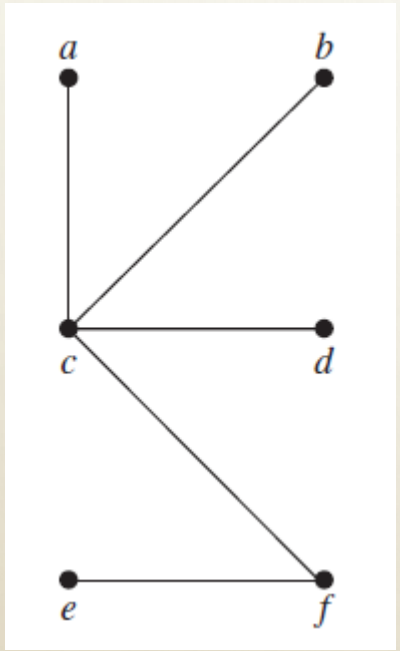
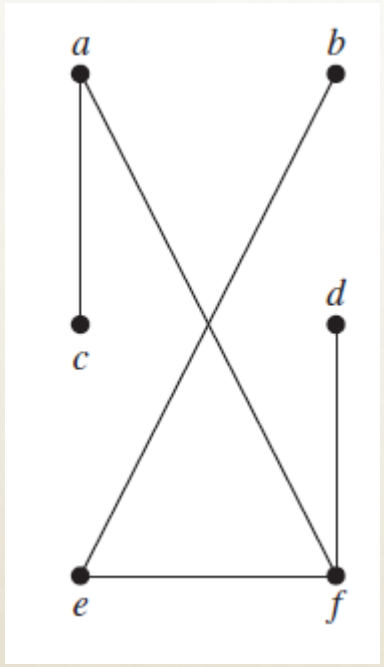
AĞAÇLAR

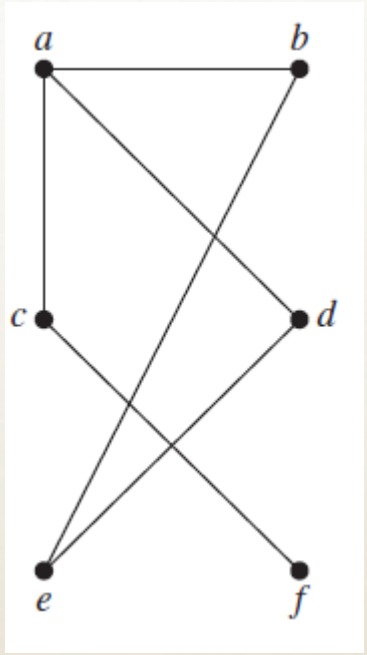
DR. ZEYNEP BANU ÖZGER

AĞAÇ

- İçerisinde çevrim bulundurmayan bağlı (connected) ve yönsüz graflara ağaç denir.

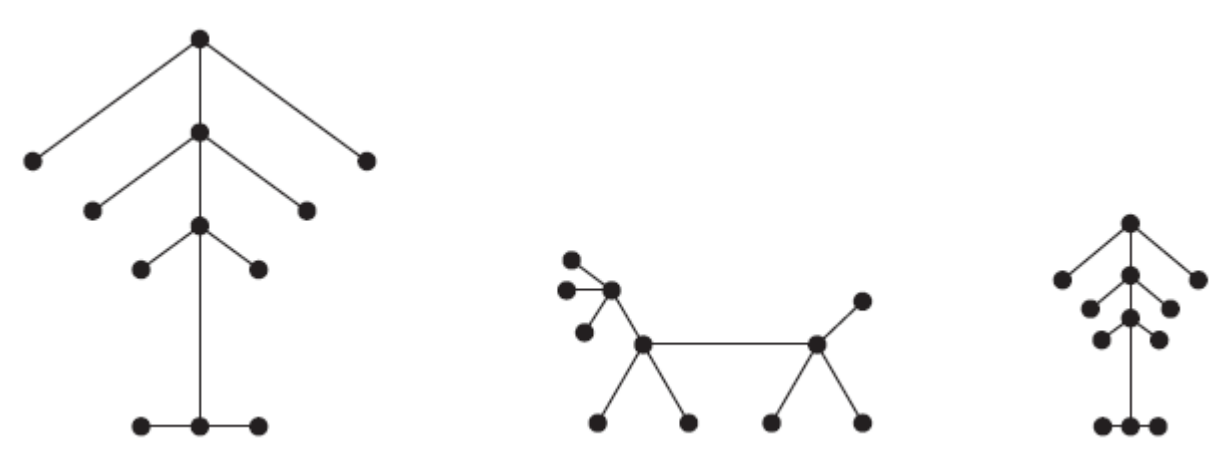






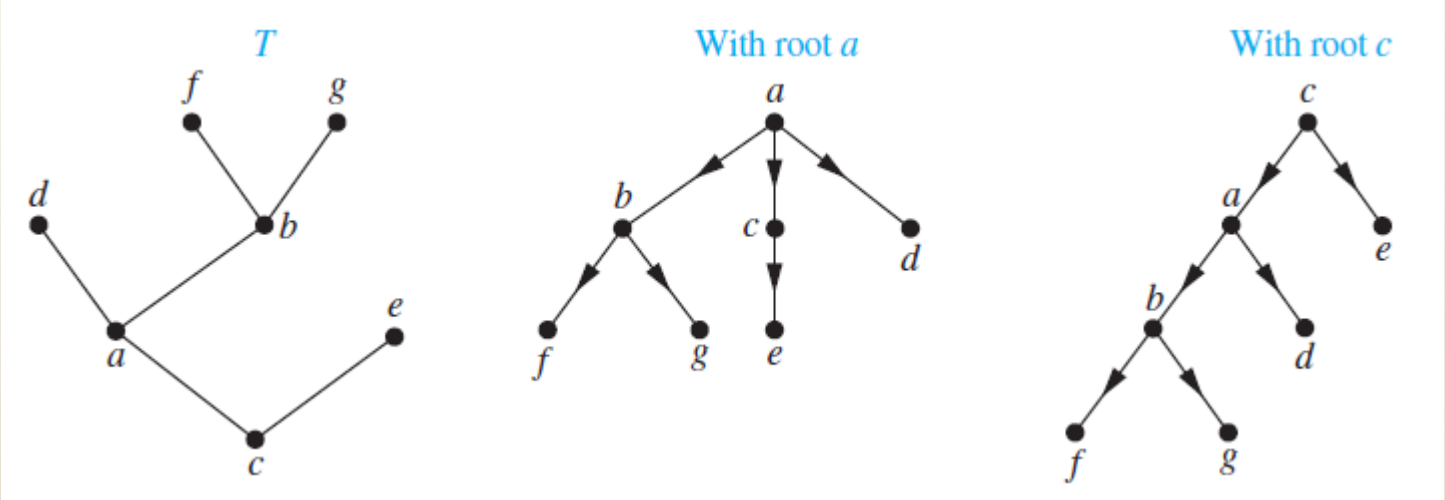
AĞAÇ

- Bir graf bağlı değil ancak kapalı çevrim içermiyorsa bu grafın her bir parçası birer ağaçtır.



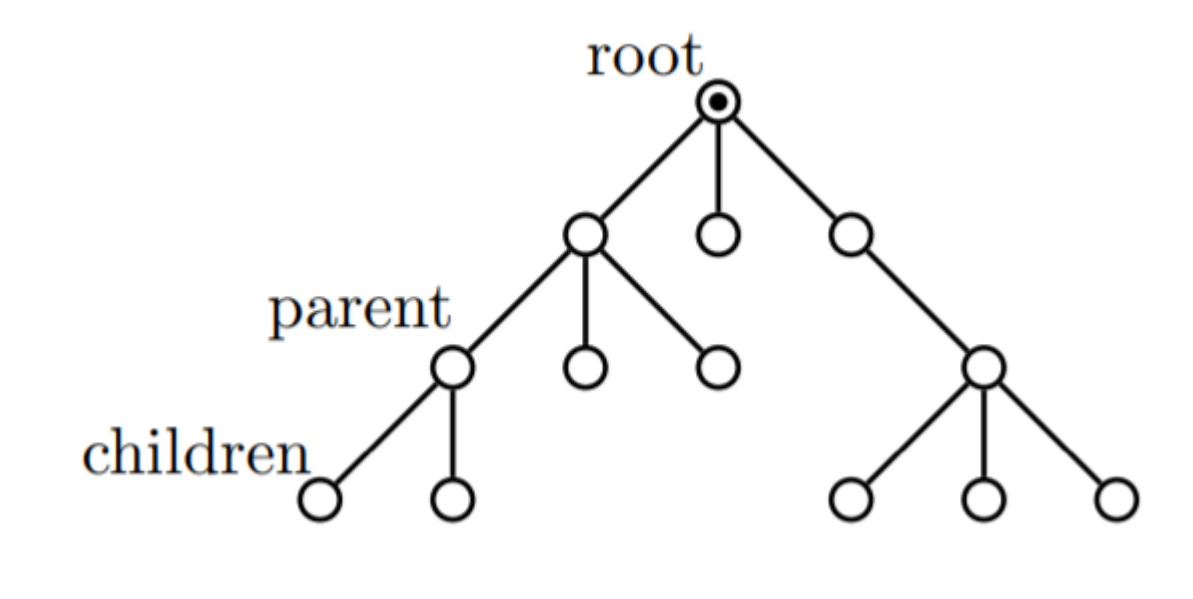
Köklü Ağaç

- Graftaki düğümlerden birinin kök olarak belirlendiği ağaçlara **köklü ağaç (rooted tree)** denir.



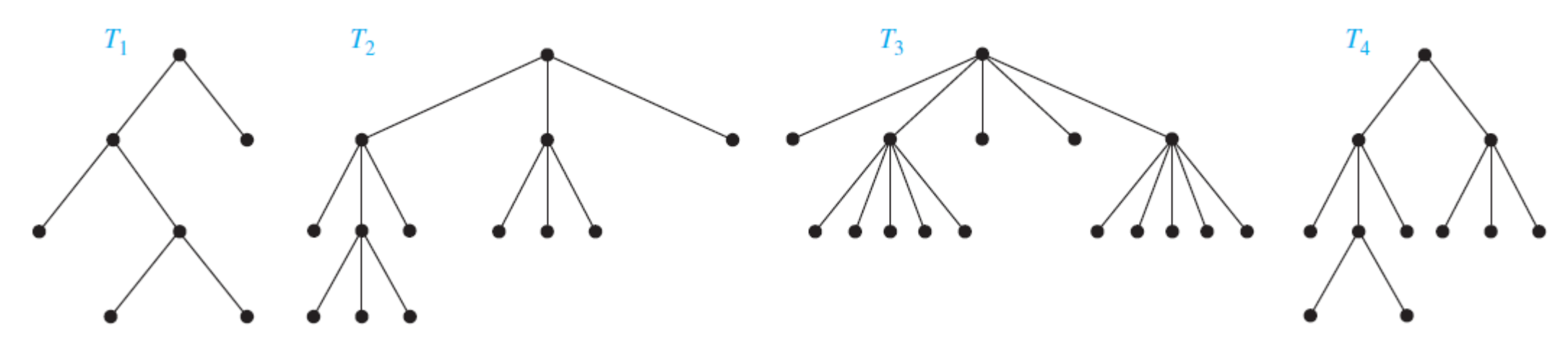
Köklü Ağaç Terminolojisi

- Kök (Root);En tepedeki düğüm.
- Yaprak (leaf); Çocuğu olmayan, ağacın en alt seviyesindeki düğümlerdir.
- İç Düğüm; Yaprak olmayan, en az bir çocuğu olan düğümlere iç düğüm denir.
- Ebeveyn- üst öğe (Parent); Bir düğümün, bir kenar ile bağlı olduğu üst düğümüdür.
- Çocuk-alt düğüm (child); Bir düğümün bir kenar ile kendisine bağlı olan tüm alt düğümleri.
- Kardeş (sibling); Aynı üst düğüme bağlı düğümlerdir.
- Ata (Ancestor); Bir düğümün bir yol ile bağlı olduğu ve kendisinden üst seviyede olan düğümler
- Torun (Descendant); Bir düğümün bir yol ile bağlı olduğu ve kendisinden alt seviyede olan düğümler. Bir düğümün atası olduğu tüm düğümler kendisinin torunudur.
- Alt ağaç (sub-graph); Kök olmayan bir a düğümü, torunlarıyla birlikte bir alt ağaçtır. a düğümü de alt ağacın köküdür.



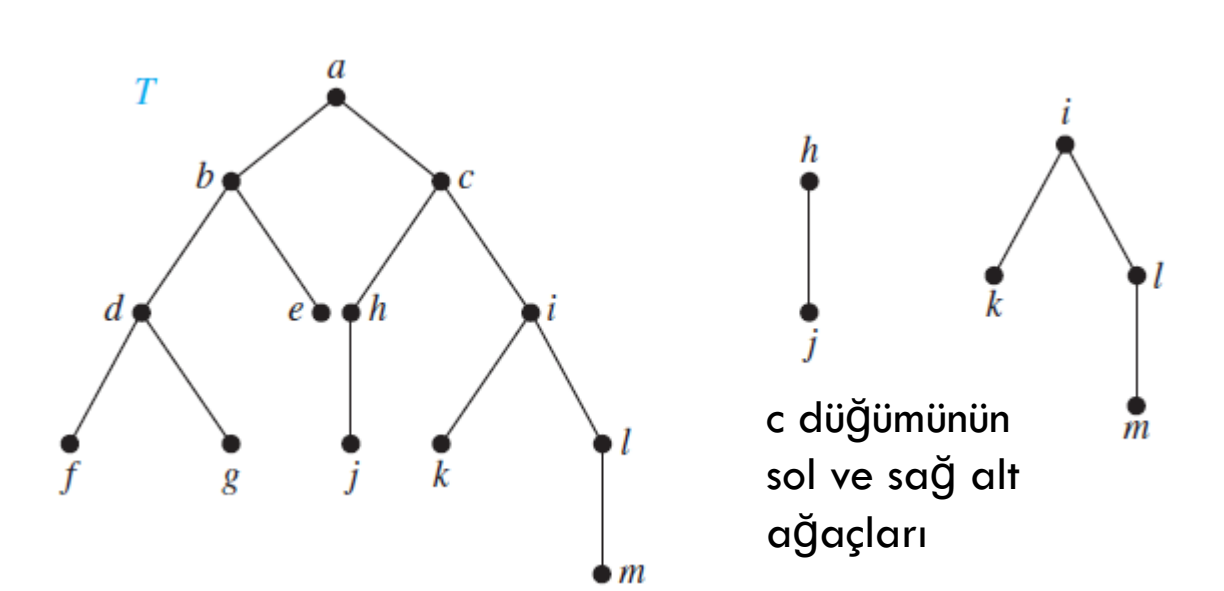
M-Ary Rooted Tree

- Köklü bir ağaçta her bir iç düğümün en fazla m çocuğu var ise buna **m-ary tree** denir.
- Tüm iç düğümlerin m tane çocuğu varsa buna da **full m-ary tree** denir.



Sıralı Köklü Ağaç

- Tüm iç düğümlerin çocuklarının sıralı olduğu köklü ağaçlara sıralı köklü ağaç (sıralı rooted tree) denir.
- Bir düğümün sol çocuğunun kökü olduğu at ağaca **sol alt ağaç (left sub tree),** sağ çocuğunun kökü olduğu alt ağaca **sağ alt ağaç (right sub tree)** denir.



Ağaçların Özellikleri

n düğüm içeren bir ağacın n-1 tane kenarı vardır.

i tane iç düğüm içeren bir full m-ary ağacında

- toplam düğüm sayısı: n=mi+1

n tane düğüm içeren bir full m-ary ağaçta

- $iç\ düğüm\ sayısı = (n-1)/m$ ve
- $yaprak\ sayısı = [(m-1)n+1]/m$ ile hesaplanır.

h yüksekliğinde bir m-ary ağaçta;

- En fazla m^h adet yaprak olabilir.

Bir v düğümünün seviyesi,

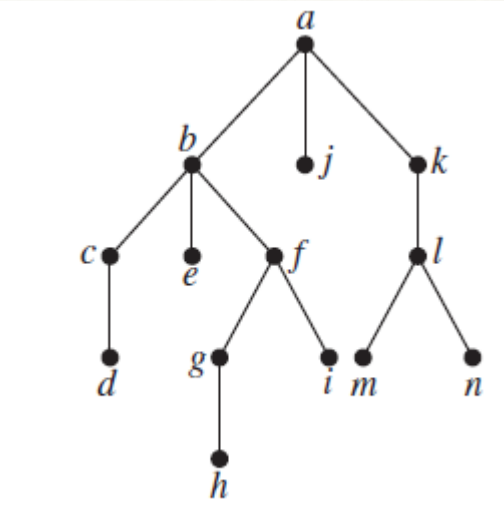
- Kökten v düğümüne kadar olan benzersiz kenar sayısıdır.

Bir köklü ağacın yüksekliği;

- Ağaçta en yüksek seviyeli düğümün seviyesidir.

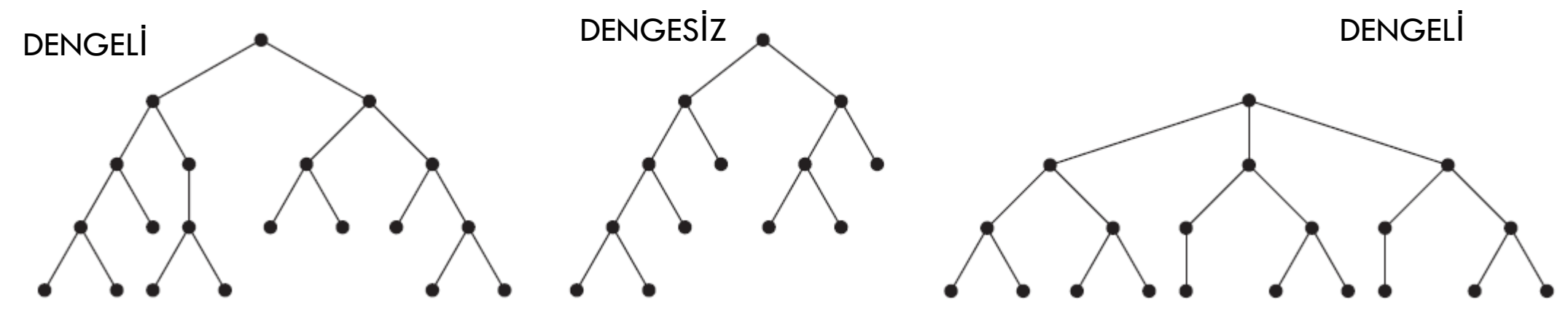
Ağaçların Özellikleri

- **Örnek;**
  - Kök düğümün seviyesi:0
  - b, j ve k düğümlerinin seviyesi:1
  - En yüksek seviyeli düğüm: h
  - h düğümünün seviyesi: 4

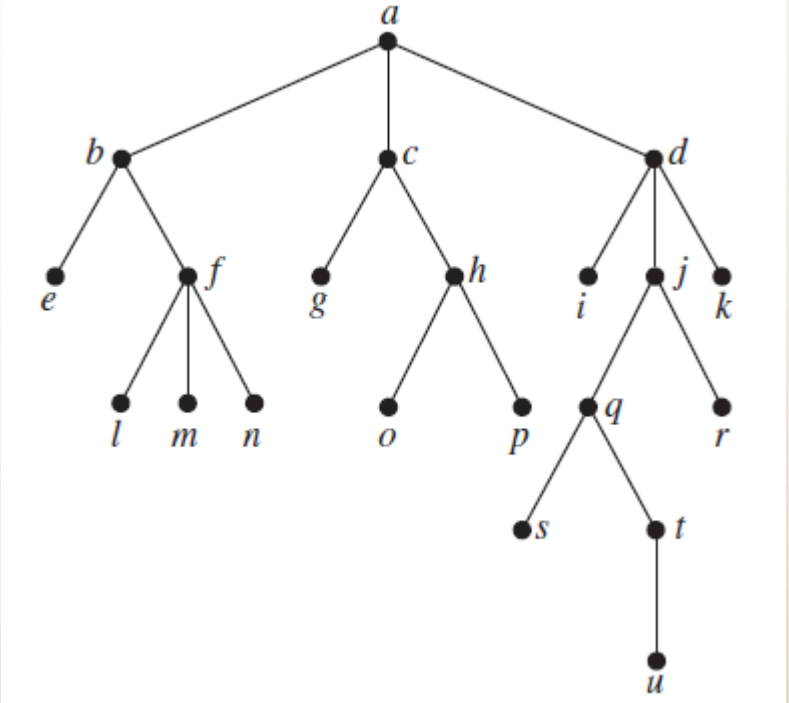


Dengeli m-ary Ağaç

- h yüksekliğinde köklü bir m-ary ağacı, tüm yaprakların seviyesi h veya h-1 ise dengeli (balanced) ağaçtır.



- Örnek:
  - Root:
    - a
  - İç düğümler:
    - a,b,c,d,f,h,j,q,t
  - Yapraklar:
    - e,l,m,n,g,o,p,i,s,u,t,r,k
  - j‘nin çocukları:
    - q ve r
  - h‘nin parentı:
    - c
  - o’nun kardeşleri:
    - p
  - m‘nin ataları:
    - f, b, a
  - b’nin torunları:
    - e, f, m, l, n



AĞAÇ UYGULAMALARI

İkili Arama Ağacı (Binary Search Tree)

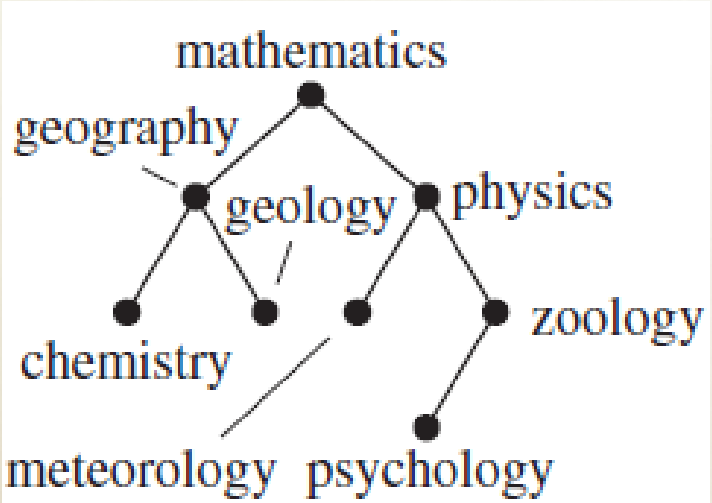
- **İkili Arama ağacı;**
  - Sıralı bir listede aranan bir elemanın bulunması için kullanılır.
  - Her iç düğümün en fazla 2 çocuğu olabilir.
  - Her bir düğüm bir anahtar ile etiketlenir.
  - Bir düğümün anahtarı sol alt ağaçtaki tüm düğümlerin anahtar değerlerinden büyük, sağ alt ağaçtaki tüm düğümlerin anahtar değerlerinden küçük olacak şekilde düğümlere anahtar atanır.

İkili Arama Ağacı (Binary Search Tree)

- **Bir İkili Arama Ağacı Oluştururken;**
  - Bir düğüm ile başlanır.
  - Listedeki ilk eleman kökün anahtar değeri olur.
  - Listedeki bir sonraki eleman kökün anahtar değeri ile
    - Aynı ise listede bir sonraki değer geçilir.
    - Büyükse, sağ alt ağaç boyunca yeni düğümün değeri sıradaki düğümün değerinden büyük olduğu sürece bir alt düğüme geçilerek ilerlenir.
    - Küçükse sol alt ağaç boyunca yeni düğümün değeri sıradaki düğümün değerinden küçük olduğu sürece bir alt düğüme geçilir.
  - Bu işlem rekürsif olarak tüm elemanlar uygun yere eklenene kadar devam eder.

İkili Arama Ağacı

- Örnek; Mathematics, physics, geography, zoology, meteorology, geloji, psychology ve chemistry kelimeleri alfabetik sırada olacak şekilde bir ikili arama ağacı oluşturun.

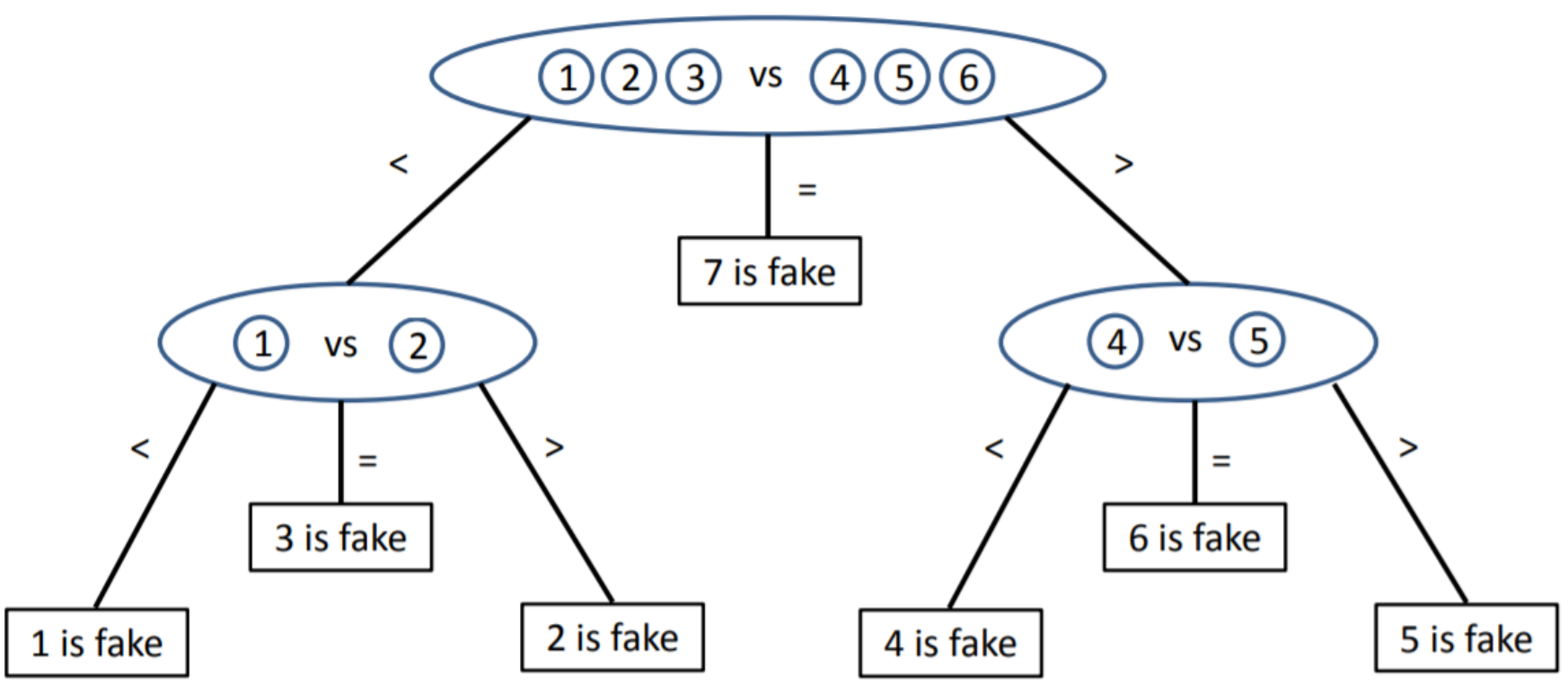


Karar Ağaçları (Decision Tree)

- Bir problemin olası çözümlerine karar vermek için kullanılan köklü bir ağaçtır.
- İç düğümler: Karar,
- Alt öğeler o kararın sonuçları.

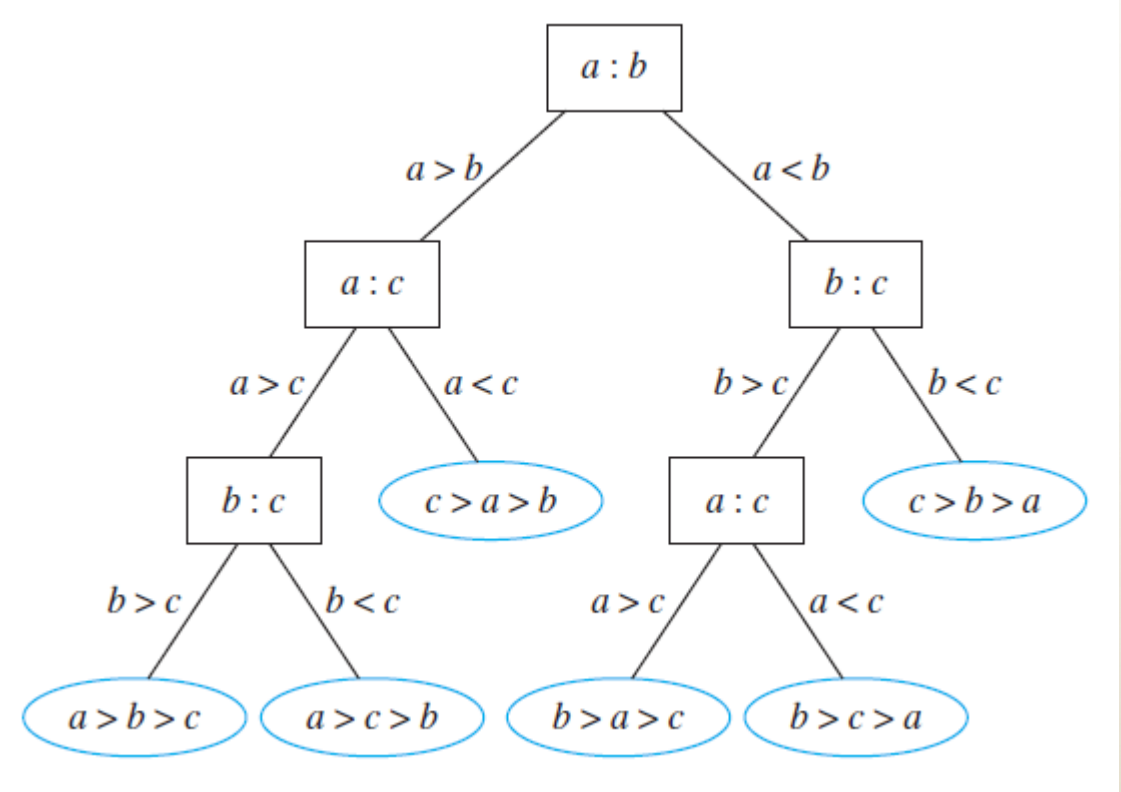
Karar Ağaçları (Decision Tree)

- **Örneğin;** 7 tane bozuk paradan 1 tanesi sahte. Sahte olmayan 6 bozuk paranın ağırlığı eşitken, sahte paranın ağırlığı daha az. Buna göre 7 bozuk paradan hangisinin sahte olduğunu karar ağacıyla modellersek;



Karar Ağaçları (Decision Tree)

- **Örneğin;** a, b ve c sayılarını sıralamak istiyoruz

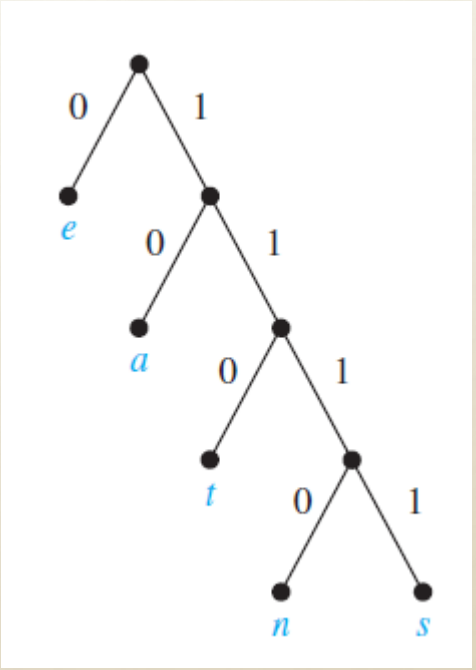


Önek Kodları (Prefix Codes)

- Bir alfabedeki tüm harfleri bir bit dizisi ile kodlamak istiyoruz.
- 26 harflik bir alfabe için 5 bit uzunluğunda bit dizileri.
- Ancak birkaç harften oluşan bir bit dizisinin harf karşılıklarını bulmak istediğimizde
  - Örneğin; e:0, a:1 ve t: 01 ile kodlanmış olsun.
  - 0101 bit dizisi eat, tea, tt, eaea kelimelerinden hangisi olduğunu nasıl ayırt edebiliriz?
- Bir alfabedeki tüm harflerin farklı bit dizisi ile kodlamak için
  - Her harfin bit dizinin ilk parçasının birbirinden farklı olacak şekilde kodlanmasına önek kodları (prefix codes) denir.

Önek Kodları (Prefix Codes)

- Yapraklar karakterler.
- Sol çocuk 0 sağ çocuk 1.
- Bir karakterin bit dizisi, kökten ilgili karakterin bulunduğu düğüme kadar ki yoldaki bitlerin arka arkaya yazılması ile elde edilir.
- Örneğin:
  - e=0, a=10, t=110, n=1110 ve s=11111
  - Decode edilecek bit dizimiz:11111011100 olsun
  - Kökten başlanır ve bir yaprağa ulaşana kadar bitlere göre kenarlar boyunca ilerlenir.
    - 1111 s yaprağına geldi ilk karakter:s
    - 10: a yaprağına geldi 2. karakter:a
    - 1110 n yaprağına geldi 3. karakter:n
    - 0 e yaprağına geldi 4. karakter:e
    - Kelime:sane

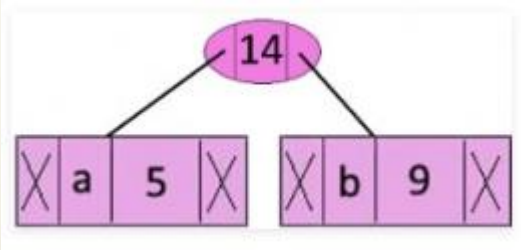


Huffman Kodlama

- Bir önek ağacıdır.
- Bir karakterin frekansı veya olasılığı ne kadar yüksekse o kadar az bitle kodlanır.
- Veri sıkıştırmanın temel algoritmalarındandır.
- Karakterlerin geçme sıklıkları veya olasılıkları bilinmelidir.

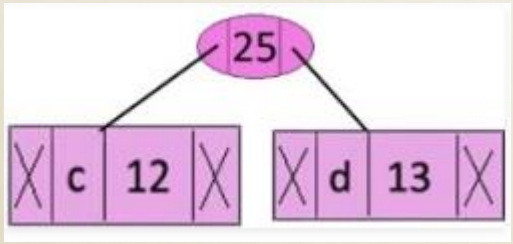
Huffman Kodlama

- **Örnek:** Harflerin geçme sıklıkları a:5, b:9, c:12, d:13, e:16 ve f:45 olsun
  - Min frekanslı 2 karakter: a ve b



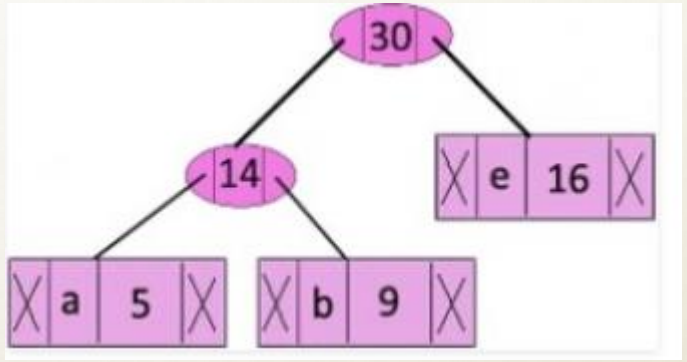
Tekrar sıralanır:

c:12, d:13, ab:14, e:16 ve f:45



Tekrar sıralanır:

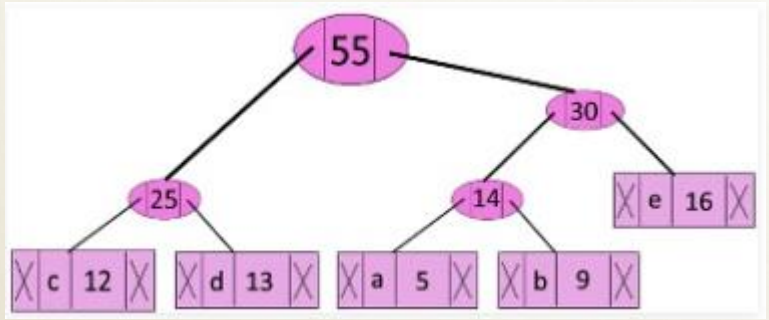
ab:14, e:16, cd:25, f:45



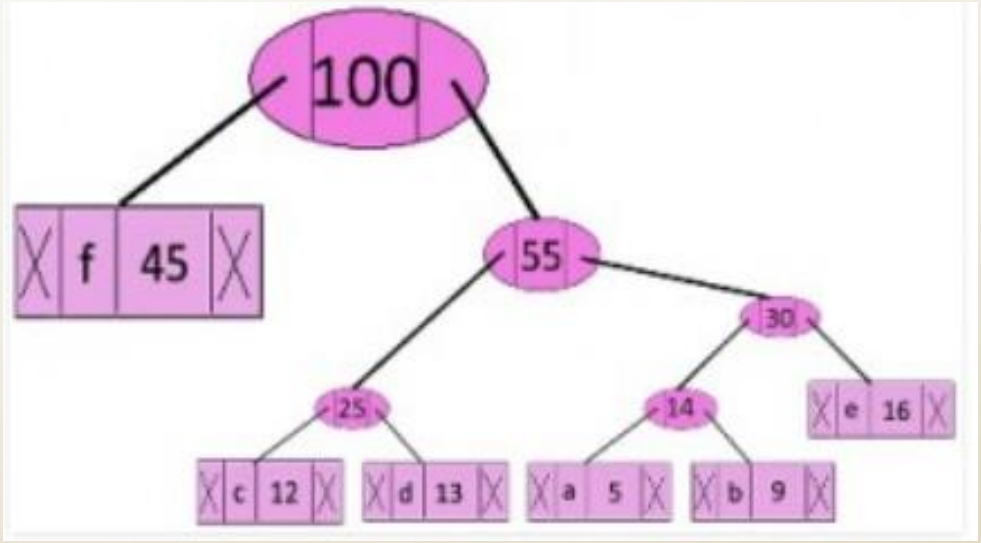
Huffman Kodlama

- **Örnek:** Harflerin geçme sıklıkları a:5, b:9, c:12, d:13, e:16 ve f:45 olsun

Tekrar sıralanır:
cd:25, abe:30, f:45

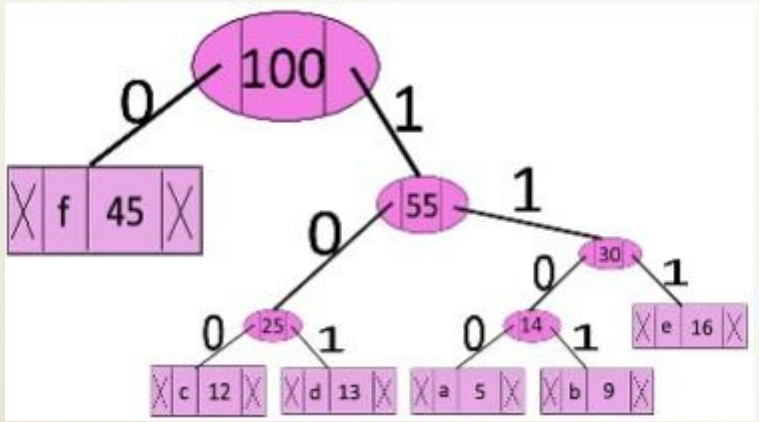


Tekrar sıralanır:
f:45, abecd:55



Huffman Kodlama

- **Örnek:** Harflerin geçme sıklıkları a:5, b:9, c:12, d:13, e:16 ve f:45 olsun



f:0
c:100
d:101
a:1100
b:1101
e:111

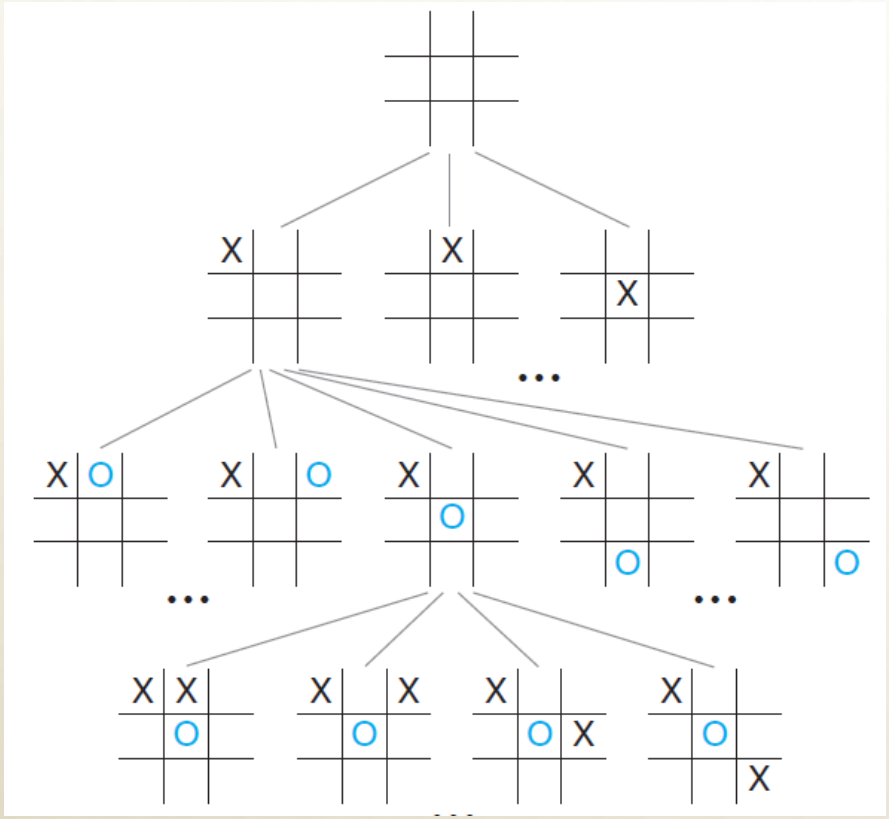
- Her bir karakteri 8 bit ile temsil etseydik;
  - 5+9+12+13+16+45=100 karakter
  - 100*8=800 bit kullanmamız gerekirdi.
  - Bu yöntemle 5*4+9*4+12*3+13*3+16*3+45*1=324 bit ile kodlayabiliriz

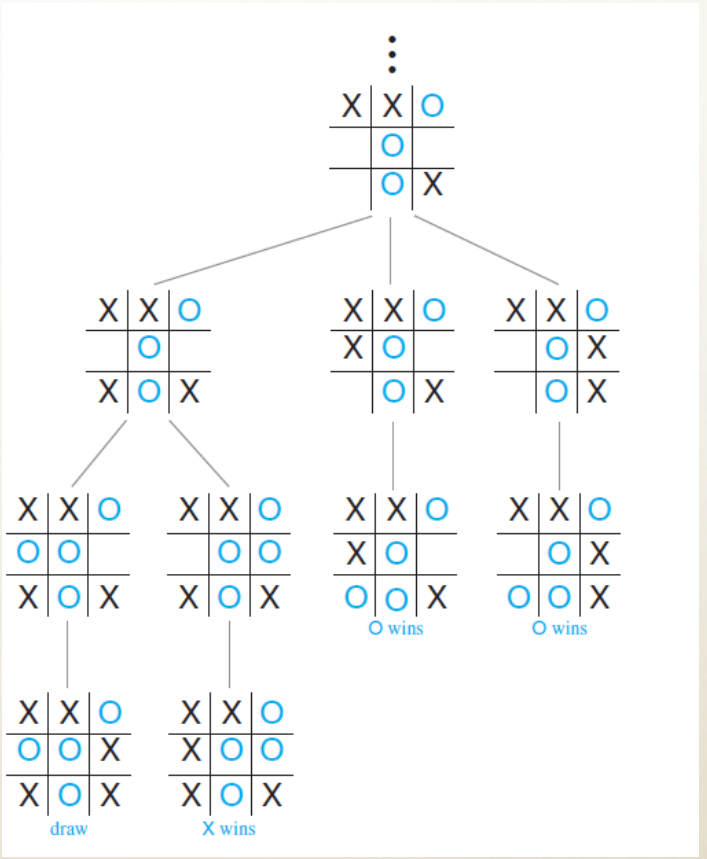
Oyun Ağacı

- Ağaçlar, tic-tac-toe, satranç, dama gibi belirli oyunları analiz etmek için de kullanılır.
- Düğümler pozisyonları, kenarlar ise olası hamleleri temsil eder.
- Kök; başlangıç pozisyonu.
- Yapraklar; oyunun son durumları.

Oyun Ağacı

- **Örnek:** Tic-Tac-Toe

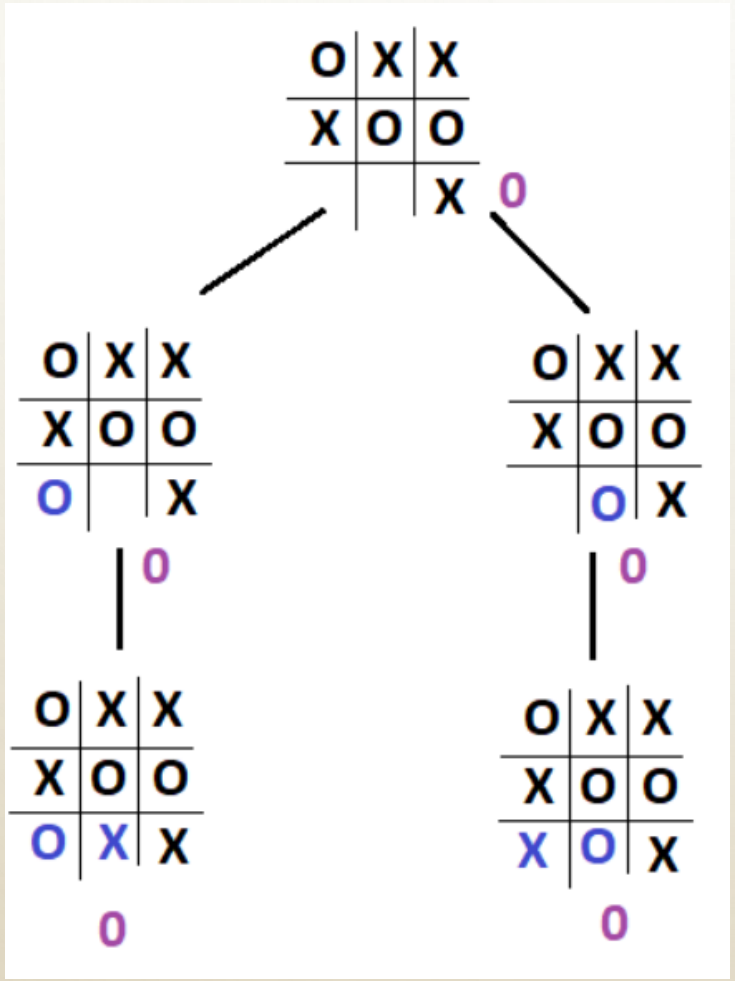




ÖRNEK

Tic-Tac-Toe oyununda anlık durumu aşağıdaki gibi ise, oyunun sonraki hamleler için ağacı nasıl olur?

O	X	X
X	O	O
		X



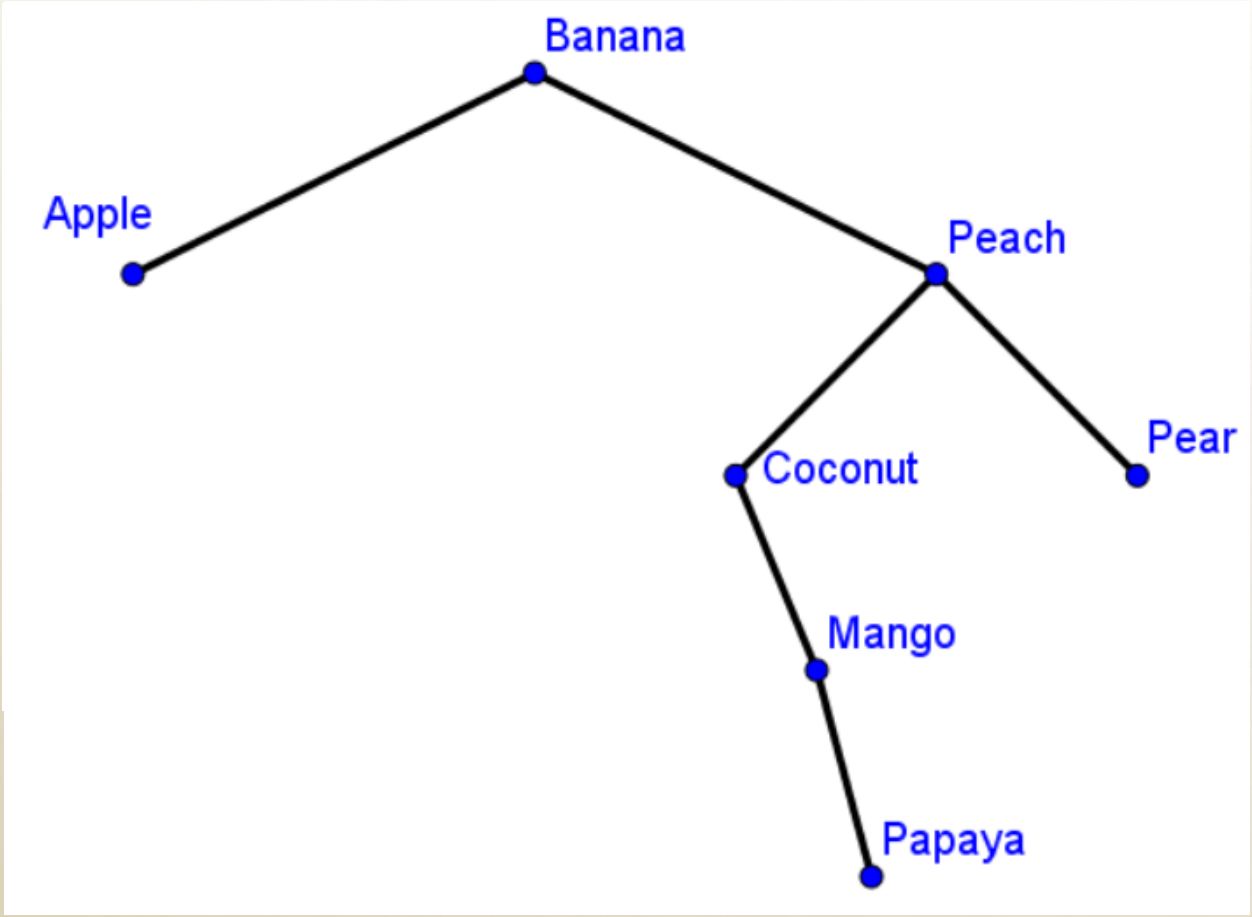
ÖRNEK

- **Aşağıda kodları verilen harfler için bit dizilerini decode edin.**
  - a:001, b:0001, r:0000, s:0100, t:011, x:01010, e:1

1. **01110100011→**
   - **t(011),**
   - **e(1),**
   - **s(0100),**
   - **t(011)**
2. **01100101010 →**
   - **t+00101010,**
   - **ta+01010**
   - **tax**

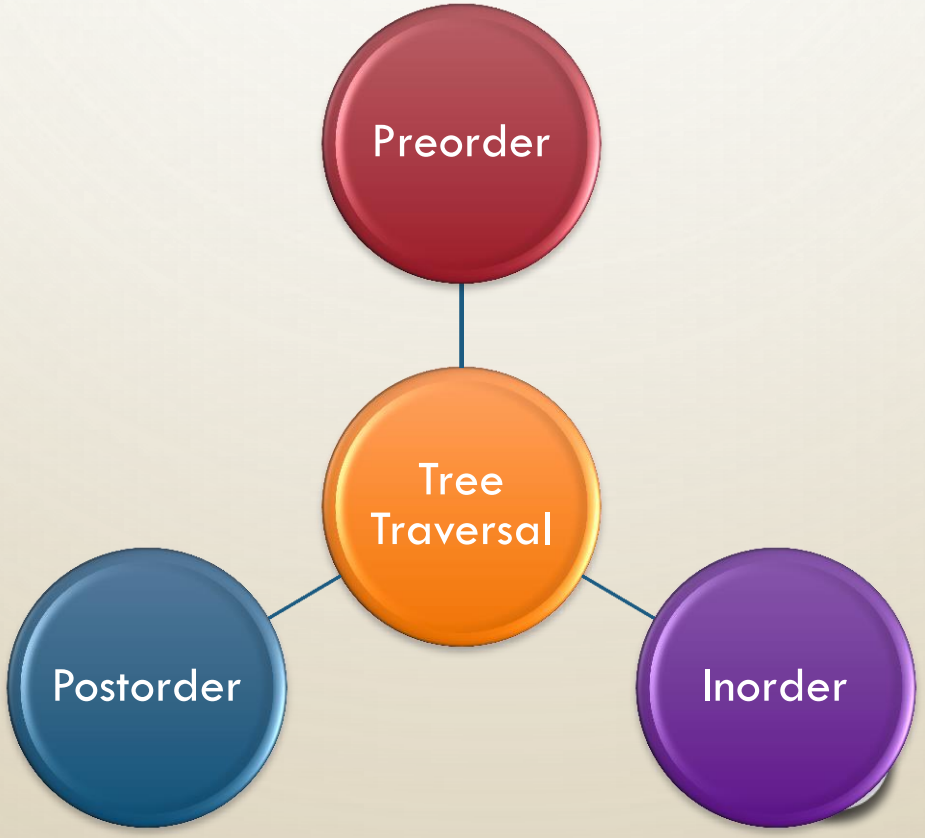
ÖRNEK

- **Aşağıdaki kelimeleri bir binary ağaca alfabetik sırada yerleştirin.**
  - banana, peach, apple, pear, coconut, mango, and papaya



AĞAÇTA DOLAŞMA

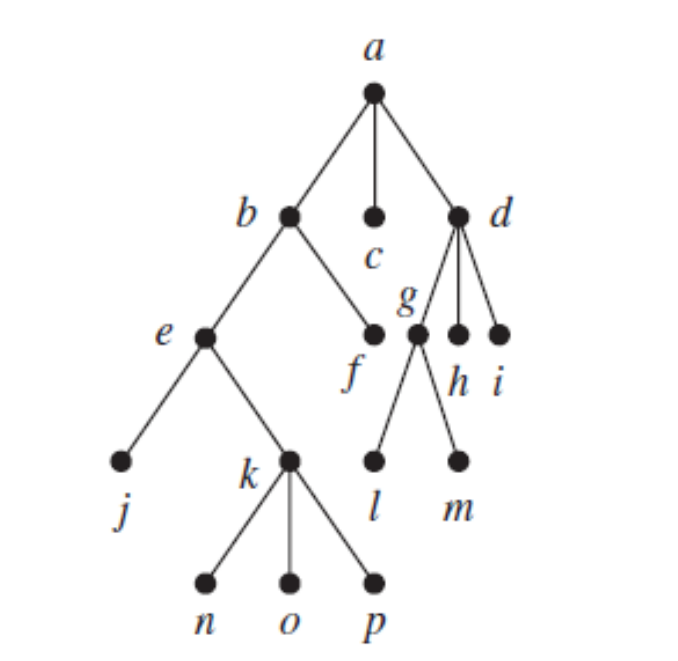
- Sıralı köklü ağaçlar genellikle bilgi depolamak için kullanılır.
- Ağaçtaki verilere erişmek için ağaçta dolanımı sağlayacak prosedürler vardır.
- Sıralı bir köklü ağacın her köşesini sistematik olarak ziyaret etme prosedürlerine geçiş algoritmaları (traversel algorithms) denir.



Preorder

- Kök, sol düğüm, sağ düğüm
- Rekürsif yapıdadır.

Preorder



a,

b,

e,

j,

k,

n,

o,

p,

f,

c,

d,

g,

l,

m,

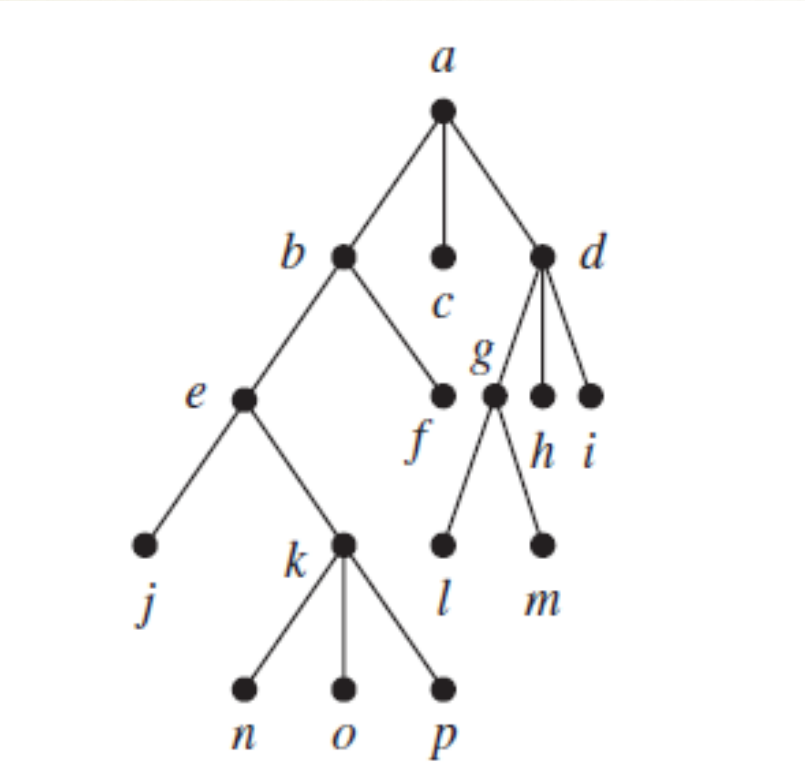
h,

i

Inorder

- Sol düğüm, kök, sağ düğüm
- Sol alt ağacı left, root, right sırasıyla rekürsif olarak dolaş
- Kökü ziyaret et
- Sağ alt ağacı left, root, right sırasıyla rekürsif olarak dolaş

Inorder



j,

e,

n,

k,

o,

p,

b,

f,

a,

c,

l,

g,

m,

d,

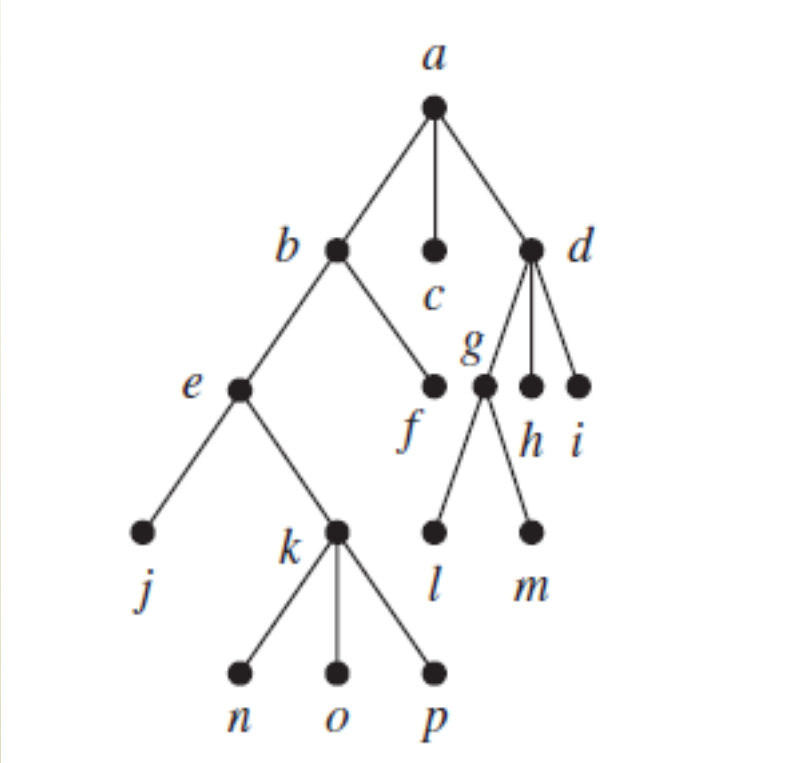
h,

i

Postorder

- Sol düğüm, sağ düğüm, kök
- Sol alt ağacı dolaş
- Sağ alt ağacı dolaş
- En son köke git

Postorder

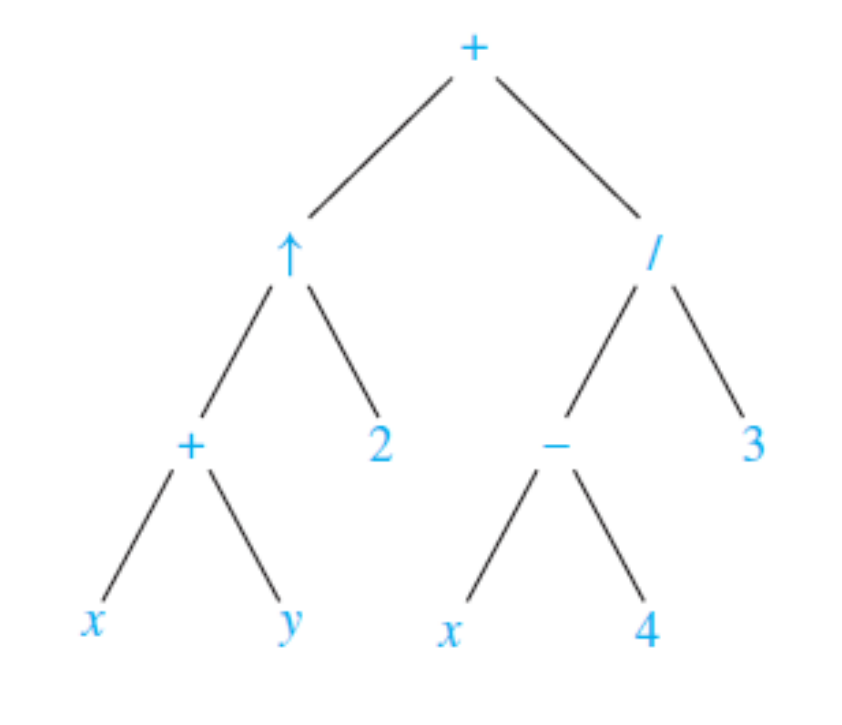


- **j, n, o, p, k, e, f, b, c, l, m, g, h, i, d, a**

Infix, Prefix Postfix

- Aritmetik ifadelerin ağaçtan okunma düzeni
- Infix; inorder
- Prefix; preorder
- Postfix; post order

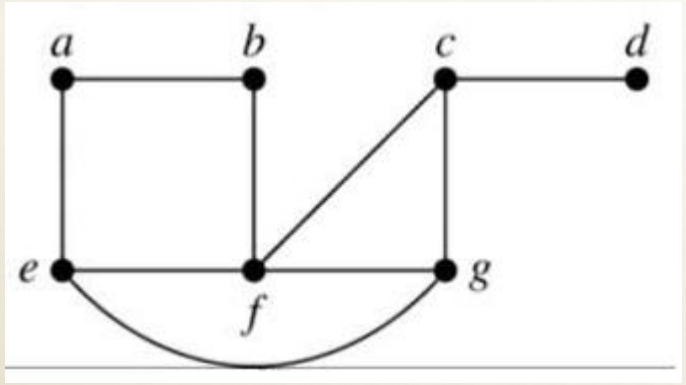
Infix, Prefix Postfix



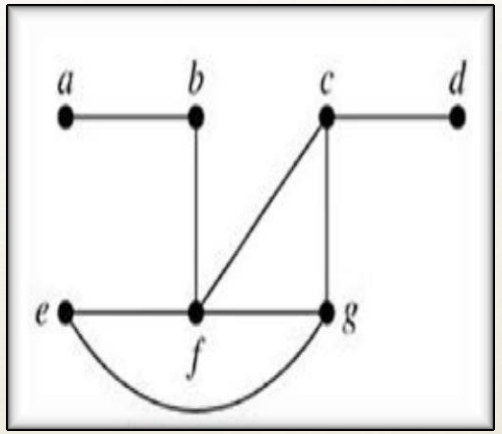
- **Infix:**
  - **Inorder:** Left, root, right
  - ((x+y)^2)+((x-4)/3)
- **Prefix:**
  - **Preorder:** Root, left, right
  - +^+xy2/-x43
- **Postfix:**
  - **Postorder:** Left, right, root
  - xy+2^x4-3/+

TARAMA AĞACI (SPANNING TREE)

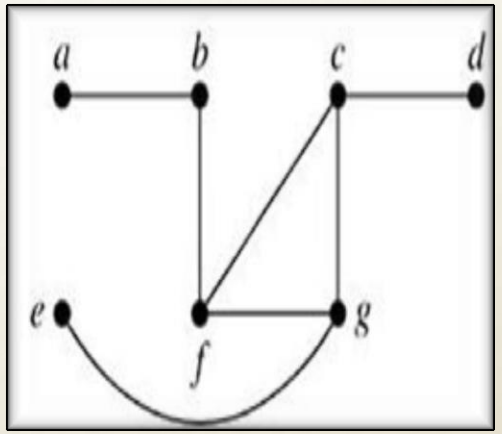
- G basit bir graf olmak üzere bir tarama ağacı, G grafının tüm düğümlerini içeren bir alt grafıdır.
- Bir grafın birden fazla tarama ağacı olabilir.



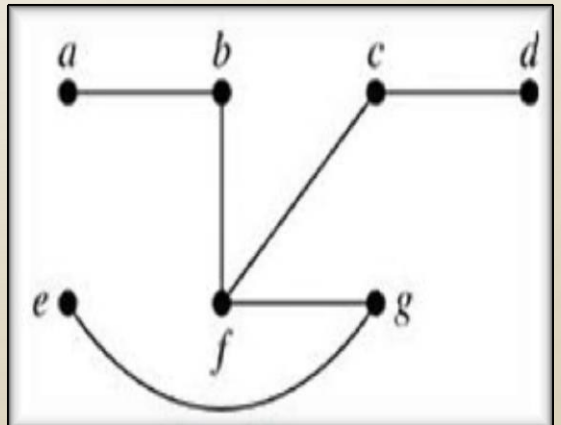
(a,e) kenarı çıkarıldı



(e,f) kenarı çıkarıldı

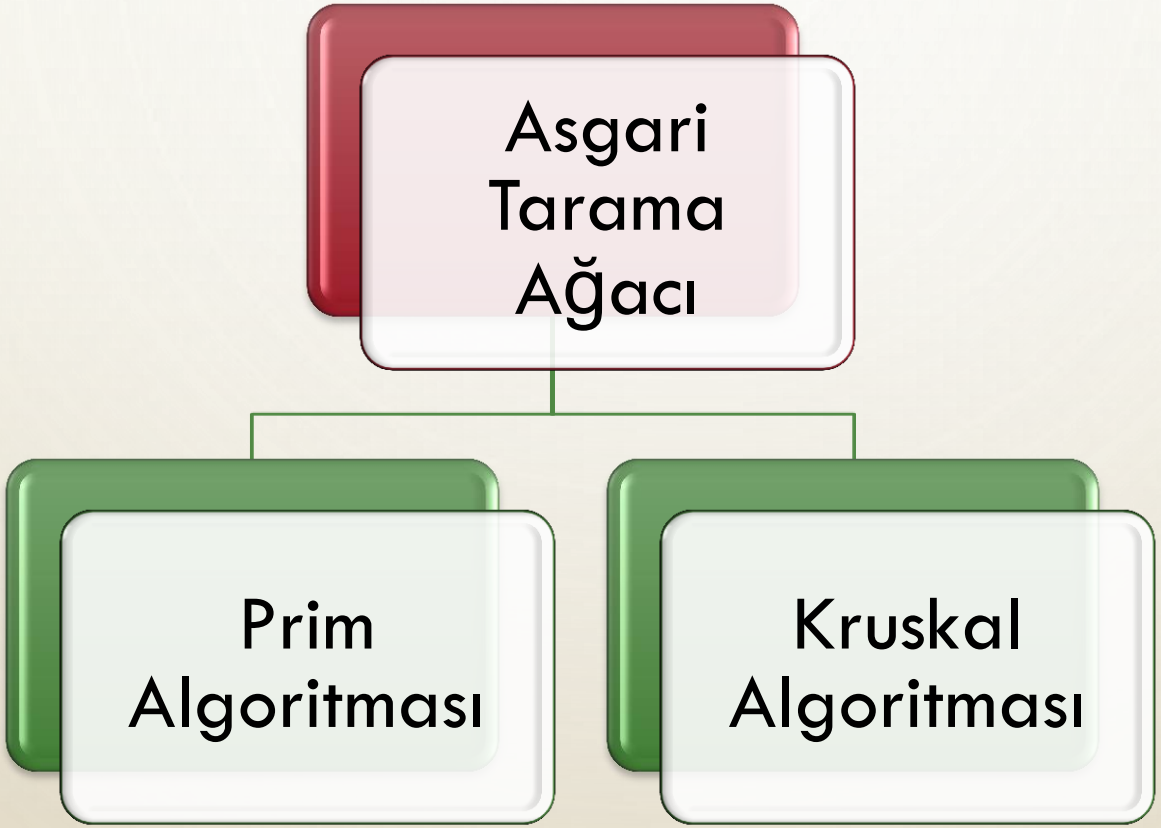


(c,g) kenarı çıkarıldı



ASGARİ TARAMA AĞACI (MINIMUM SPANNING TREE)

- Bir G ağırlıklı grafında, kenarların ağırlıkları toplamı minimum olan tarama ağacıdır.

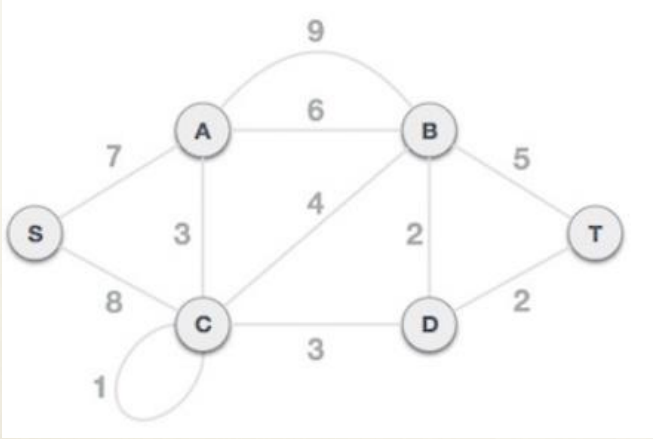


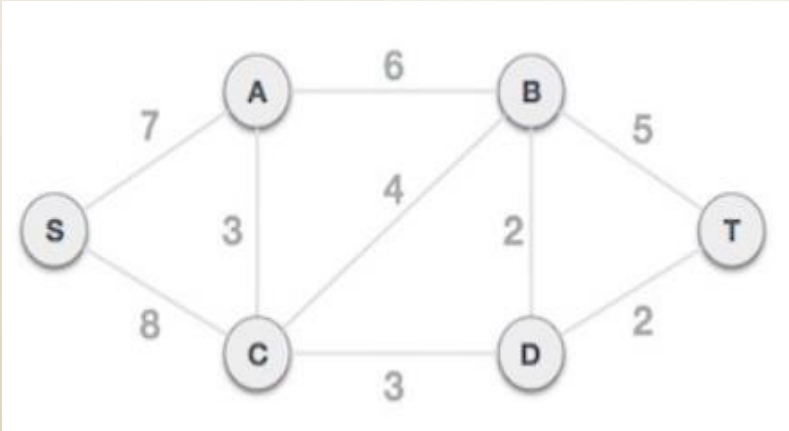
Kruskal Algoritması

- Her bir düğümün bireysel bir ağaç olduğunu varsayarak başlar.
- Her adımda en az maliyetin olduğu kenarı ekler.
- Bu işlem tüm düğümleri içeren tek bir yol olmasına kadar devam eder.

Kruskal Algoritması

- **ÖRNEK;**



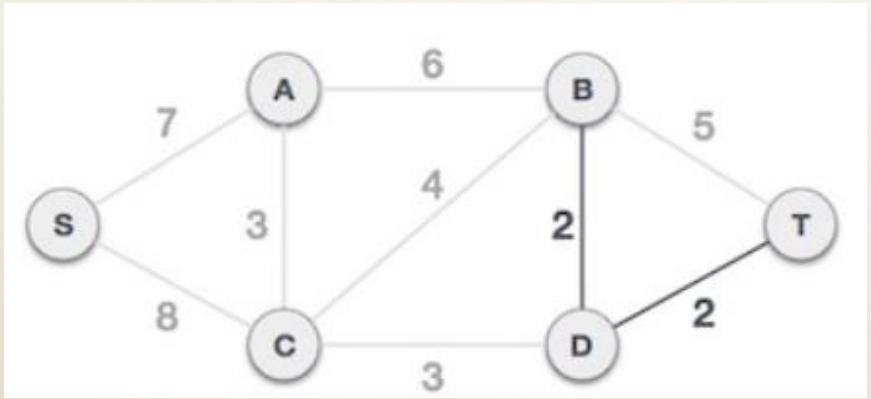


Kruskal Algoritması

- **ÖRNEK;**

B, D	D, T	A, C	C, D	C, B	B, T	A, B	S, A	S, C
2	2	3	3	4	5	6	7	8

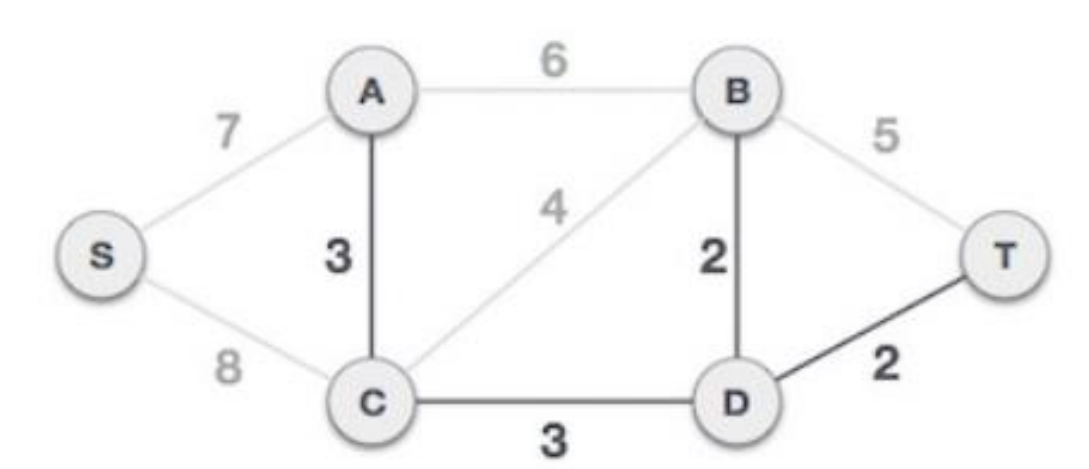
- Kenarlar ağırlıklarına göre artan sırada sıralanır.



Kruskal Algoritması

- **ÖRNEK;**

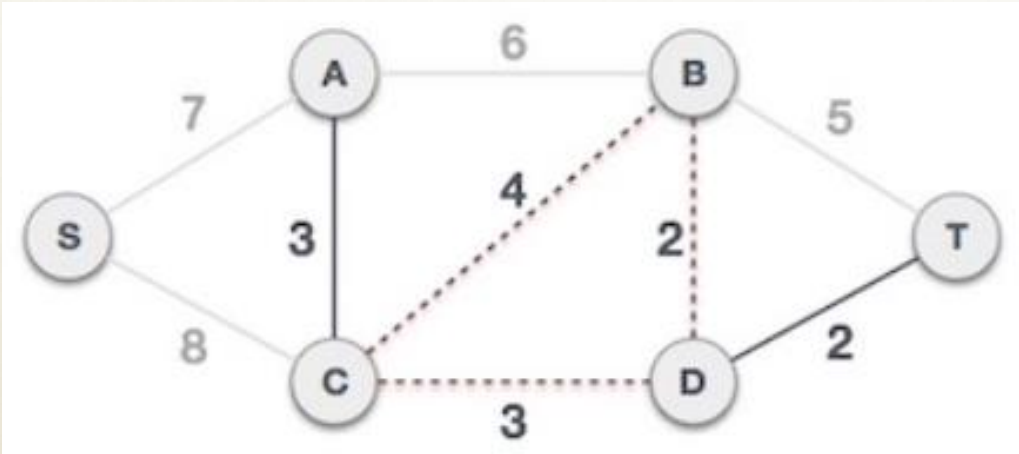
B, D	D, T	A, C	C, D	C, B	B, T	A, B	S, A	S, C
2	2	3	3	4	5	6	7	8



Kruskal Algoritması

- **ÖRNEK;**

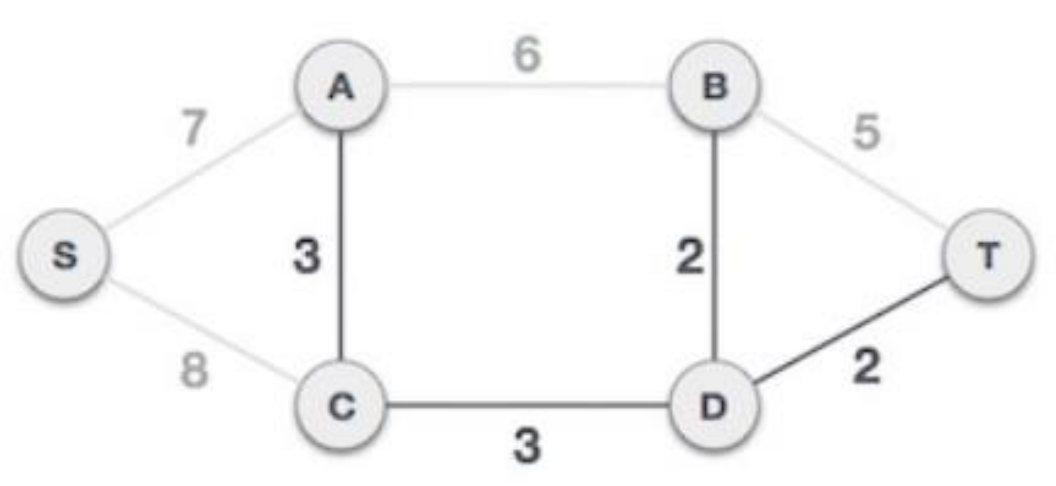
B, D	D, T	A, C	C, D	C, B	B, T	A, B	S, A	S, C
2	2	3	3	4	5	6	7	8



Kruskal Algoritması

- **ÖRNEK;**

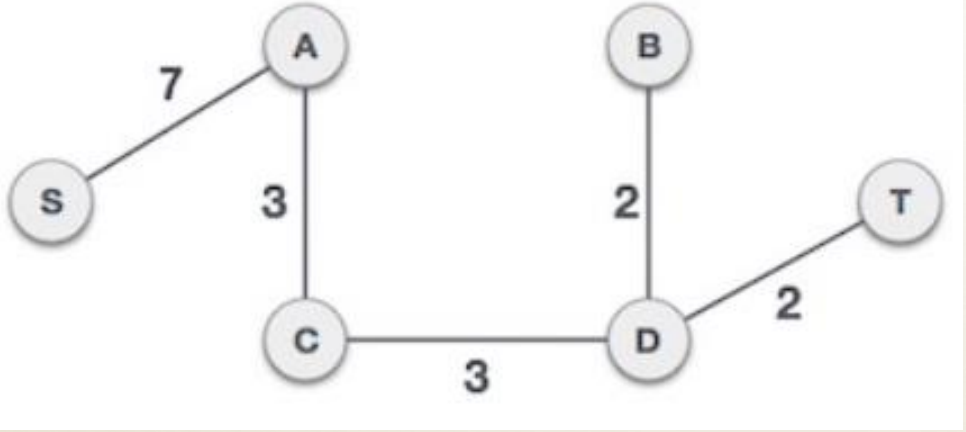
B, D	D, T	A, C	C, D	C, B	B, T	A, B	S, A	S, C
2	2	3	3	4	5	6	7	8



Kruskal Algoritması

- **ÖRNEK;**

B, D	D, T	A, C	C, D	C, B	B, T	A, B	S, A	S, C
2	2	3	3	4	5	6	7	8



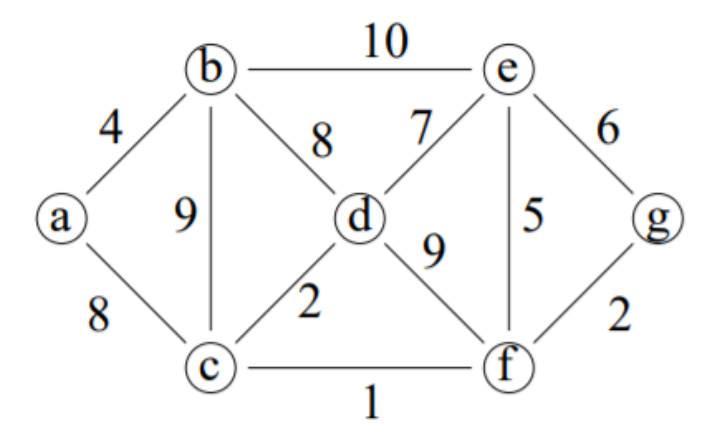
- Elde edilen Min spanning tree maliyet: 2+2+3+3+7=17

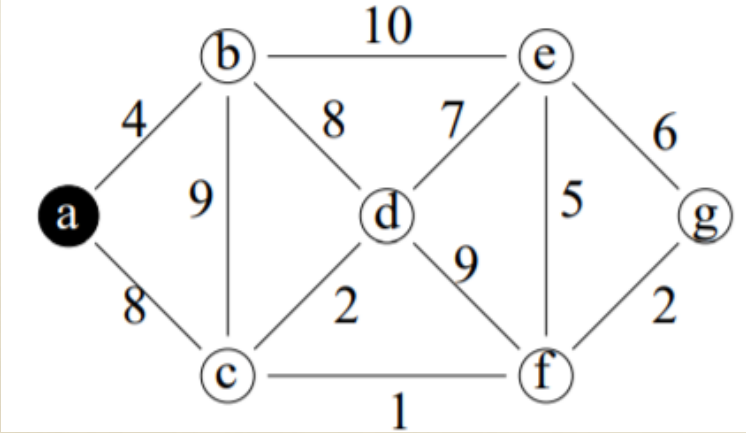
Prim Algoritması

- Kruskal'dan farklı olarak her adımda kenar değil düğüm ekler.
- Adımları şöyledir;
  - Rastgele bir başlangıç noktası seçilir.
  - Maliyeti daha az olan düğüme gidilir.
  - Gidilebilecek düğümler listesine artık 2. düğümden gidilebilecek olanlar da eklenmiştir.
  - Tüm düğümler dolaşıldığında algoritma sonlanır.

Prim Algoritması

- **ÖRNEK;**

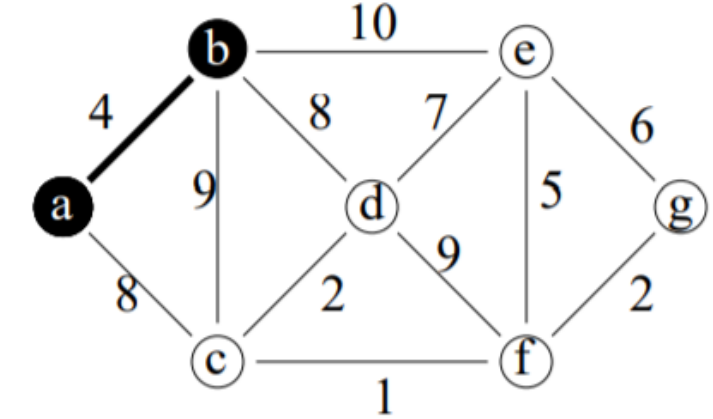




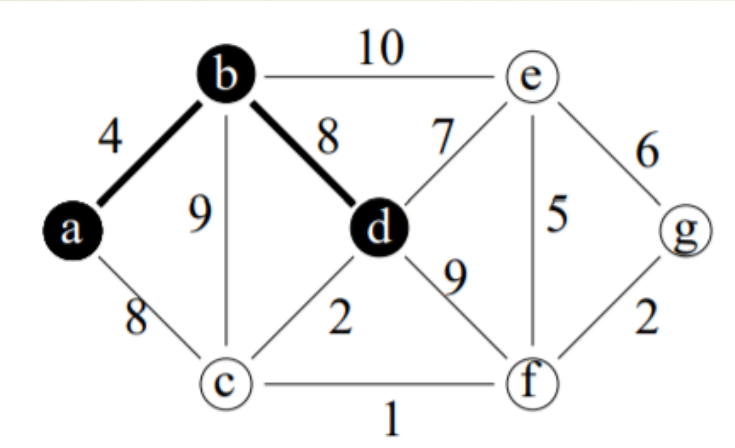
Step 0;
S={a}
V={b,c,d,e,f,g}
Seçilebilecek düğümler: {(a,b),(a,c)}

Prim Algoritması

- **ÖRNEK;**



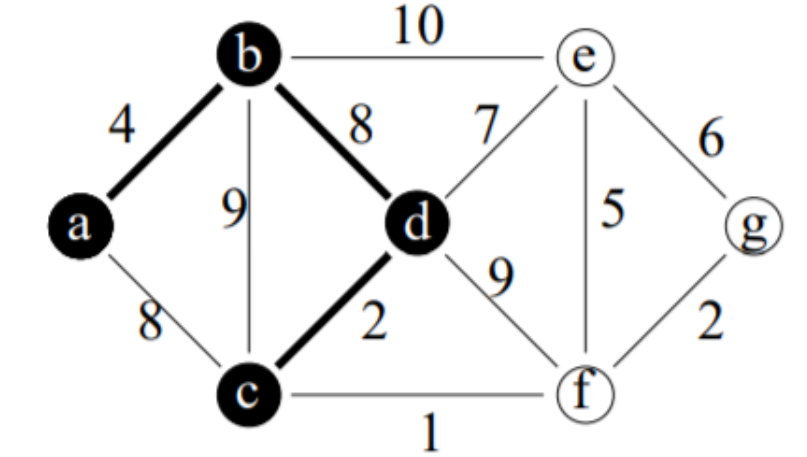
Step 1;
S={a,b}
V={c,d,e,f,g}
A={(a,b)}
Seçilebilecek düğümler : {(a,c),(b,e),(b,d)}



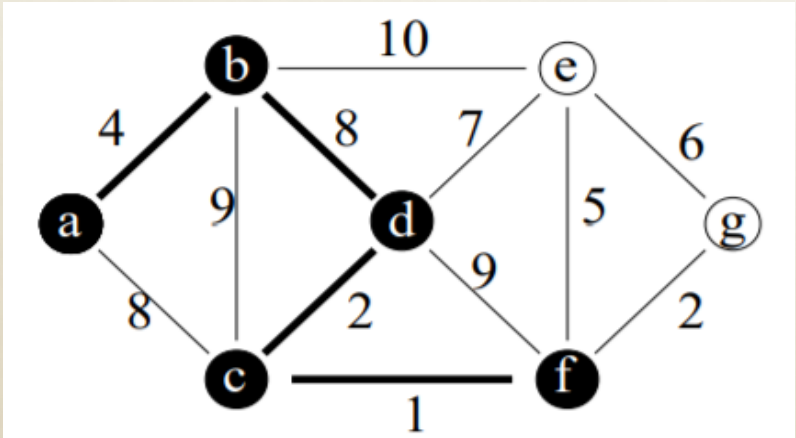
Step 2;
S={a,b,d}
V={c,e,f,g}
A={(a,b),(b,d)}
Seçilebilecek düğümler :
{(a,c),(b,e),(d,c),(d,e),(d,f)}

Prim Algoritması

- **ÖRNEK;**



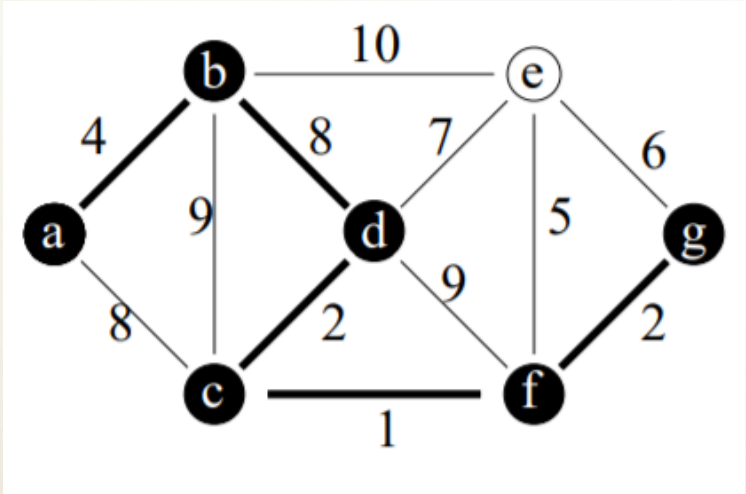
Step 3;
S={a,b,d,c}
V={e,f,g}
A={(a,b),(b,d),(d,c)}
Seçilebilecek düğümler :
{(a,c),(b,e),(d,e),(d,f),(c,f)}



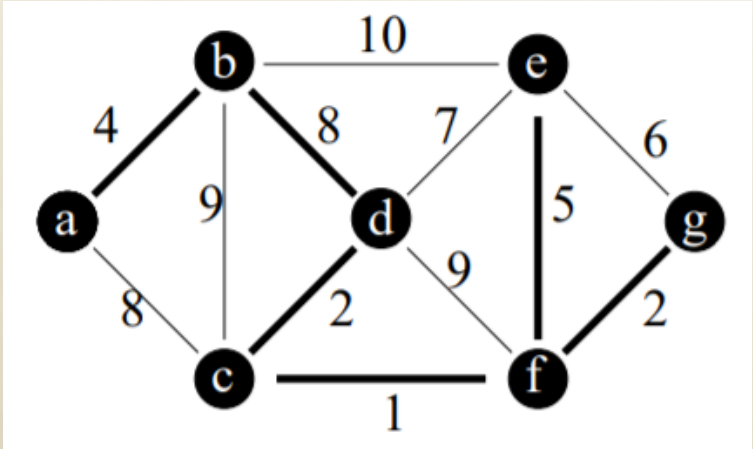
Step 4;
S={a,b,d,c,f}
V={e,g}
A={(a,b),(b,d),(d,c),(c,f)}
Seçilebilecek düğümler :
{(a,c),(b,e),(d,e),(d,f),(f,e),(f,g)}

Prim Algoritması

- **ÖRNEK;**



Step 5;
S={a,b,d,c,f,g}
V={e}
A={(a,b),(b,d),(d,c),(c,f),(f,g)}
Seçilebilecek düğümler :
{(a,c),(b,e),(d,e),(d,f),(f,e),(g,e)}



Step 6;
S={a,b,d,c,f,g,e}
V={}
A={(a,b),(b,d),(d,c),(c,f),(f,g),(g,e)}

Kruskal vs Prim

Prim

- Bir düğüm ile başlar
- Komşuluk matrisi, binary heap veya fibonacci heap ile gerçeklenebilir
- Yoğun (dense) graflarda hızlıdır
- Karmaşıklık: O(E + V log V)
- Min mesafeli düğümü ekleyebilmek için bir düğüme birden fazla defa uğrayabilir.
- Eklenecek bir sonraki düğüm eklenmiş olanlardan biriyle bağlantılıdır.

Kruskal

- En az ağırlıklı kenar ile başlar
- Ayrık küme kullanır
- Sparse graflarda hızlıdır.
- Karmaşıklık: O(E log V)
- Kenarları sadece bir kere gezer, cycle oluşup oluşmamasına göre ekler veya eklemez
- Eklenecek bir sonraki düğüm eklenmiş olanlardan biriyle bağlantılı olabilir de olmayabilir de