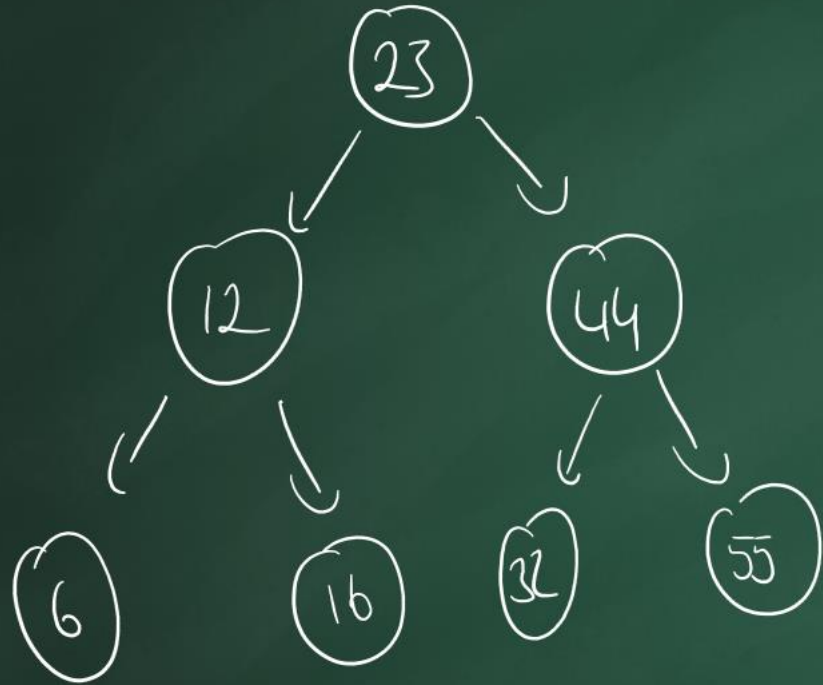
1. Level Order Dolaşma
2. Maksimum derinlik (Maximum depth)
3. Maksimum elemanın bulunması (Maximum item)
4. En derin elemanın bulunması (Deepest node)
5. Yaprak sayısının bulunması (Number of Leaves)
6. Ağaçtaki full-düğüm sayısının bulunması (Number of Full Nodes)
7. Ağaçtaki yarım-düğüm sayısının bulunması (Number of Half Nodes)
8. Bir ağacın genişliği hesaplayan metot tasarımı (Width of tree)
9. Yığın ile PreOrder Traversal (PreOrder Traversal with Stack)
10. Kökten yapraklara olan yolları yazdıran metot tasarımı (Paths)
11. Bir ağacın simetrisini alan metot tasarımı (MirrorOfTree)

Problem : Level Order traversal gerçekleştirilen metot tasarımı



Output → 23, 12, 44, 6, 16, 32, 55

# Level Order Traversal



Output

23, 12, 44, 6, 16, 32, 55

q → 23

De Queue

23

q → 12 44

12

q → 44 6 16

44

q → 6 16 32 55

6

q → 16 32 55

16

q → 32 55

32

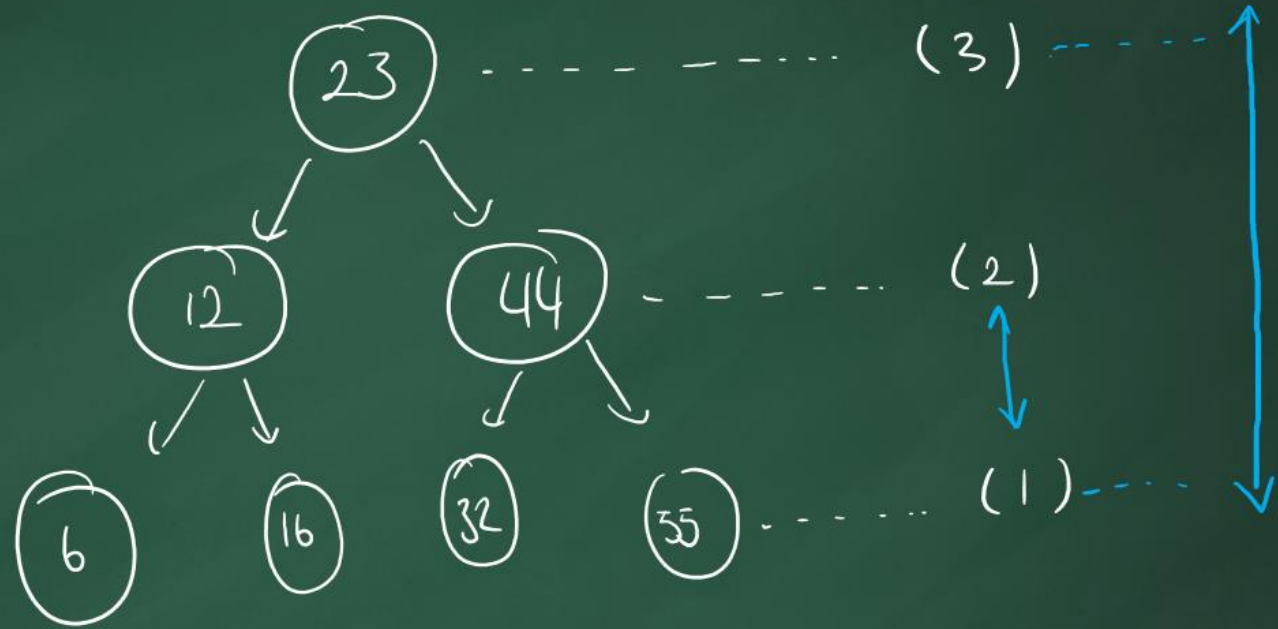
q → 55

55

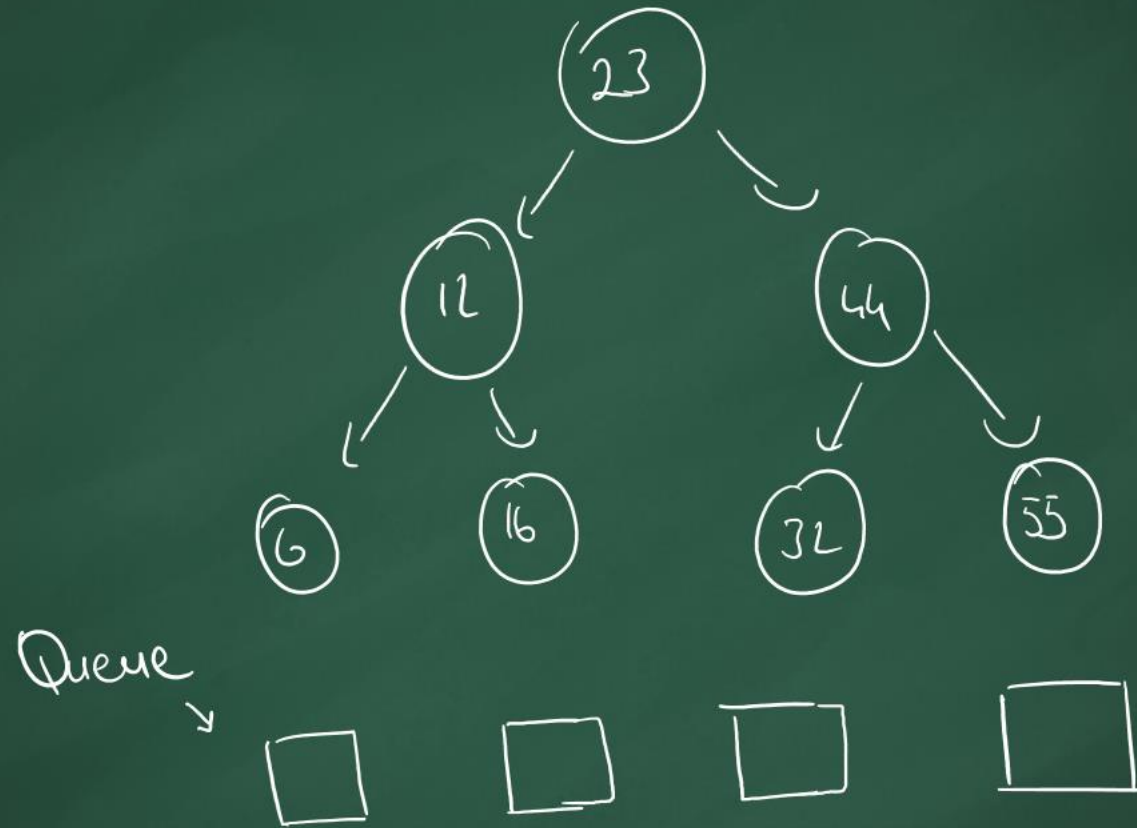
q → { }

"First-in First-Out - FIFO"  
Queue

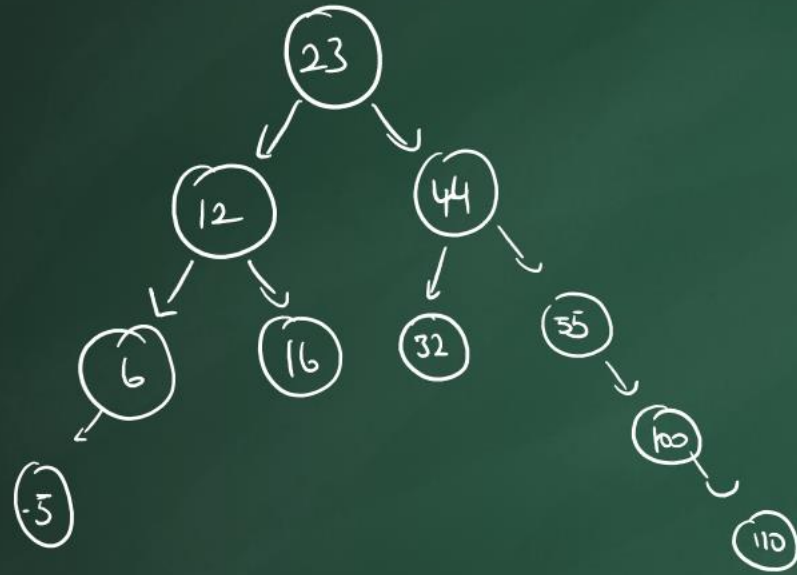
Problem: Bir ağaçtaki maksimum derinliğin bulunması.



Problem : Maksimum elemen iteratif olarak bulunması



Problem : Ağacdaki en derin düğümü bulan netot tasarımı



9 → 23  
 → 12 44  
 → 44 6 16  
 → 6 16 32 55  
 → 16 32 55 -5  
 → 32 55 -5  
 → 55 -5  
 → -5 100  
 → 100  
 → 120

Enquey

23

12

44

6

16

32

55

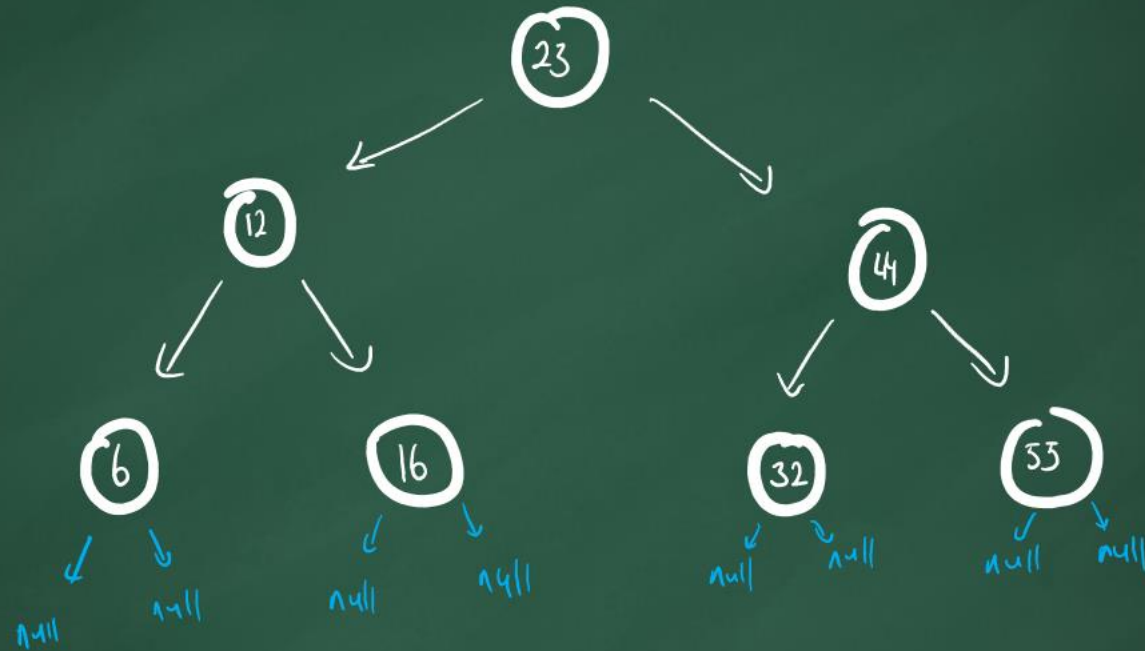
-5

100

120

base case → return 120

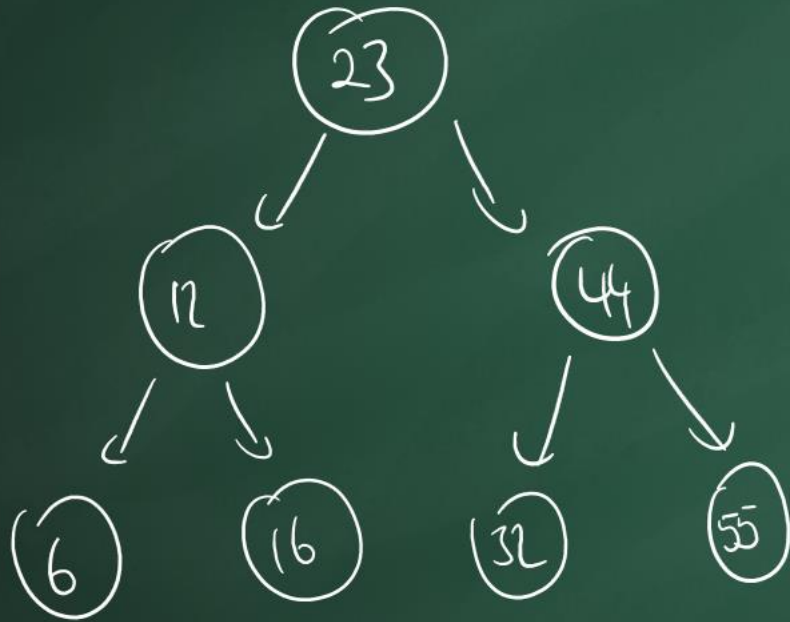
Problem : Ağactaki yaprak sayısını bulan metod tasarımı.



if (temp.Left == null && temp.Right == null)  
count++



Problem → Ağaçtaki full düğüm sayısını verecek metod tasarımı.



– Level-order traversal

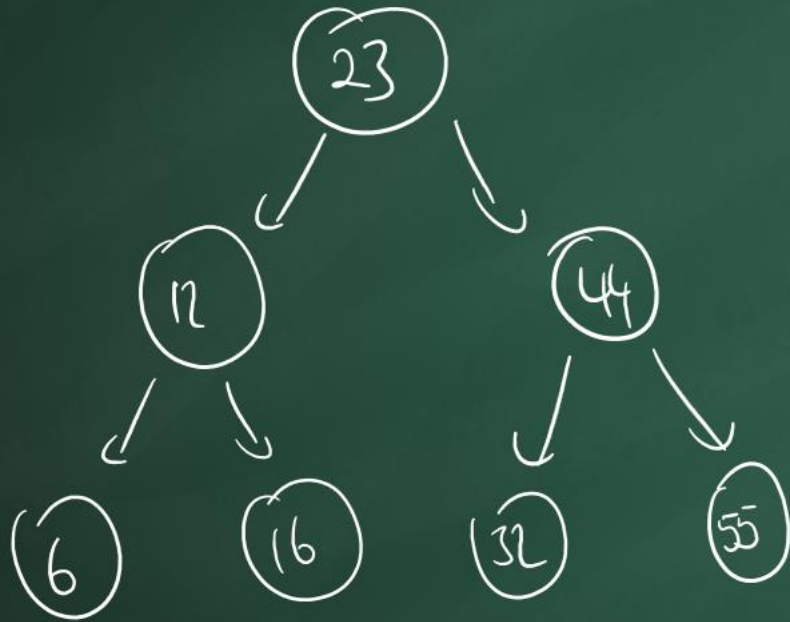
– Node

if (temp.Left != null &&  
temp.Right != null)

count = count + 1

Polarma sonucunda temp düğümün sayısı elde edilir.

Problem → Ağaçtaki yarım düğüm sayısını verecek metod tasarımı.



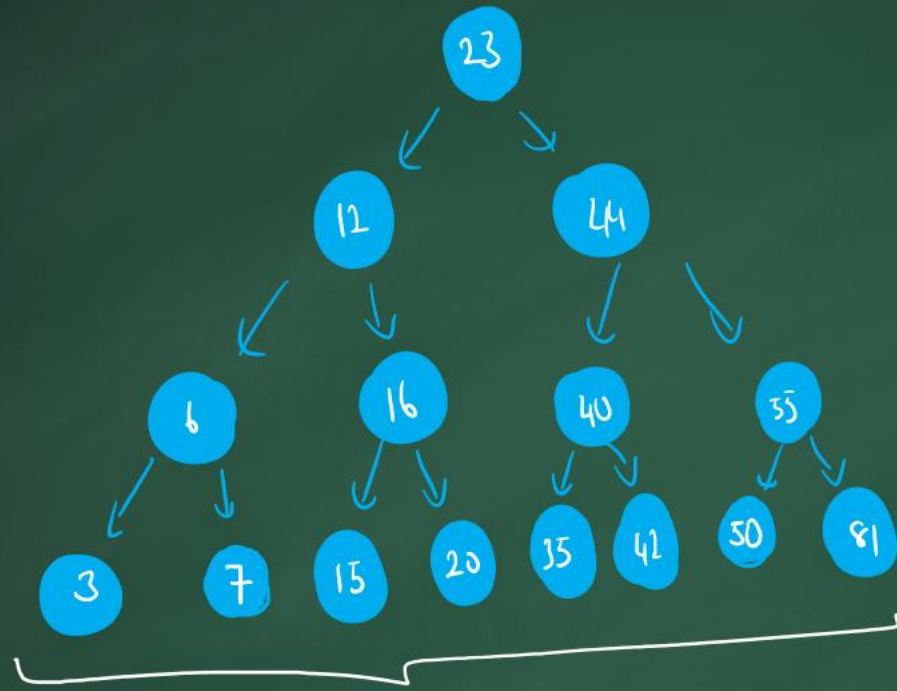
— Level-order traversal

— Node

. temp.Left = null && temp.Right != null ||  
. temp.Left != null && temp.Right = null

count = count + 1

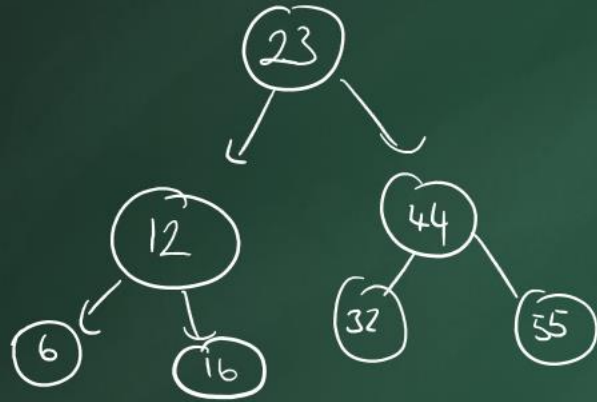
Polarma sonucunda tüm düğüm sayıları elde edilir.



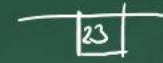
Width of tree (8)

Problem → Bir ağacın genişliğini bulan metod tasarımı  
gerçekleştiriniz.

# PreOrder Traversal with Stack



Stack(S) → S



S → push → 23

..... (S.Count > 0)

while

S → pop()

S → push → 44

S → push → 12

≡



S → pop()

S → push → 32

S → push → 32

≡

23



≡

12



S → pop()

S → push → 16

S → push → 6

≡



≡

6

S → pop()

S → pop()

≡



S → pop()

≡

16



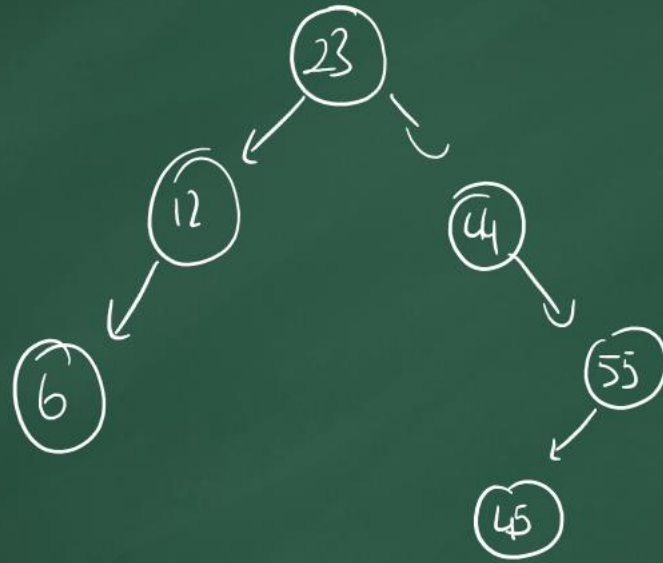
≡

55



.... end while

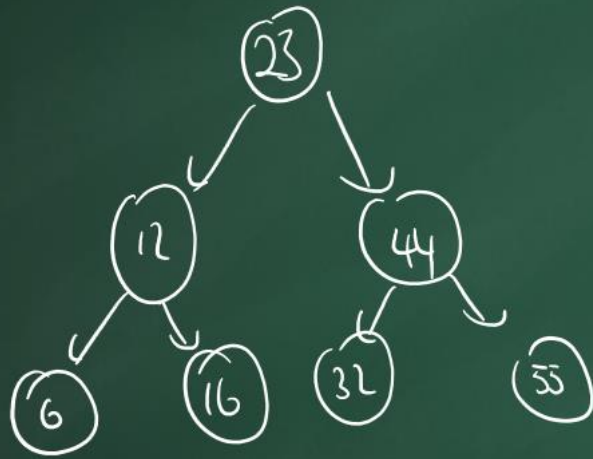
Problem: Kökten yapıtlara olan yolların yazılması



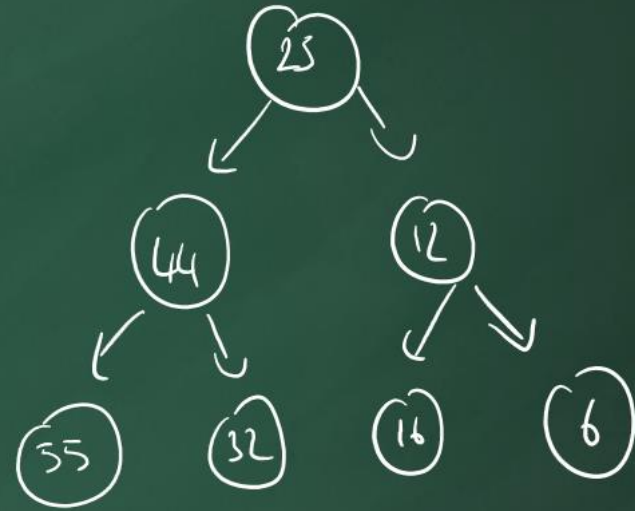
Çıktı → 23-12-6  
23-44-55-45

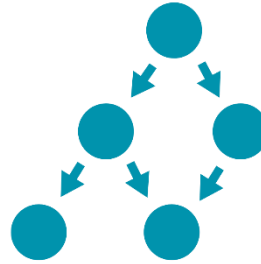


**Problem:** Bir ağacın aynalanmasını / simetrisini sağlayan  
metot tasarımı gerçekleştiriniz.



aynalaması  
→  
mirrorOfTree





Veri Yapıları ve Algoritmalar

**ZAFER CÖMERT**

Öğretim Üyesi