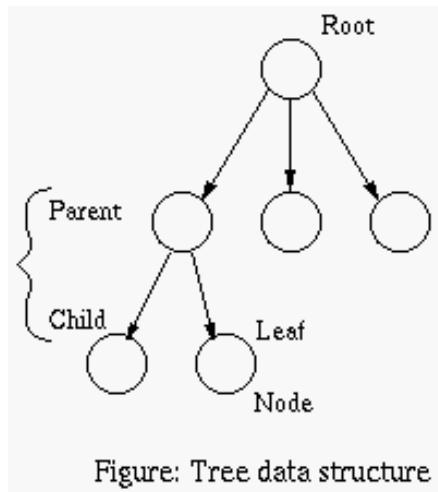


# Veri Yapıları ve Algoritmalar

Ağaçlar

# Ağaçlar

- Dizi, bağlı liste, kuyruk veri yapıları doğrusaldır. (linear)
- Ağaçlar ise iki boyutlu doğrusal olmayan veri yapılarıdır.

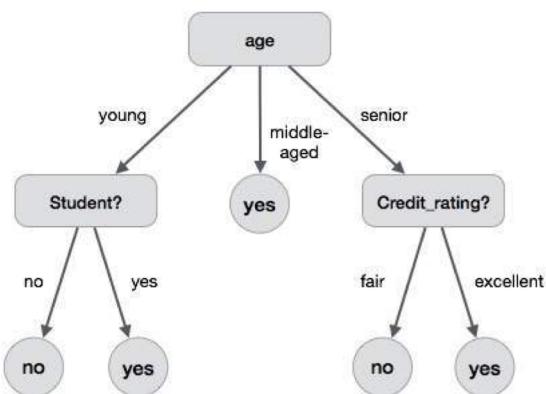
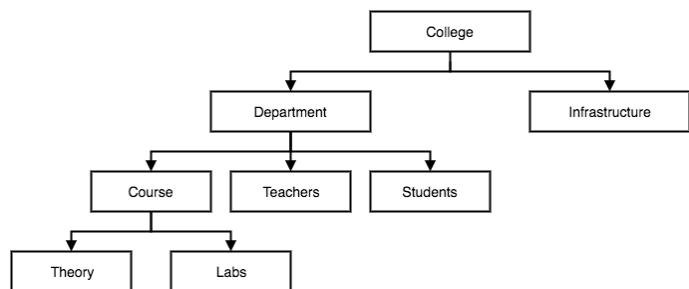


[http://yzgrafik.ege.edu.tr/~ugur/VY\\_01\\_02/Vy\\_99\\_00\\_00\\_01/bolum4.doc](http://yzgrafik.ege.edu.tr/~ugur/VY_01_02/Vy_99_00_00_01/bolum4.doc)

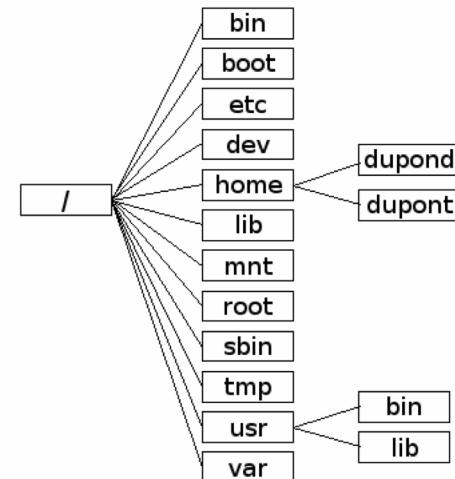
<https://qph.fs.quoracdn.net/main-qimg-10396323bd7b014c34df3ca39185fd6e>

# Ağaçlar

- Nerede Kullanılır ?
  - Organizasyon şemaları
  - Karar ağaçları
  - Dosya sistemi



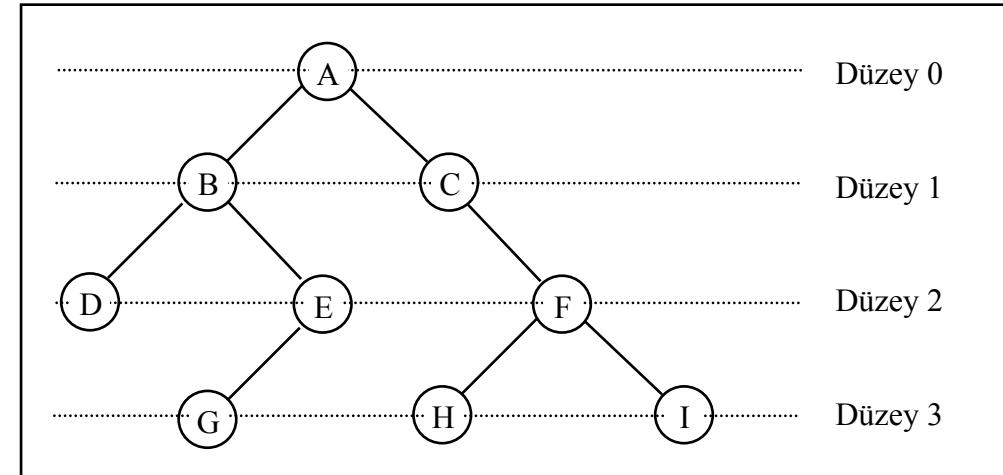
[https://www.tutorialspoint.com/data\\_mining/images/dm\\_decision\\_tree.jpg](https://www.tutorialspoint.com/data_mining/images/dm_decision_tree.jpg)



<http://labor-liber.org/images/linux/arbo-unix.gif>

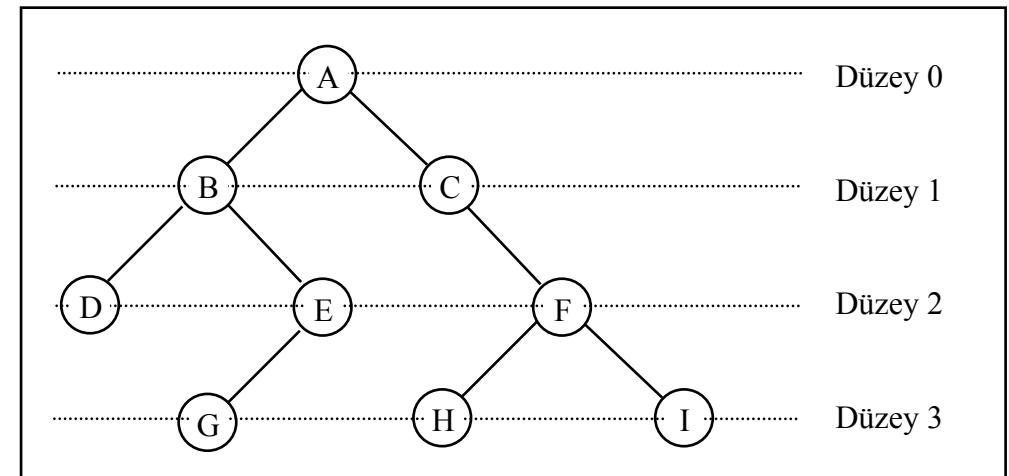
# Ağaçlar Temel Kavramlar

- Kök (Root)
- Düğüm (Node)
- Çocuk (Child)
- Baba (Parent)
- Kardeş (Brother)
- Düzey/Derinlik Düğüm (Level/Depth)
- Derinlik (Depth of tree)
- Üst düğüm (Ancestor)
- Alt düğüm (Descendant)
- Yaprak (Leaf)
- Dal (Branch)



# Ağaçlar Temel Kavramlar

- Root => A
- Node => A,B,C,D,E,F,G,H,I
- Child => (B,C) A'nın çocukları
- Parent => A (B,C)'nin çocukları
- Brother => D-E, B-C, H-I
- Level/Depth => D.nin derinliği=2
- Depth of tree => 3
- Anchestor => A hepsinin atası
- Descendant =>
- Leaf => D, G, H, I

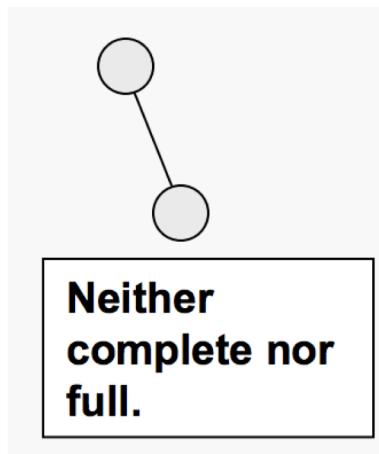
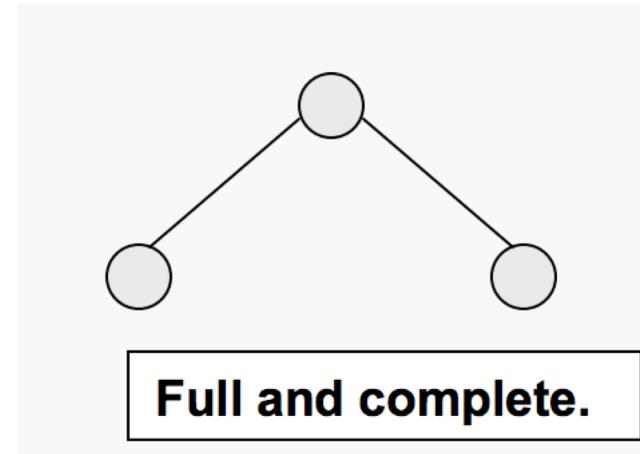
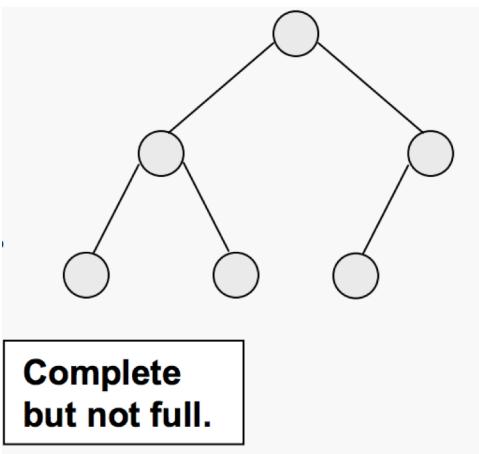
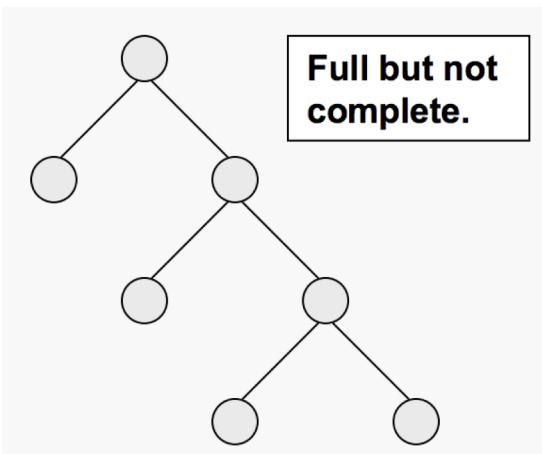


# Ağaçlar

- Ağaç türleri
  - Genel ağaçlar
    - Düğümlerin çocuk sayılarında sınır yok.
  - İkili ağaçlar (Binary trees)
    - Her düğüm en fazla iki alt düğüm içerebilir.
  - Full binary tree
    - Her yaprağı aynı derinlikte olan veya
    - Yaprak olmayan düğümlerin 0/2 çocuğu olan ağaçlardır.
    - $n$  yaprak varsa toplam  $2n-1$  düğüm vardır.
  - Complete binary tree
    - Yeni derinliğe mümkün olduğunca soldan ulaşılır.

# Ağaçlar

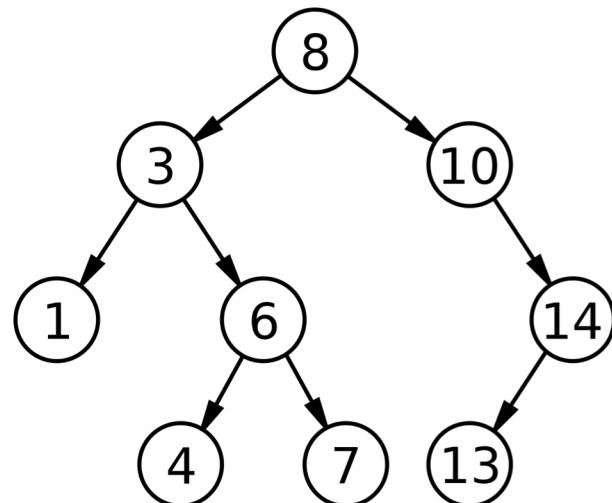
- Ağaç türleri



<http://courses.cs.vt.edu/~cs3114/Fall09/wmcquain/Notes/T03a.BinaryTreeTheorems.pdf>

# Ağaçlar

- İkili arama ağaçları
  - Kökün solundaki alt ağaçlar (varsa) kökten küçük olmalıdır.
  - Kökün sağındaki alt ağaçlar (varsa) kökten büyük olmalıdır.
  - Sol/sağ alt ağaçlar yine ikili arama ağaçları olmalıdır.



[https://upload.wikimedia.org/wikipedia/commons/thumb/d/da/Binary\\_search\\_tree.svg/2000px-Binary\\_search\\_tree.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/d/da/Binary_search_tree.svg/2000px-Binary_search_tree.svg.png)

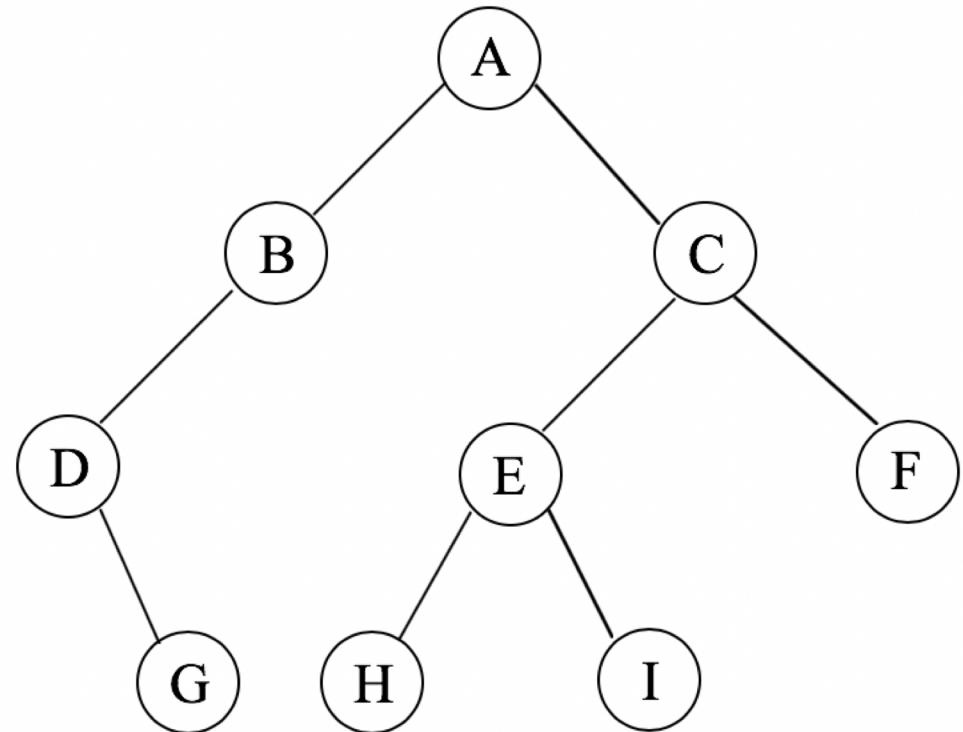
# Ağaçlar

- İkili ağaç üzerinde dolaşma: Özyinelemeli bir işlemdir.
  - Preorder (Depth first order) dolaşma
  - Inorder (Symmetric) dolaşma
  - Postorder

# İkili ağaçlarda dolaşma

- **Preorder (depth-first order) dolaşma**

- Köke uğra (visit)
- Sol alt ağacı preorder olarak dolaş.
- Sağ alt ağacı preorder olarak dolaş.

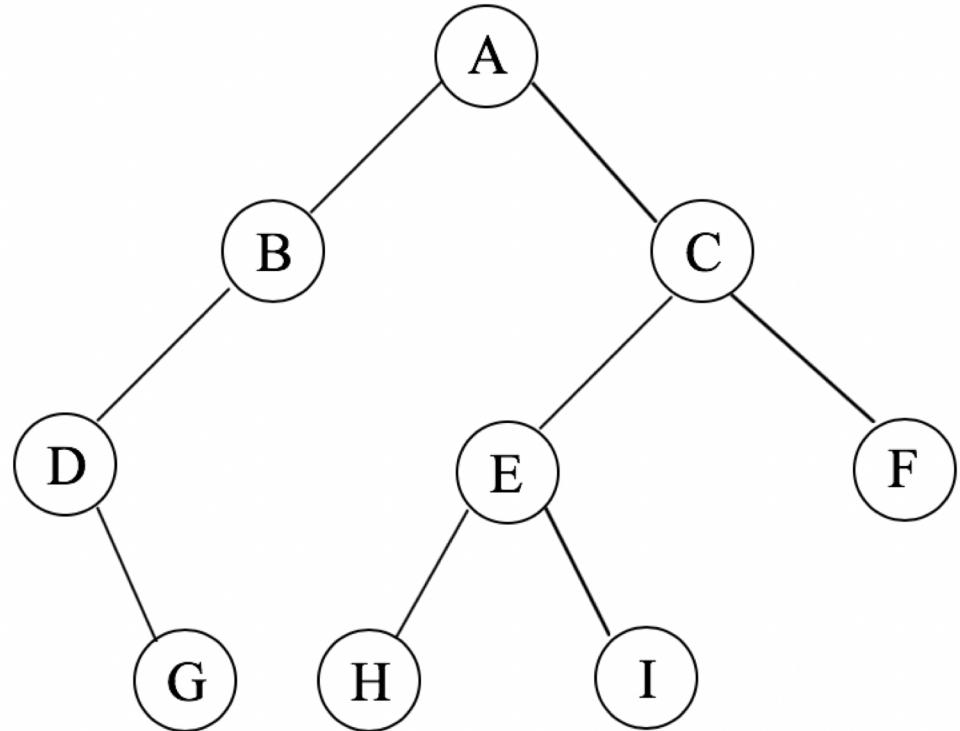


# Preorder: İkili ağaçlarda dolaşma

- **Preorder (depth-first order) dolaşma**

- Köke uğra (visit)
- Sol alt ağacı preorder olarak dolaş.
- Sağ alt ağacı preorder olarak dolaş.

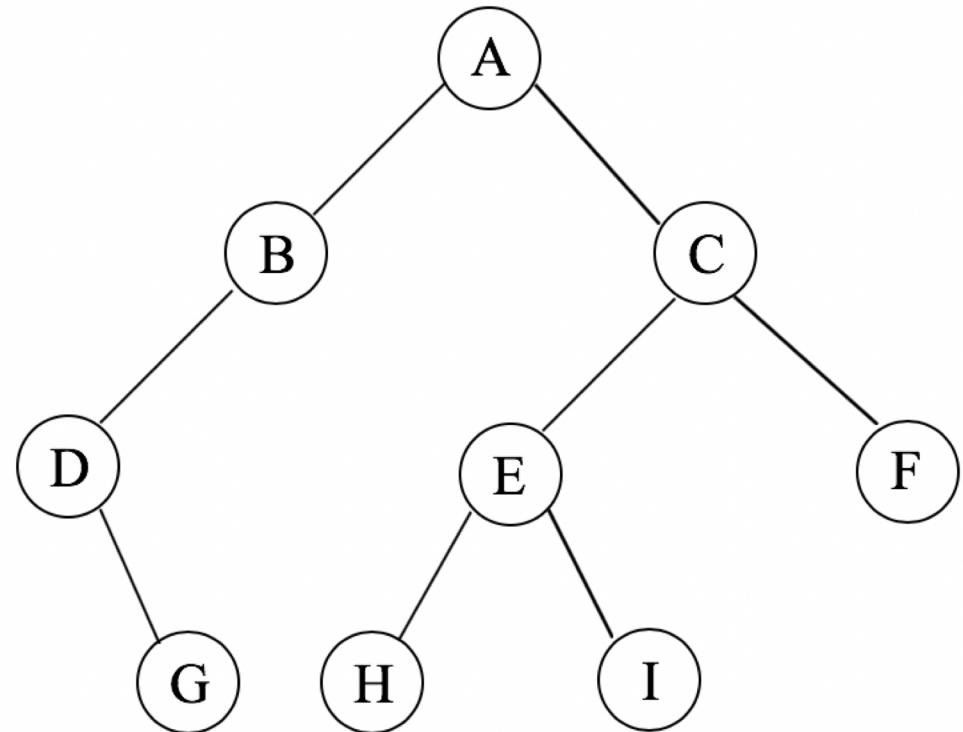
**Preorder:** ABDGCEHIF



# İkili ağaçlarda dolaşma

- **Inorder (Symmetric order) dolaşma**

- Sol alt ağacı inorder'a göre dolaş
- Köke uğra (visit)
- Sağ alt ağacı inorder'a göre dolaş.

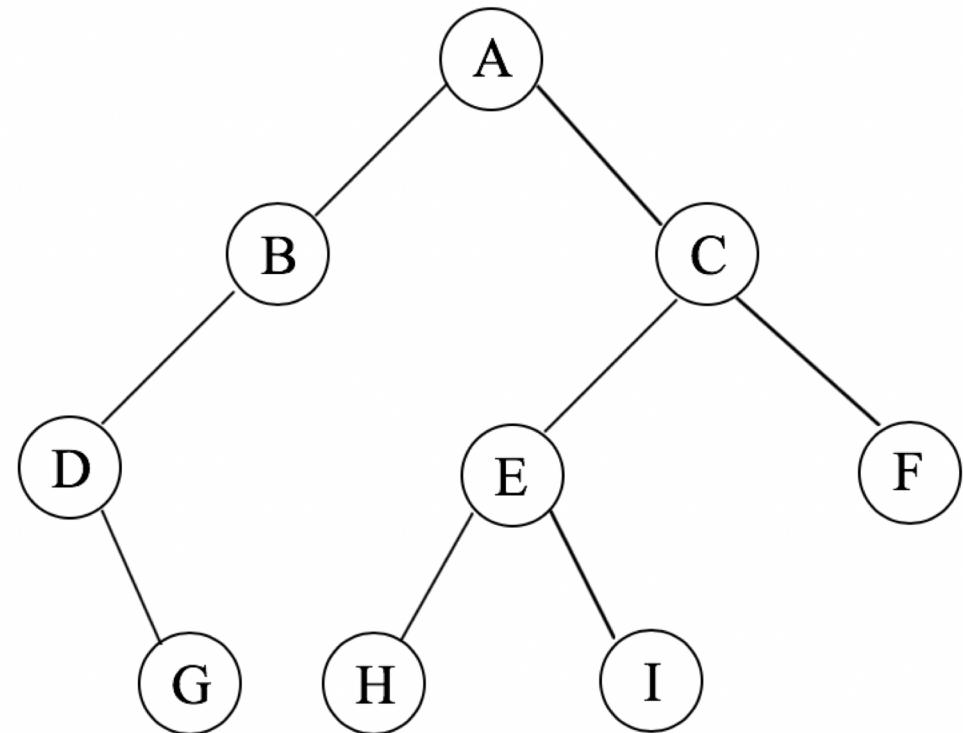


# Inorder: İkili ağaçlarda dolaşma

- **Inorder (Symmetric order) dolaşma**

- Sol alt ağacı inorder'a göre dolaş
- Köke uğra (visit)
- Sağ alt ağacı inorder'a göre dolaş.

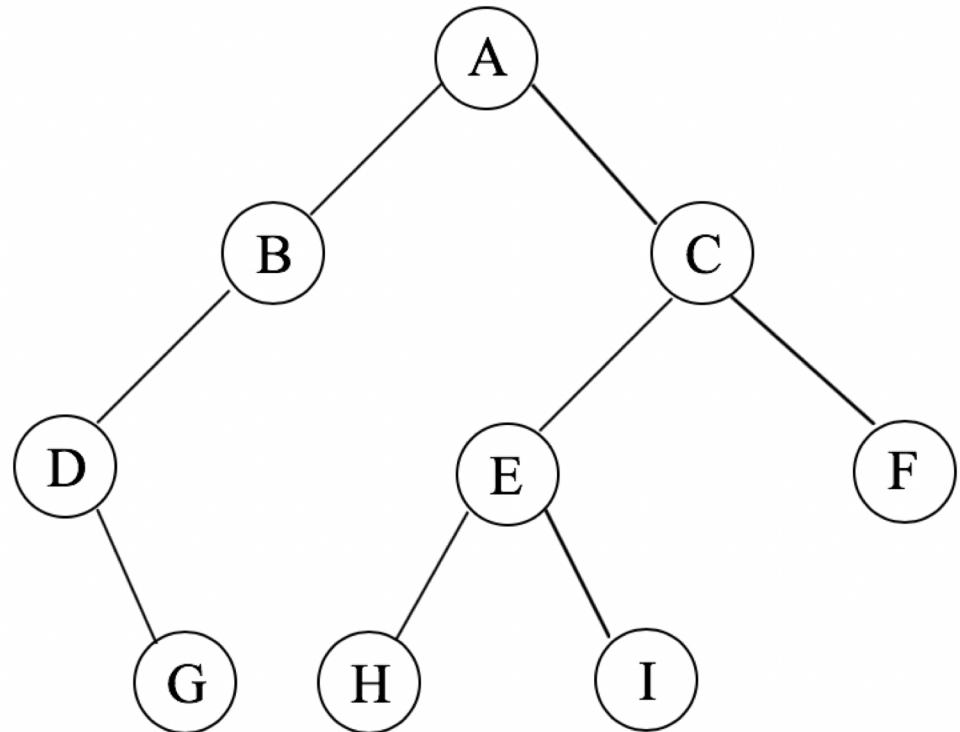
**Inorder:** DGBAHEICF



# İkili ağaçlarda dolaşma

- **Postorder dolaşma**

- Sol alt ağacı postorder'a göre dolaş
- Sağ alt ağacı postorder'a göre dolaş.
- Köke uğra (visit)

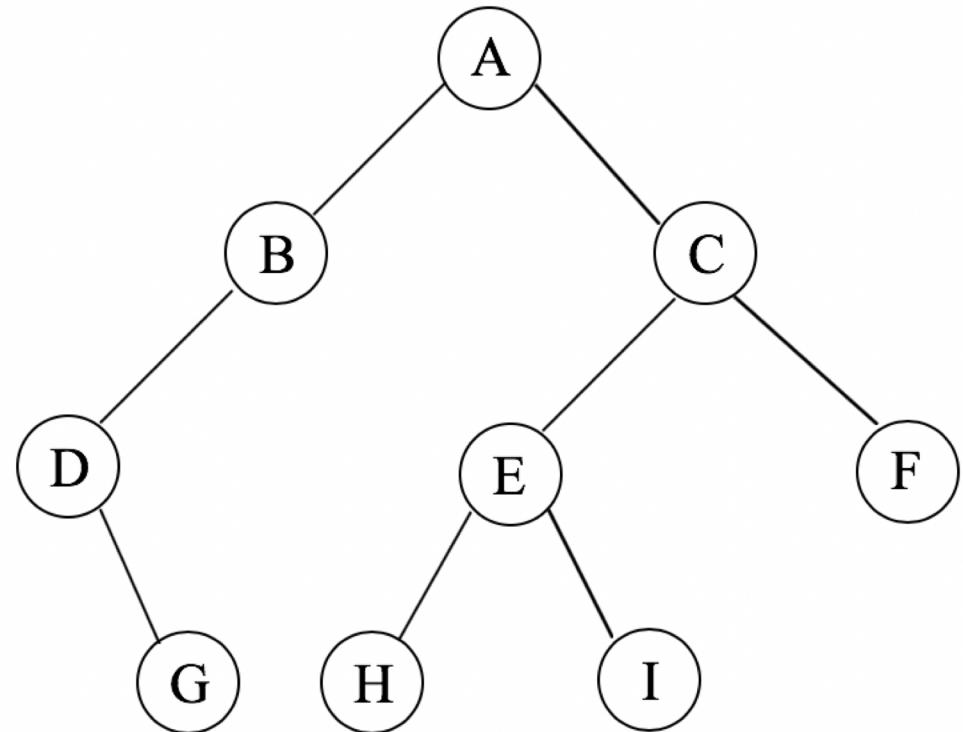


# Postorder: İkili ağaçlarda dolaşma

- **Postorder dolaşma**

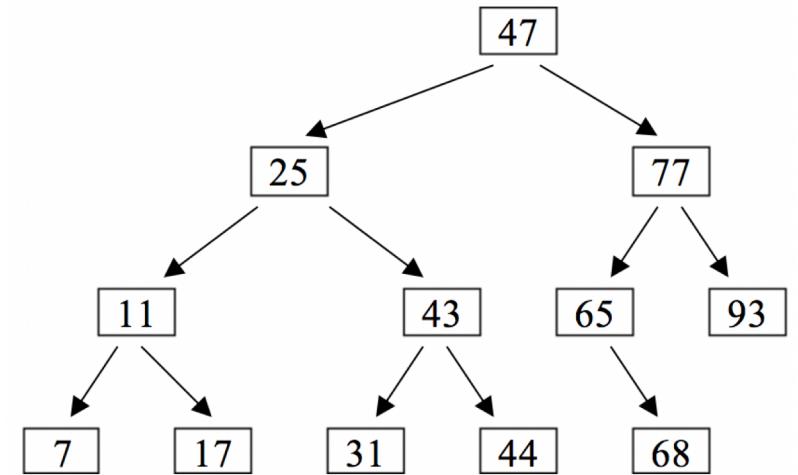
- Sol alt ağacı postorder'a göre dolaş
- Sağ alt ağacı postorder'a göre dolaş.
- Köke uğra (visit)

**Postorder:** GDBHIEFCA



# İkili arama ağaçları

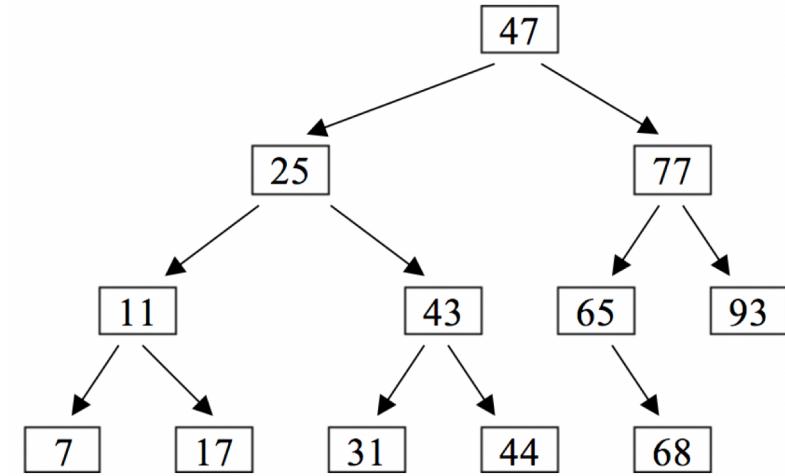
- Sol alt ağaç kökten küçüktür, sağ alt ağaç kökten büyüktür.
- $n$  düğümlü ise, en fazla  $\log_2 n$  düzey vardır.



- 1000 elemanlı ikili ağaçta en fazla kaç karşılaştırırmaya yapılır.?

# İkili arama ağacı oluşturma

- Sırasız bir sayı dizisi geldiğinde
  - İlk sayı **root** olarak eklenir.
  - Diğerleri karşılaştırma yapılarak eklenir.
  - Sol alt ağaçtaki tüm düğümler kendi atasından küçük, sağ alt ağaçtaki tüm düğümler kendi atasından büyük olmalıdır.

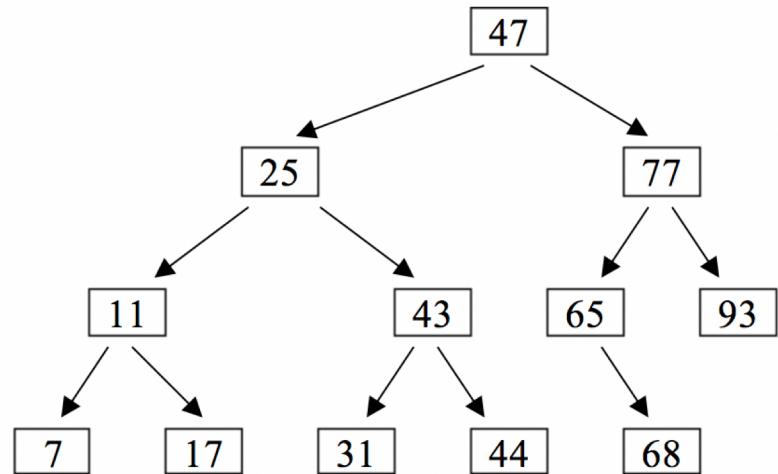


Örnek => 47, 25, 43, 77, 65, 68, 93, 11, 17, 44, 31, 7

# İkili arama ağacında dolaşma

- İkili arama ağacını *inorder* olarak dolaşın?

Cevap?



# Diğer

- İkili ağaç veri modeli
- Düğüm ekleme
- Düğüm silme
- Dolaşma
- Ağacı yazdırma/görüntüleme
- Ağacı dolaşma
- Derinliğini hesaplama