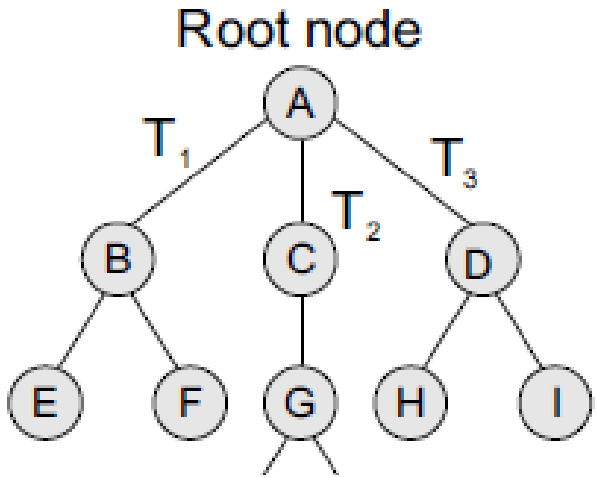
BLM267

Bölüm 9: Ağaçlar
C Kullanarak Veri Yapıları, İkinci Baskı

- giriiş
- Ağaç Türleri
- Genel Bir Ağaçtan İkili Bir Ağaç Oluşturma
- İkili Bir Ağacı Geçmek
- Huffman'ın Ağacı

giriiş

- Bir ağaç, bir düğümün ağacın kökü olarak belirlendiği ve kalan tüm düğümlerin her biri kökün bir alt ağacı olan boş olmayan kümelere bölünebildiği bir veya daha fazla düğüm kümesi olarak yinelemeli olarak tanımlanır.



- A düğümü kök düğümdür, B, C ve D düğümleri kök düğümün çocuklarıdır ve A düğümünde köklenen ağacın alt ağaçlarını oluştururlar.

giriiş

- **Temel Terminoloji**
  - ***Kök düğümü Kök düğümü R, ağaçtaki en üst düğümdür. Eğer R = NULL ise, bu ağacın boş olduğu anlamına gelir.***
  - ***Alt ağaçlar Eğer kök düğüm R NULL değilse, T1, T2 ve T3 ağaçları R'nin alt ağaçları olarak adlandırılır.***
  - ***Yaprak Düğüm Çocuğu olmayan düğüme yaprak düğüm veya terminal düğüm denir.***
  - ***Yol Ardışık kenarlardan oluşan bir diziye yol denir. Örneğin, Şekil'de, kök düğüm A'dan düğüm I'e giden yol şu şekilde verilmiştir: A, D ve I.***
  - ***Ata Düğümü Bir düğümün atası, kök düğümden o düğüme giden yoldaki herhangi bir öncül düğümdür. Kök düğümün herhangi bir atası yoktur. Şekilde verilen ağaçta, A, C ve G düğümleri K düğümünün atalarıdır.***
  - ***Alt düğüm Alt düğüm, düğümden yaprak düğüme giden herhangi bir yoldaki herhangi bir ardıl düğümdür. Yaprak düğümlerin hiçbir alt düğümü yoktur. Şekilde verilen ağaçta, C, G, J ve K düğümleri A düğümünün alt düğümleridir.***

giriiş

- **Temel Terminoloji**
  - ***Seviye numarası Ağaçtaki her düğüme, kök düğümün seviye 0, kök düğümün çocuklarının seviye numarası 1 olacak şekilde bir seviye numarası atanır. Böylece, her düğüm ebeveyninden bir seviye daha yüksektir. Bu nedenle, tüm çocuk düğümlerin ebeveynin seviye numarası + 1 ile verilen bir seviye numarası vardır.***
  - ***Derece Bir düğümün derecesi, bir düğümün sahip olduğu çocuk sayısına eşittir. Bir yaprak düğümün derecesi sıfırdır.***
  - ***Giriş derecesi Bir düğümün giriş derecesi, o düğüme gelen kenar sayısıdır.***
  - ***Dış derece Bir düğümün dış derecesi, o düğümden ayrılan kenar sayısıdır.***

Ağaç Türleri

- Ağaçlar aşağıdaki 6 tiptedir:
  - 1. Genel ağaçlar
  - 2. Ormanlar
  - 3. İkili ağaçlar
  - 4. İkili arama ağaçları
  - 5. İfade ağaçları
  - 6. Turnuva ağaçları

Ağaç Türleri

- **Genel ağaçlar**
  - Genel ağaçlar, öğeleri hiyerarşik olarak depolayan veri yapılarıdır.
  - Bir ağacın en üst düğümü kök düğümdür ve kök hariç her düğümün bir ebeveyni vardır.
  - Genel bir ağaçtaki bir düğümün (yaprak düğümler hariç) sıfır veya daha fazla alt ağacı olabilir.
  - Her düğümde 3 alt ağaç bulunan genel ağaçlara üçlü ağaçlar denir.
  - Ancak, herhangi bir düğüm için alt ağaç sayısı değişken olabilir. Örneğin, bir düğümün 1 alt ağacı olabilirken, başka bir düğümün 3 alt ağacı olabilir.

Ağaç Türleri

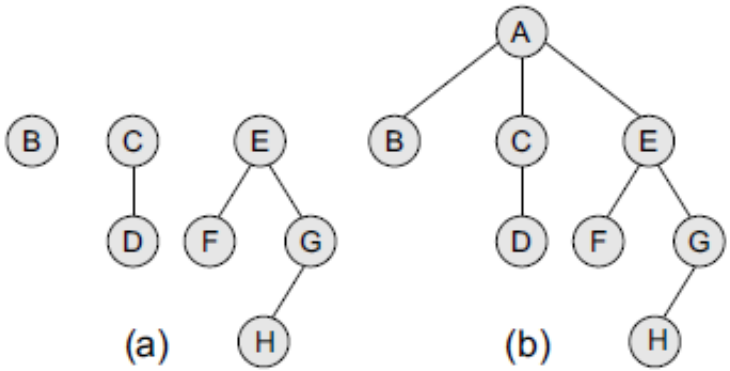
- **Genel ağaçlar**
  - Genel ağaçlar ADT'ler olarak temsil edilebilmesine rağmen, kendisine bağlı maksimum sayıda alt ağaç bulunan bir düğüme başka bir alt ağaç eklendiğinde her zaman bir sorun ortaya çıkar.
  - Hatta düğümleri arama, gezinme, ekleme ve silme algoritmaları bile çok daha karmaşık hale geliyor çünkü herhangi bir düğüm için sadece iki olasılık değil, birden fazla olasılık bulunuyor.
  - Genel bir ağaç ikili ağaca dönüştürüldüğünde iyi biçimlendirilmiş veya tam olmayabilir, ancak böyle bir dönüşümün avantajları programcıya ikili ağaçlar için kullanılan işlemler için algoritmaları küçük değişikliklerle kullanma olanağı sağlar.

Ağaç Türleri

- **Ormanlar**
  - Orman, ağaçların bir araya gelerek oluşturdukları ayrı ayrı topluluktur.
  - Kök ve kök düğümünü 1. seviyedeki düğümlere bağlayan kenarların silinmesiyle ayrık ağaçlar (veya ormanlar) kümesi elde edilir.
  - Bir ağacın her düğümünün bir alt ağacın kökü olduğunu daha önce görmüştük.
  - Dolayısıyla bir düğümün hemen altında bulunan tüm alt ağaçlar bir ormanı oluşturur.
  - Orman, sıfır veya daha genel ağaçlardan oluşan sıralı bir küme olarak da tanımlanabilir.
  - Genel bir ağacın mutlaka bir kökü olması gerekirken, bir orman boş olabilir çünkü tanımı gereği bir kümedir ve kümeler boş olabilir.

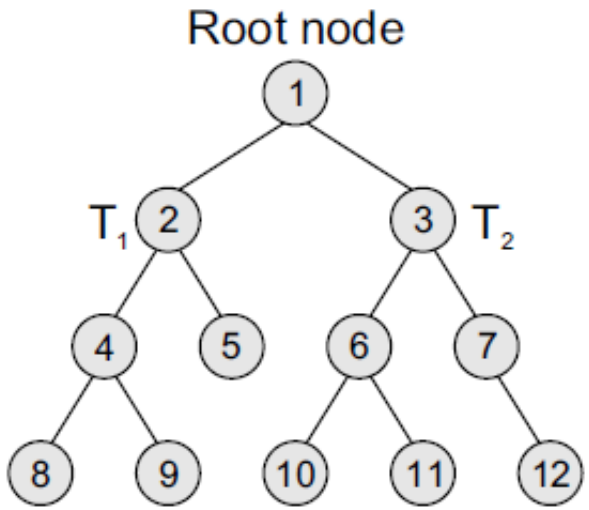
Ağaç Türleri

- **Ormanlar**
  - Ağacın kök düğümü olarak tek bir düğüm ekleyerek bir ormanı bir ağaca dönüştürebiliriz. Örneğin, Şekil a bir ormanı gösterir ve Şekil b karşılık gelen ağacı gösterir.
  - Benzer şekilde, genel bir ağacı, ağacın kök düğümünü silerek bir ormana dönüştürebiliriz.



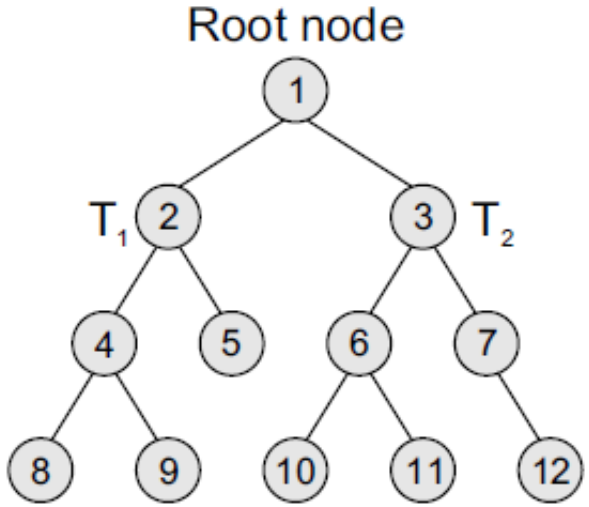
Ağaç Türleri

- **İkili Ağaçlar**
  - İkili ağaç, düğüm adı verilen öğelerin bir koleksiyonu olarak tanımlanan bir veri yapısıdır. İkili ağaçta, en üstteki öğeye kök düğüm denir ve her düğümün 0, 1 veya en fazla 2 çocuğu vardır.
  - Sıfır çocuğu olan bir düğüme yaprak düğüm veya terminal düğüm denir. Her düğüm bir veri öğesi, sol çocuğu işaret eden bir sol işaretçi ve sağ çocuğu işaret eden bir sağ işaretçi içerir.
  - Kök eleman bir 'kök' işaretçisi tarafından işaret edilir. Eğer kök = NULL ise, bu ağacın boş olduğu anlamına gelir.
  - Şekil ikili bir ağacı göstermektedir. Şekilde, R kök düğümdür ve iki ağaç T1 ve T2, R'nin sol ve sağ alt ağaçları olarak adlandırılır. T1, R'nin sol halefi olarak adlandırılır. Aynı şekilde, T2, R'nin sağ halefi olarak adlandırılır

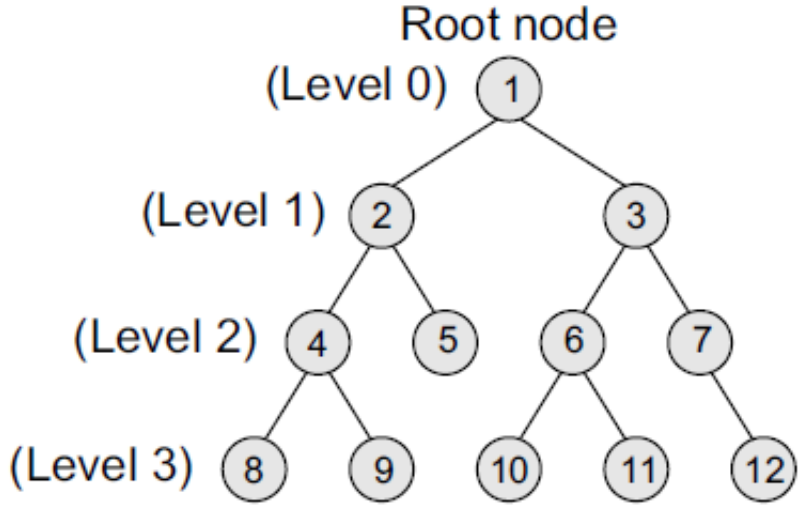


Ağaç Türleri

- **İkili Ağaçlar**
  - Kök düğümün sol alt ağacının 2, 4, 5, 8 ve 9 düğümlerinden oluştuğunu unutmayın. Benzer şekilde, kök düğümün sağ alt ağacı 3, 6, 7, 10, 11 ve 12 düğümlerinden oluşur.
  - Ağaçta kök düğüm 1'in iki ardılı vardır: 2 ve 3. Düğüm 2'nin iki ardılı düğümü vardır: 4 ve 5.
  - Düğüm 4'ün iki ardılı vardır: 8 ve 9. Düğüm 5'in ardılı yoktur. Düğüm 3'ün iki ardıl düğümü vardır: 6 ve 7. Düğüm 6'nın iki ardılı vardır: 10 ve 11. Son olarak, düğüm 7'nin yalnızca bir ardılı vardır: 12.
  - İkili bir ağaç, ağaçtaki her düğümün bir sol alt ağaç ve bir sağ alt ağaç içermesi nedeniyle tanımı gereği yinelemelidir. Terminal düğümler bile boş bir sol alt ağaç ve boş bir sağ alt ağaç içerir.
  - Şekle bakın. 5, 8, 9, 10, 11 ve 12 numaralı düğümlerin ardılı yoktur ve bu neder kları söylenir.

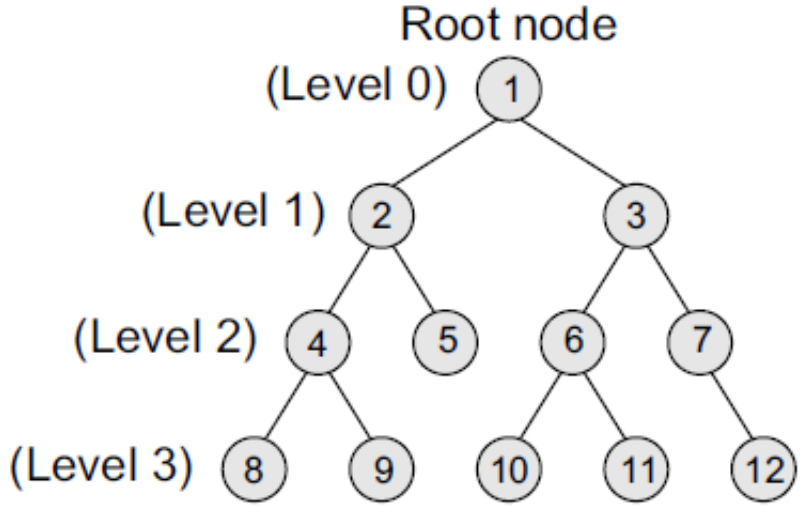


Ağaç Türleri



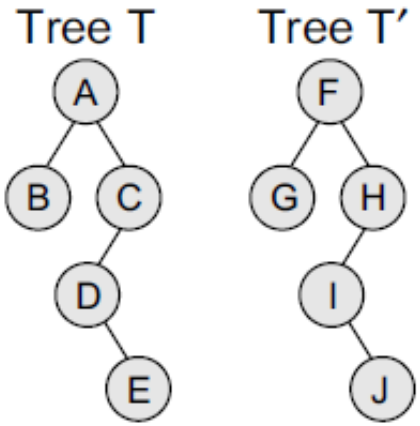
- **İkili Ağaçlar (Terminoloji)**
  - ***Ebeveyn Eğer N, T'de sol ardılı S1 ve sağ ardılı S2 olan herhangi bir düğümse, o zaman N, S1 ve S2'nin ebeveyni olarak adlandırılır. Buna uygun olarak, S1 ve S2, N'nin sol çocuğu ve sağ çocuğu olarak adlandırılır. Kök düğüm dışındaki her düğümün bir ebeveyni vardır.***
  - ***Seviye numarası İkili ağaçtaki her düğüme bir seviye numarası atanır (Şekil'e bakın). Kök düğüm 0 seviyesinde olacak şekilde tanımlanır. Kök düğümün sol ve sağ çocuğu 1 seviye numarasına sahiptir. Benzer şekilde, her düğüm ebeveynlerinden bir seviye daha yüksektir. Bu nedenle tüm çocuk düğümler ebeveynin seviye numarası + 1 olarak seviye numarasına sahip olacak şekilde***

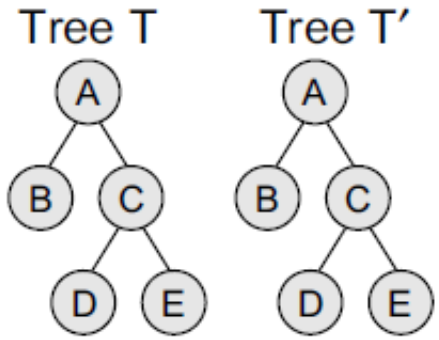
Ağaç Türleri



- **İkili Ağaçlar (Terminoloji)**
  - ***Bir düğümün derecesi Bir düğümün sahip olduğu çocuk sayısına eşittir. Bir yaprak düğümünün derecesi sıfırdır. Örneğin, ağaçta, 4. düğümün derecesi 2, 5. düğümün derecesi sıfır ve 7. düğümün derecesi 1'dir.***
  - ***Kardeş Aynı düzeyde olan ve aynı ebeveyni paylaşan tüm düğümle kardeş (brothers) denir. Örneğin, düğümler 2 ve 3; düğümler 4 ve 5 düğümler 6 ve 7; düğümler 8 ve 9; ve düğümler 10 ve 11 kardeşlerdir.***
  - ***Yaprak düğümü Çocuğu olmayan bir düğüme yaprak düğüm veya terminal düğüm denir. Ağaçtaki yaprak düğümleri şunlardır: 8, 9, 5, 10, 11 ve 12.***

Ağaç Türleri



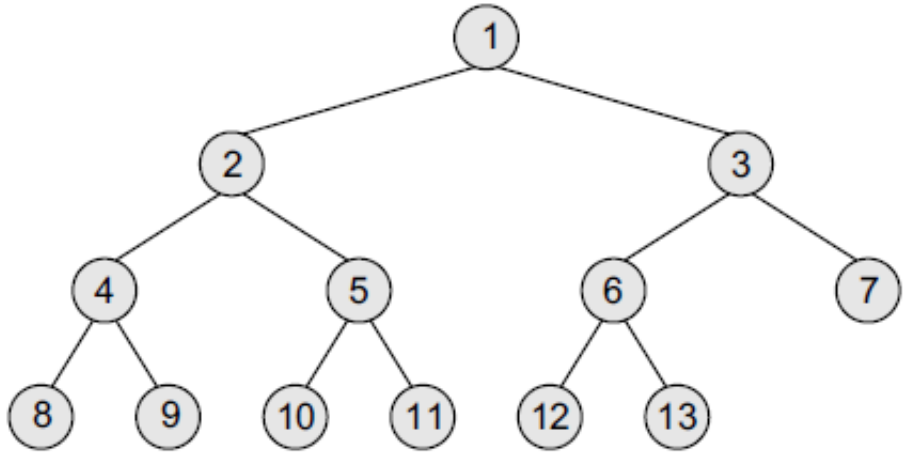


- **İkili Ağaçlar (Terminoloji)**
  - ***Benzer ikili ağaçlar İki ikili ağaç T ve T', her iki ağacın da aynı yapıya sahip olması durumunda benzer olarak adlandırılır. Şekilde iki benzer ikili ağaç gösterilmektedir.***
  - ***Kopyalar İki ikili ağaç T ve T'nin benzer yapıya sahip olmaları ve karşılık gelen düğümlerde aynı içeriğe sahip olmaları durumunda kopya oldukları söylenir. Şekil, T'nin T'nin bir kopyası olduğunu gösterir.***
  - ***Kenar Bir düğüm N'yi haleflerinden herhangi birine bağlayan çizgidir. n düğümlü bir ikili ağacın tam olarak n – 1 kenarı vardır çünkü kök düğüm hariç her düğüm bir kenar aracılığıyla ebeveynine bağlıdır.***
  - ***Yol Ardışık kenarların bir dizisi. Örneğin, Şekil'de, kök düğümden düğüm 8'e giden yol şu şekilde verilmiştir: 1, 2, 4 ve 8.***

Ağaç Türleri

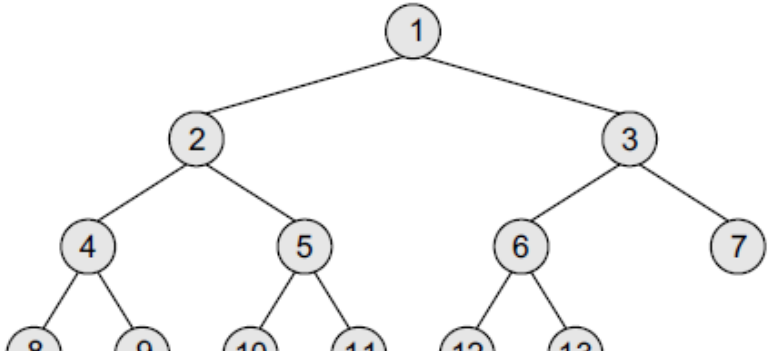
- **İkili Ağaçlar (Terminoloji)**
  - ***Derinlik Bir düğümün (N) derinliği, kök R'den düğüm N'ye giden yolun uzunluğu olarak verilir. Kök düğümün derinliği sıfırdır.***
  - ***Bir ağacın yüksekliği Kök düğümden ağaçtaki en derin düğüme kadar olan yoldaki düğümlerin toplam sayısıdır. Sadece kök düğümü olan bir ağacın yüksekliği 1'dir.***
  - Yüksekliği h olan bir ikili ağaç en az h düğüme ve en fazla 2^h – 1 düğüme sahiptir. Bunun nedeni her seviyenin en az bir düğüme sahip olması ve en fazla 2 düğüme sahip olabilmesidir. Yani, her seviyenin iki düğümü varsa, yüksekliği h olan bir ağaç en fazla 2^h – 1 düğüme sahip olacaktır çünkü seviye 0'da kök adı verilen tek bir eleman vardır. n düğümlü bir ikili ağacın yüksekliği en az log2(n+1) ve en fazla n'dir.
  - ***Bir düğümün giriş derecesi/dış derecesi Bir düğüme ulaşan kenar sayısıdır. Kök düğüm, giriş derecesi sıfıra eşit olan tek düğümdür.***
  - Benzer şekilde, bir düğümün dış derecesi, o düğümden ayrılan kenar sayısıdır.
  - İkili ağaçlar genellikle ikili arama ağaçları, ifade ağaçları, turnuva ağaçları ve ikili yığınları uygulamak için kullanılır.

Ağaç Türleri



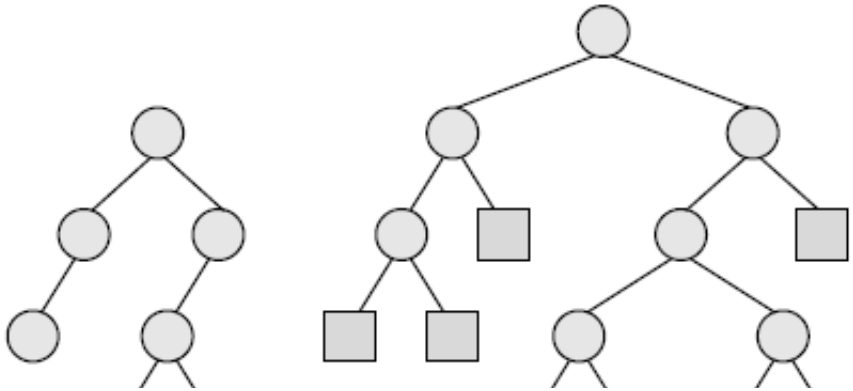
- **İkili Ağaçların Tamamı**
  - Tam ikili ağaç, iki özelliği karşılayan bir ikili ağaçtır. Birincisi, tam ikili ağaçta, muhtemelen sonuncusu hariç her seviye tamamen doludur. İkincisi, tüm düğümler mümkün olduğunca solda görünür.
  - Tam bir ikili ağaç T^n'de tam olarak n düğüm vardır ve T'nin r düzeyinde en fazla 2r düğüm bulunabilir.
  - Şekilde tam bir ikili ağaç gösterilmektedir.
  - Şekilde, seviye 0'ın $2^0 = 1$ düğüme, seviye 1'in $2^1 = 2$ düğüme, seviye 2'nin $2^2 = 4$ düğüme, seviye 3'ün 6 düğüme sahip olduğunu ve bunun maksimum olan $2^3 = 8$ düğümden küçük olduğunu unutmayın.

Ağaç Türleri



- **İkili Ağaçların Tamamı**
  - Şekilde, T^{13} ağacının tam olarak 13 düğümü vardır. Bunlar, okuyucunun verilen düğümün ana düğümünü, sağ alt düğümünü ve sol alt düğümünü bulmasını kolaylaştırmak için özellikle 1'den 13'e kadar etiketlenmiştir.
  - Formül şu şekilde verilebilir: K bir ebeveyn düğüm ise, sol çocuğu 2 × K olarak hesaplanabilir ve sağ çocuğu 2 × K + 1 olarak hesaplanabilir. Örneğin, 4 numaralı düğümün çocukları 8 (2 × 4) ve 9'dur (2 × 4 + 1). Benzer şekilde, K düğümünün ebeveyni | K/2 | olarak hesaplanabilir.
  - 4 numaralı düğüm verildiğinde, ebeveyni | 4/2 | = 2 olarak hesaplanabilir. Tam olarak n düğümü olan bir ağacın yüksekliği T^n şu şekilde verilir: Hn = | log2 (n + 1) |
  - Bu, eğer T ağacında 10.00.000 düğüm varsa, yüksekliğinin

Ağaç Türleri



- **Genişletilmiş İkili Ağaçlar**
  - İkili ağaç T'nin, ağaçtaki her düğümün ya hiç çocuğu yoksa ya da tam olarak iki çocuğu varsa, genişletilmiş ikili ağaç (veya 2-ağaç) olduğu söylenir. Şekil, sıradan bir ikili ağacın genişletilmiş ikili ağaca nasıl dönüştürüldüğünü gösterir.
  - Genişletilmiş bir ikili ağaçta, iki çocuğu olan düğümlere iç düğümler, çocuğu olmayan düğümlere ise dış düğümler denir. Şekilde, iç düğümler dairelerle, dış düğümler ise karelerle gösterilir.
  - İkili bir ağacı genişletilmiş bir ağaca dönüştürmek için, her boş alt ağaç yeni bir düğümle değiştirilir.
  - Ağaçtaki orijinal düğümler iç düğümlerdir ve eklenen yeni düğümlere dış düğümler denir.

Ağaç Türleri

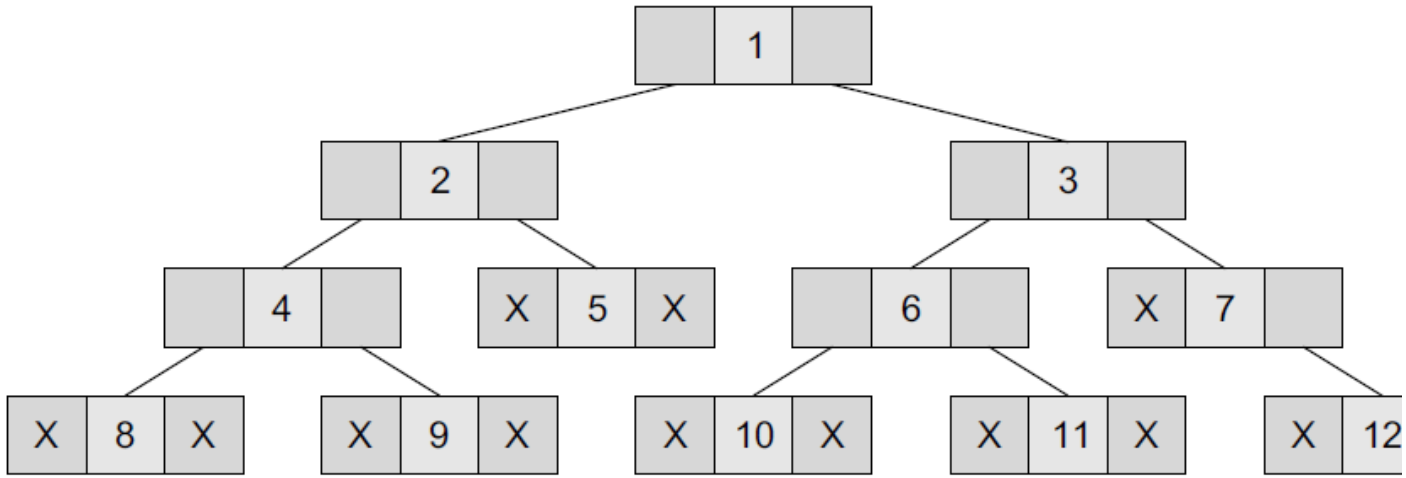
- ***Bellekte İkili Ağaçların Temsili***
- Bilgisayarın belleğinde ikili ağaç, ya bağlantılı gösterim kullanılarak ya da sıralı gösterim kullanılarak tutulabilir.
  - ***İkili ağaçların bağlantılı gösterimi İkili ağacın bağlantılı gösteriminde, her düğümün üç bölümü olacaktır: veri öğesi, sol düğüme bir işaretçi ve sağ düğüme bir işaretçi.***
  - Yani C'de ikili ağaç aşağıda verilen düğüm tipiyle oluşturulur.

```
 yapı düğümü {
yapı düğümü *sol;
int veri;
yapı düğümü *sağ;
 };
```

  - Her ikili ağacın, ağacın kök elemanına (en üst eleman) işaret eden bir ROOT işaretçisi vardır. ROOT = NULL ise, ağaç boştur. İkili ağacın bağlantılı gösteriminin şematik diyagramı, bir sonraki slaytta verilen Şekil'de gösterilmiştir.

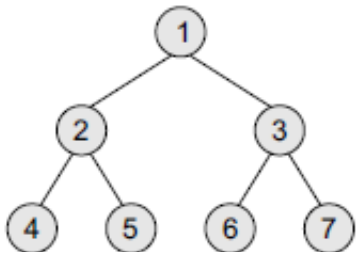
Ağaç Türleri

- ***Bellekte İkili Ağaçların Temsili***
  - Bu Şekilde, sol konum düğümün sol çocuğunu işaret etmek veya düğümün sol çocuğunun adresini depolamak için kullanılır. Orta konum verileri depolamak için kullanılır. Son olarak, sağ konum düğümün sağ çocuğunu işaret etmek veya düğümün sağ çocuğunun adresini depolamak için kullanılır.
  - Boş alt ağaçlar X (NULL anlamına gelir) kullanılarak gösterilir.



Ağaç Türleri

- ***Bellekte İkili Ağaçların Temsili***



ROOT
3

15
AVAIL

	LEFT	DATA	RIGHT
1	–1	8	–1
2	–1	10	–1
3	5	1	8
4			
5	9	2	14
6			
7			
8	20	3	11
9	1	4	12
10			
11	–1	7	18
12	–1	9	–1
13			
14	–1	5	–1
15			
16	–1	11	–1
17			
18	–1	12	–1

Ağaç Türleri

- ***Bellekte İkili Ağaçların Temsili***

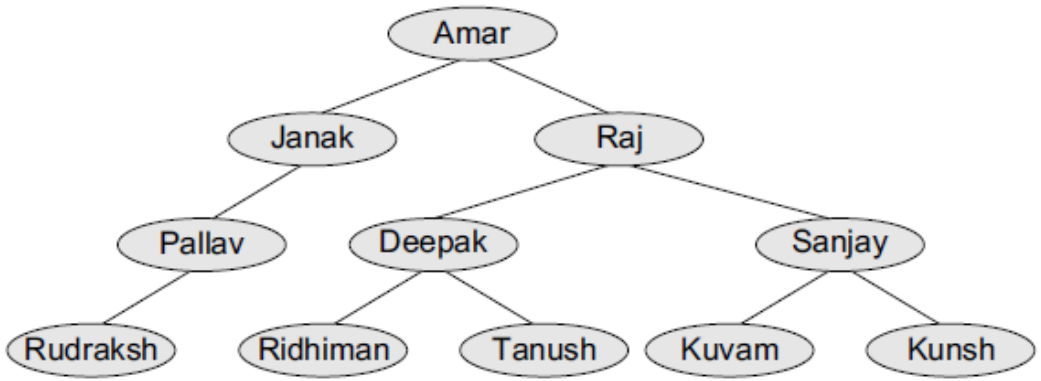
Example 9.1 Given the memory representation of a tree that stores the names of family members, construct the corresponding tree from the given data.

Solution

ROOT 3

7 AVAIL

	LEFT	NAMES	RIGHT
1	12	Pallav	–1
2			
3	9	Amar	13
4			
5			
6	19	Deepak	17
7			
8			
9	1	Janak	–1
10			
11	–1	Kuvam	–1
12	–1	Rudraksh	–1
13	6	Raj	20
14			
15	–1	Kunsh	–1
16			
17	–1	Tanush	–1
18			
19	–1	Ridhiman	–1
20	11	Sanjay	15

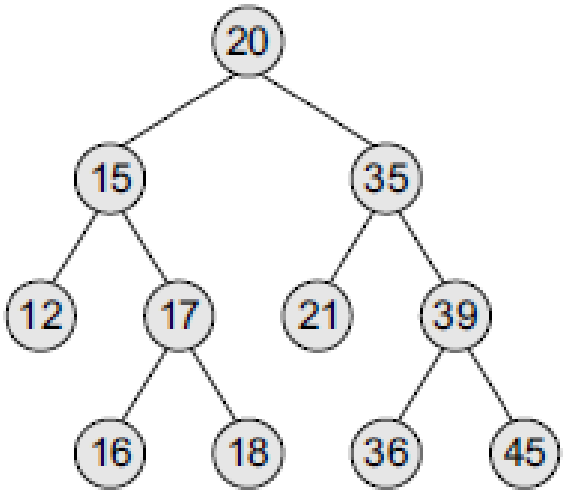


Ağaç Türleri

- ***Bellekte İkili Ağaçların Temsili***
  - ***İkili ağaçların sıralı gösterimi Ağaçların sıralı gösterimi tek veya bir boyutlu diziler kullanılarak yapılır.***
  - Bellek gösterimi için en basit teknik olmasına rağmen, çok fazla bellek alanı gerektirdiği için verimsizdir.
  - Sıralı ikili ağaç aşağıdaki kuralları takip eder:
    - Tree elemanlarını depolamak için TREE adı verilen tek boyutlu bir dizi kullanılır.
    - Ağacın kökü ilk konumda depolanacaktır. Yani, TREE[1] kök öğesinin verilerini depolayacaktır.
    - K konumunda depolanan bir düğümün çocukları (2 × K) ve (2 × K+1) konumlarında depolanacaktır.
    - TREE dizisinin maksimum boyutu (2h–1) olarak verilir; burada h ağacın yüksekliğidir.
    - Boş bir ağaç veya alt ağaç NULL kullanılarak belirtilir. TREE[1]= NULL ise ağaç boştur.
  - Sonraki slaytta verilen şekil ikili bir ağacı ve buna karşılık gelen sıralı gösterimi göstermektedir. Ağacın 11 düğümü vardır ve yüksekliği 4'tür.

Ağaç Türleri

- ***Bellekte İkili Ağaçların Temsili***
  - ***İkili ağaçların sıralı gösterimi***



1	20
2	15
3	35
4	12
5	17
6	21
7	39
8	
9	
10	16
11	18
12	
13	
14	36
15	45

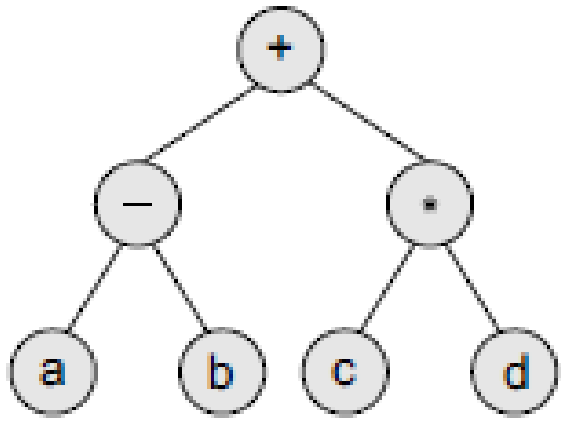
Ağaç Türleri

- **İkili Arama Ağaçları**
  - İkili arama ağacı, sıralı ikili ağaç olarak da bilinir, düğümlerin bir sıraya göre düzenlendiği ikili ağacın bir çeşididir.
  - Bir sonraki bölümde ikili arama ağaçları kavramını ve bunlar üzerinde gerçekleştirilen farklı işlemleri ele alacağız.

Ağaç Türleri

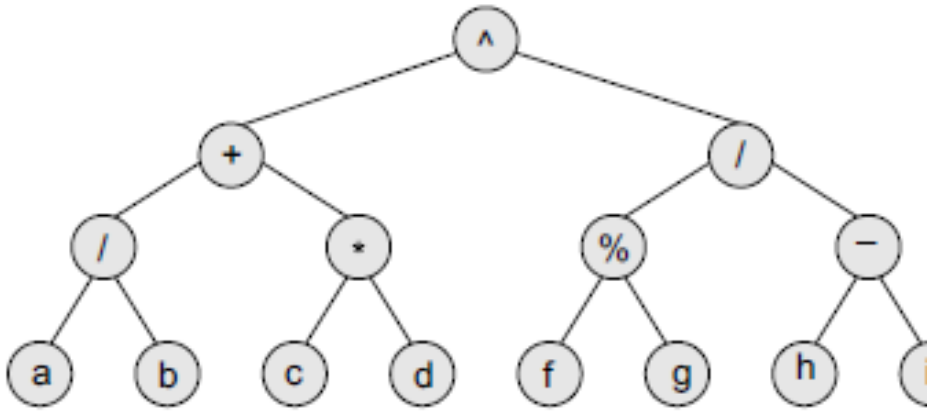
- **İfade Ağaçları**
  - İkili ağaçlar cebirsel ifadeleri depolamak için yaygın olarak kullanılır. Örneğin, verilen cebirsel ifadeyi ele alalım:

    Deney = (a – b) + (c * d)
  - Bu ifade, Şekil'de gösterildiği gibi ikili bir ağaç kullanılarak gösterilebilir.



Ağaç Türleri

- **İfade Ağaçları**
  - İkili ağaç verildiğinde, bunun temsil ettiği ifadeyi yazın.



Yukarıdaki ikili ağacın ifadesi şu şekildedir:

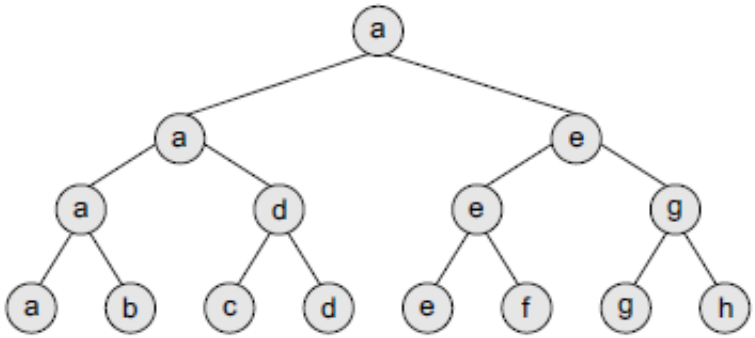
[{(a/b) + (c*d)} ^ {(f % g)/(h – i)}]

Ağaç Türleri

- **Turnuva Ağaçları**
  - Hepimiz biliyoruz ki bir turnuvada, diyelim ki satrançta, n sayıda oyuncu katılır. Tüm bu oyuncular arasından kazananı belirlemek için birkaç maç oynanır ve genellikle oyunda üç tur oynanır.
  - 1. turdaki her maçta, iki oyuncunun birbirine karşı oynadığı bir dizi maç oynanır.
  - 1. turda oynanacak maç sayısı oyuncu sayısına bağlı olacaktır. Örneğin, bir satranç turnuvasına 8 oyuncu katılıyorsa, 1. turda 4 maç oynanacaktır. 1. turdaki her maç iki oyuncu arasında oynanacaktır.
  - Daha sonra 2. turda, 1. tur kazananları birbirlerine karşı oynayacaklar. Benzer şekilde, 3. turda, 2. tur kazananları birbirlerine karşı oynayacaklar ve 3. turu kazanan kişi kazanan ilan edilecek.
  - Bu kavramı temsil etmek için turnuva ağaçları kullanılmaktadır.

Ağaç Türleri

- **Turnuva Ağaçları**
  - Bir turnuva ağacında (seçim ağacı olarak da adlandırılır), her dış düğüm bir oyuncuyu, her iç düğüm ise alt düğümleri tarafından temsil edilen oyuncular arasında oynanan maçın kazananını temsil eder.
  - Bu turnuva ağaçlarına aynı zamanda kazanan ağaçları da denir çünkü her seviyedeki kazananı kaydetmek için kullanılır.
  - Toplamda isimleri a, b, c, d, e, f, g ve h ile gösterilen 8 oyuncu var. 1. turda a ve b; c ve d; e ve f; ve son olarak g ve h birbirlerine karşı oynuyorlar.
  - 2.turda 1.turun galipleri, yani a, d, e, g takımları birbirleriyle karşılaşır.
  - 3.turda 2.turun galipleri olan a ve e takımları birbirleriyle karşılaşıyor.
  - Ağaçta kök düğüm a kazananı belirtir.

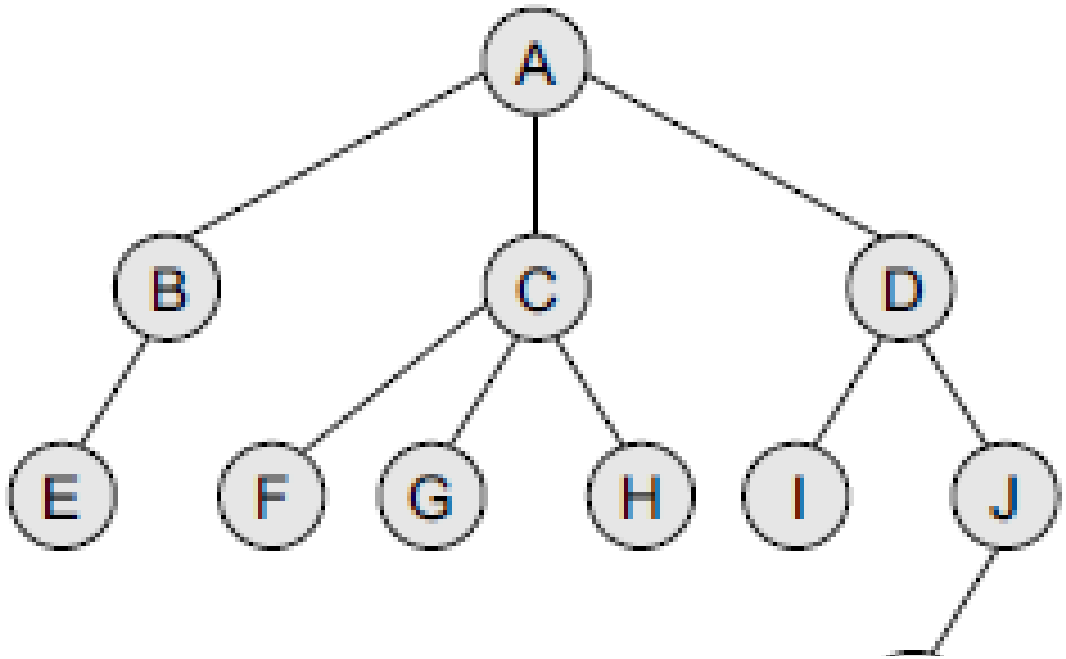


Genel Bir Ağaçtan İkili Bir Ağaç Oluşturma

- Genel bir ağacı ikili bir ağaca dönüştürme kuralları aşağıda verilmiştir. Genel bir ağacın ikili bir ağaca dönüştürüldüğünü ve ikili bir arama ağacına dönüştürülmediğini unutmayın.
  - *Kural 1: İkili ağacın kökü = Genel ağacın kökü*
  - *Kural 2: Bir düğümün sol çocuğu = Genel ağaçtaki ikili ağaçtaki düğümün en soldaki çocuğu*
  - *Kural 3: İkili ağaçtaki bir düğümün sağ çocuğu = Genel ağaçtaki düğümün sağ kardeşi*

Genel Bir Ağaçtan İkili Bir Ağaç Oluşturma

- Verilen genel ağacı ikili ağaca dönüştürün.

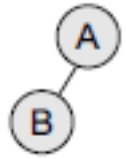


Genel Bir Ağaçtan İkili Bir Ağaç Oluşturma

Now let us build the binary tree.

Step 1: Node A is the root of the general tree, so it will also be the roo

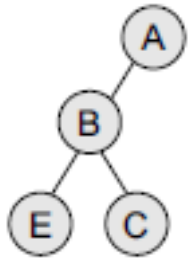
Step 1



Step 2: Left child of node A is the leftmost child of node A in the ge child of node A is the right sibling of the node A in the general tree. S right sibling in the general tree, it has no right child in the binary tree

Step 3: Now process node B. Left child of B is E and its right child i general tree).

Step 2



Step 4: Now process node C. Left child of C is F (leftmost child) an (right sibling in general tree).

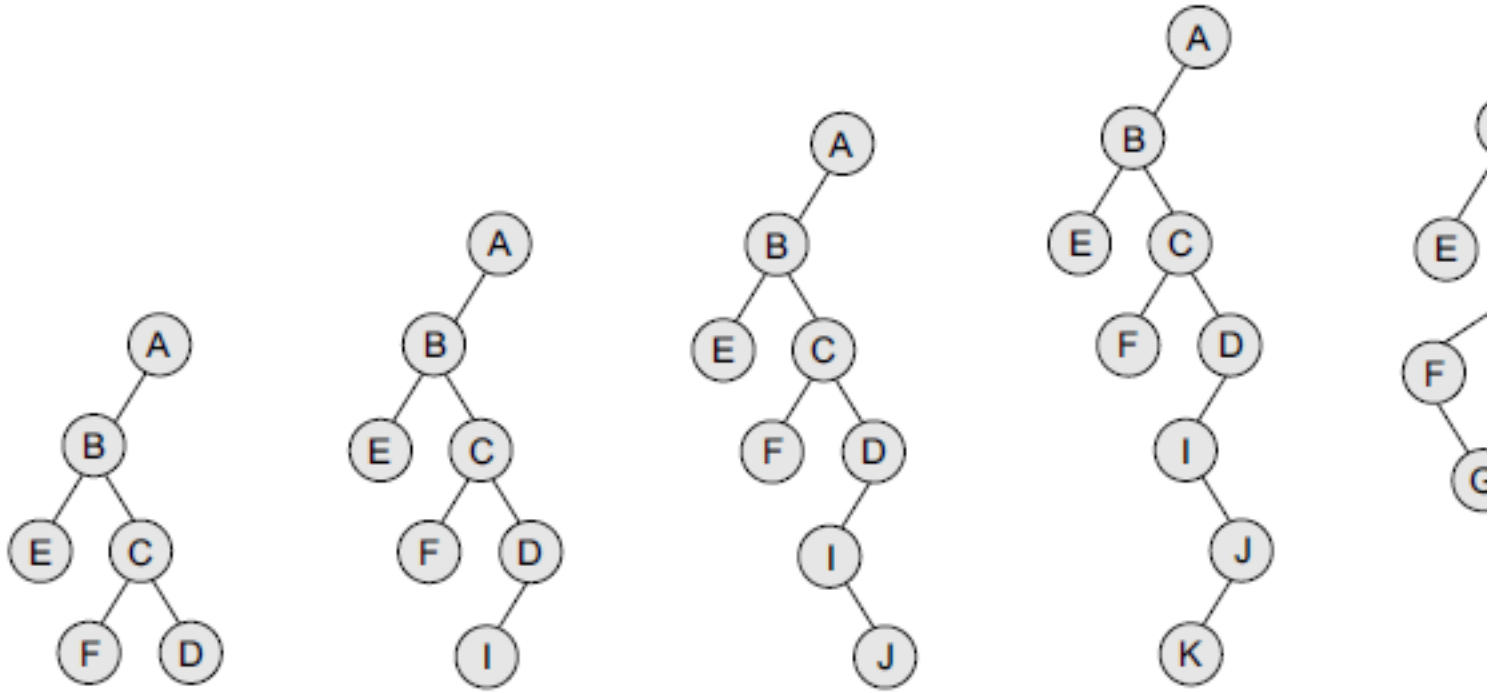
Step 5: Now process node D. Left child of D is I (leftmost child). Th child of D because it has no right sibling in the general tree.

Step 6: Now process node I. There will be no left child of I in the b has no left child in the general tree. However, I has a right sibling J, as the right child of I.

Step 3

Step 7: Now process node J. Left child of J is K (leftmost child). Th child of J because it has no right sibling in the general tree.

Genel Bir Ağaçtan İkili Bir Ağaç Oluşturma



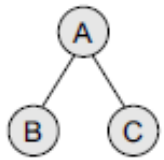
Step 8: Now process all the unprocessed nodes (E, F, G, H, K) in the same fashi binary tree can be given as follows.

İkili Bir Ağacı Geçmek

- İkili bir ağacı dolaşmak, ağaçtaki her düğümü sistematik bir şekilde tam olarak bir kez ziyaret etme işlemidir.
- Elemanların ardışık olarak gezildiği doğrusal veri yapılarından farklı olarak ağaç, elemanların birçok farklı şekilde gezilebildiği doğrusal olmayan bir veri yapısıdır.
- Ağaç gezintileri için farklı algoritmalar mevcuttur.
- Bu algoritmalar düğümlerin ziyaret edilme sırasına göre farklılık göstermektedir.
- Bu bölümde bu algoritmaları tartışacağız.

İkili Bir Ağacı Geçmek

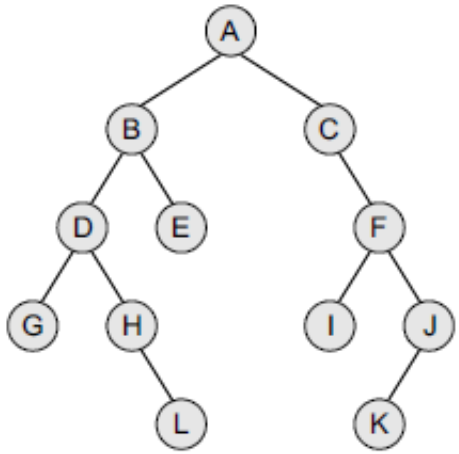
- **Ön Sipariş Gezintisi**
  - Boş olmayan bir ikili ağacı ön sırada dolaşmak için, her düğümde aşağıdaki işlemler yinelemeli olarak gerçekleştirilir.
  - Algoritma şu şekilde çalışır:
    - 1. Kök düğümü ziyaret etmek,
    - 2. Sol alt ağacı dolaşın ve son olarak
    - 3. Sağ alt ağacı dolaşın.
  - Şekilde verilen ağacı ele alalım. Ağacın ön sıra geçişi A, B, C olarak verilmiştir.
  - Önce kök düğüm, sonra sol alt ağaç ve sonra sağ alt ağaç. Ön sıra geçişi, derinlik öncelikli geçiş olarak da adlandırılır. Bu algoritmada, sol alt ağaç her zaman sağ alt ağaçtan önce geçilir.
  - Ön düzendeki 'ön' kelimesi, kök düğümün sol ve sağ alt ağaçlardaki diğer düğümlerden önce erişildiğini belirtir. Ön düzen algoritması ayrıca NLR geçiş algoritması (Düğüm-Sol-Sağ) olarak da bilinir.



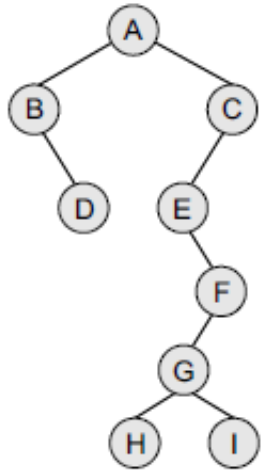

```
Step 1: Repeat Steps 2 to 4 while TREE != NULL
Step 2:            Write TREE -> DATA
Step 3:            PREORDER(TREE -> LEFT)
Step 4:            PREORDER(TREE -> RIGHT)
       [END OF LOOP]
Step 5: END
```

İkili Bir Ağacı Geçmek

- **Ön Sipariş Gezintisi**



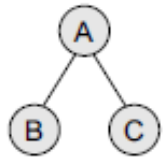
A, B, D, G, H, L, E, C, F, I, J, and K



A, B, D, C, D, E, F, G, H, and I

İkili Bir Ağacı Geçmek

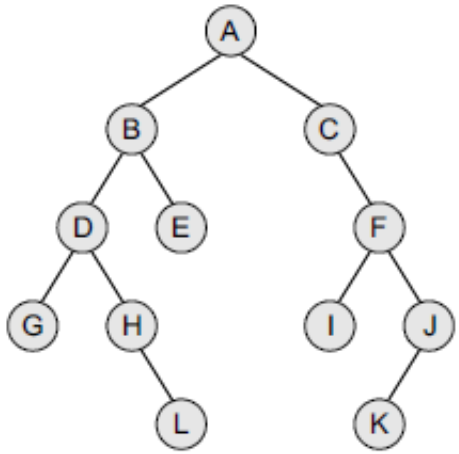
- **Sıralı Geçiş**
  - Boş olmayan bir ikili ağacı sırayla dolaşmak için, her düğümde aşağıdaki işlemler yinelemeli olarak gerçekleştirilir.
  - Algoritma şu şekilde çalışır:
    - 1. Sol alt ağacı dolaşarak,
    - 2. Kök düğümü ziyaret edin ve son olarak
    - 3. Sağ alt ağacı dolaşın.
  - Şekilde verilen ağacı ele alalım. Ağacın sıralı geçişi B, A ve C olarak verilmiştir.
  - Önce sol alt ağaç, sonra kök düğüm ve sonra sağ alt ağaç. Sıralı geçiş simetrik geçiş olarak da adlandırılır.
  - Bu algoritmada, kök düğümden ve sağ alt ağaçtan önce her zaman sol alt ağaç gezilir.
  - Sıralı algoritmadaki 'in' kelimesi, kök düğüme sol ve sağ alt ağaçlar arasında erişildiğini belirtir. Sıralı algoritma aynı zamanda LNR geçiş algoritması (Sol-Düğüm-Sağ) olarak da bilinir.



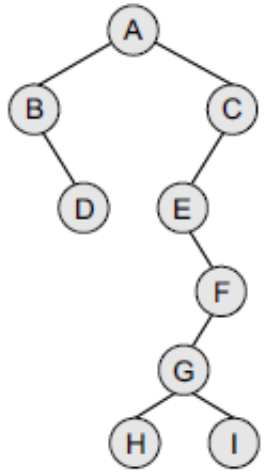

```
Step 1: Repeat Steps 2 to 4 while TREE != NULL
Step 2:            INORDER(TREE -> LEFT)
Step 3:            Write TREE -> DATA
Step 4:            INORDER(TREE -> RIGHT)
       [END OF LOOP]
Step 5: END
```

İkili Bir Ağacı Geçmek

- **Sıralı Geçiş**



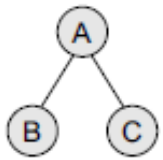
G, D, H, L, B, E, A, C, I, F, K, and J



B, D, A, E, H, G, I, F, and C

İkili Bir Ağacı Geçmek

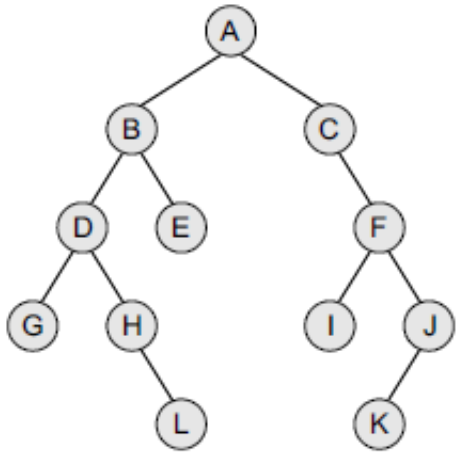
- **Sipariş Sonrası Geçiş**
  - Boş olmayan bir ikili ağacı son sırada dolaşmak için, her düğümde aşağıdaki işlemler yinelemeli olarak gerçekleştirilir.
  - Algoritma şu şekilde çalışır:
    - 1. Sol alt ağacı dolaşarak,
    - 2. Sağ alt ağacı dolaşın ve son olarak
    - 3. Kök düğümü ziyaret etmek.
  - Şekilde verilen ağacı ele alalım. Ağacın post-order geçişi B, C ve A olarak verilmiştir. Önce sol alt ağaç, sonra sağ alt ağaç ve son olarak kök düğüm.
  - Bu algoritmada, sol alt ağaç her zaman sağ alt ağaçtan ve kök düğümden önce geçilir. Post sırasındaki 'post' kelimesi, kök düğüme sol ve sağ alt ağaçlardan sonra erişildiğini belirtir.
  - Post-order algoritması aynı zamanda LRN geçiş algoritması (Sol-Sağ-Düğüm) olarak da bilinir.



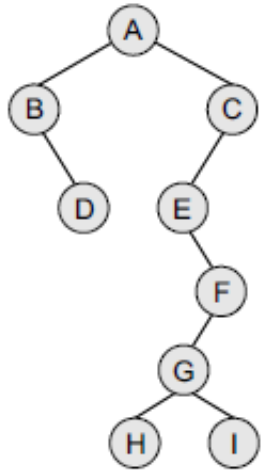

```
Step 1: Repeat Steps 2 to 4 while TREE != NULL
Step 2:            POSTORDER(TREE -> LEFT)
Step 3:            POSTORDER(TREE -> RIGHT)
Step 4:            Write TREE -> DATA
       [END OF LOOP]
Step 5: END
```

İkili Bir Ağacı Geçmek

- **Sipariş Sonrası Geçiş**



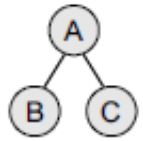
G, L, H, D, E, B, I, K, J, F, C, and A



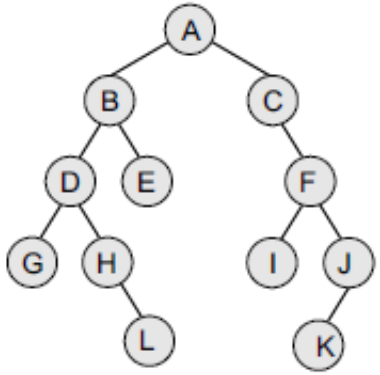
D, B, H, I, G, F, E, C, and A

İkili Bir Ağacı Geçmek

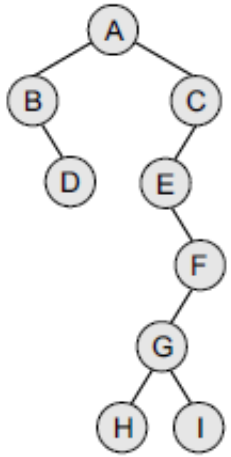
- **Seviye Sırası Gezinme**
  - Düzey sırası geçişinde, bir sonraki düzeye geçmeden önce bir düzeydeki tüm düğümlere erişilir.
  - Bu algoritmaya genişlik öncelikli gezinme algoritması da denir. Şekilde verilen ağaçları göz önünde bulundurun ve bu ağaçların seviye sırasına dikkat edin.



A, B, and C



A, B, C, D, E, F, G, H, I, J, L, and K

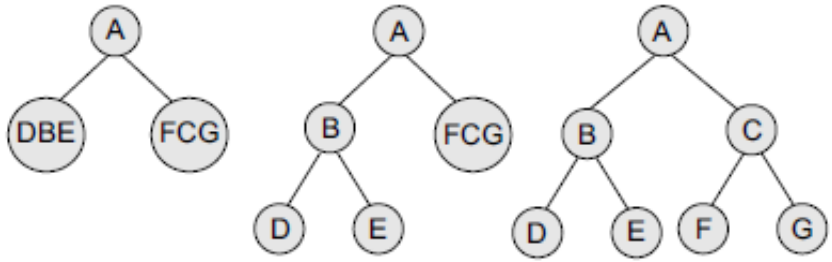


A, B, C, D, E, F, G, H, and I

İkili Bir Ağacı Geçmek

- **Gezinme Sonuçlarından İkili Bir Ağaç Oluşturma**
  - En az iki geçiş sonucu verilirse ikili bir ağaç inşa edebiliriz. İlk geçiş sıralı geçiş olmalı ve ikincisi ön sıralı veya son sıralı geçiş olabilir.
  - Sıralı geçiş sonucu sol ve sağ alt düğümleri belirlemek için kullanılacak ve ön sıra/son sıra kök düğümü belirlemek için kullanılabilir. Örneğin, aşağıda verilen geçiş sonuçlarını ele alalım:

    Sıralı Geçiş: D B E A F C G Ön Sıralı Geçiş: A B D E C F G
  - Burada, sıralı geçiş dizisi ve ön sıralı geçiş dizisi var. Ağacı oluşturmak için aşağıda verilen adımları izleyin:
  - ***Adım 1 Ağacın kök düğümünü belirlemek için ön sipariş dizisini kullanın. İlk öğe kök düğüm olacaktır.***
  - ***Adım 2 Sıralı geçiş dizisindeki kök düğümün sol tarafındaki elemanlar kök düğümün so alt ağacını oluşturur. Benzer şekilde, sıralı geçiş dizisindeki kök düğümün sağ tarafındaki elemanlar kök düğümün sağ alt ağacını oluşturur.***
  - ***Adım 3 Ön sıralı gezinme dizisinden her bir öğeyi yinelemeli olarak seçin ve sıralı gezinme dizisinden sol ve sağ alt ağaçlarını oluşturun.***
  - Ağacın gezinme sonuçlarından oluşturulduğu Şekle bakın.



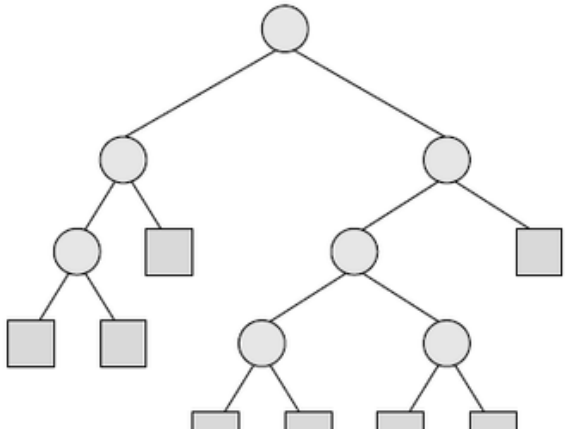
Huffman'ın Ağacı

- Huffman kodlaması, David A. Huffman tarafından geliştirilen, yaygın olarak kayıpsız veri sıkıştırma tekniği olarak kullanılan bir entropi kodlama algoritmasıdır.
- Huffman kodlama algoritması, kaynak karakteri kodlamak için değişken uzunluklu bir kod tablosu kullanır; değişken uzunluklu kod tablosu, kaynak karakterin oluşma olasılığına göre türetilir.
- Huffman algoritmasının arkasındaki temel fikir, daha az yaygın kaynak karakterler için kullanılanlardan daha kısa bit dizileri kullanarak en yaygın karakterleri kodlamasıdır.
- Algoritma, bir dizide saklanan düğümlerden oluşan ikili bir ağaç oluşturarak çalışır.
- Bir düğüm, yaprak düğüm veya dahili düğüm olabilir.
- Başlangıçta, ağaçtaki tüm düğümler yaprak düzeyindedir ve kaynak karakteri ve onun oluşma sıklığını (ağırlık olarak da bilinir) depolar.

Huffman'ın Ağacı

- Dahili düğüm ağırlığı depolamak için kullanılırken ve alt düğümlerine bağlantılar içerirken, harici düğüm gerçek karakteri içerir.
- Geleneksel olarak, '0' sol çocuğu, '1' ise sağ çocuğu takip etmeyi temsil eder.
- n yaprak düğümü olan bitmiş bir ağaçta n – 1 iç düğüm olacaktır.
- Algoritmanın çalışma süresi ağaçtaki yolların uzunluğuna bağlıdır.
- Huffman kodlamasının daha fazla detayına girmeden önce, ağaçtaki yolların uzunluğunun nasıl hesaplanacağını öğrenelim.
- İkili bir ağacın dış yol uzunluğu, kökten harici bir düğüme kadar her yol üzerindeki tüm yol uzunluklarının toplamı olarak tanımlanır. İç yol uzunluğu da aynı şekilde tanımlanır.
- İkili ağacın iç yol uzunluğu, kökten dahili düğüme kadar her yol üzerindeki tüm yol uzunluklarının toplamı olarak tanımlanır.

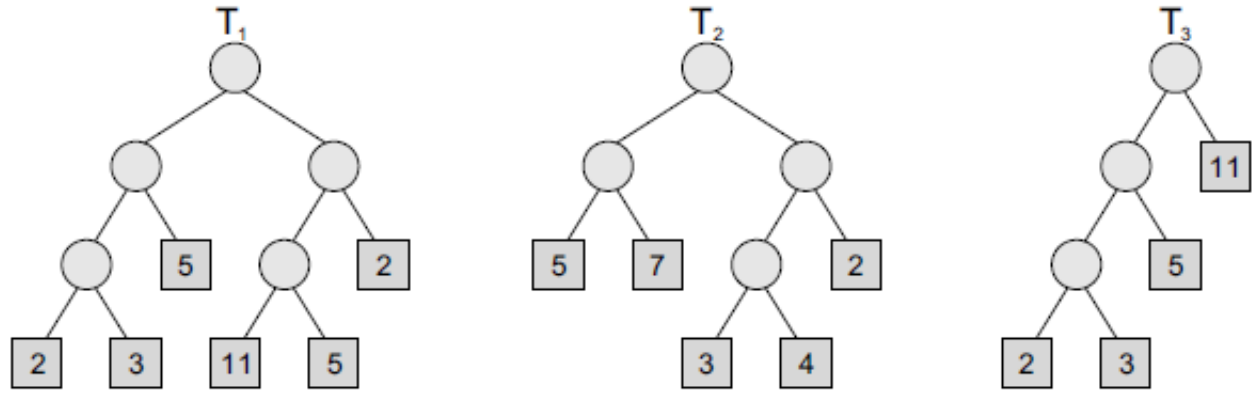
Huffman'ın Ağacı



- Dahili yol uzunluğu, LI = 0 + 1 + 2 + 1 + 2 + 3 + 3 = 12
- Harici yol uzunluğu, LE = 2 + 3 + 3 + 2 + 4 + 4 + 4 + 4 = 26
- LI + 2 * n = 12 + 2 * 7 = 12 + 14 = 26 = LE'ye dikkat edin.
- Böylece, LI + 2n = LE olur; burada n, iç düğümlerin sayısıdır.
- Şimdi eğer ağacın n adet dış düğümü varsa ve her dış düğüme bir ağırlık atanırsa, ağırlıklı yol uzunluğu P, ağırlıklı yol uzunluklarının toplamı olarak tanımlanır.
- Bu nedenle, P = $W_1L_1 + W_2L_2 + ... + WnLn$ burada Wi ve Li, harici bir düğüm olan Ni'nin ağırlığı ve yol uzunluğudur.

Huffman'ın Ağacı

Example 9.9 Consider the trees T_1, T_2, and T_3 given below, calculate their weighted external path lengths.



Binary tree

Solution

Weighted external path length of T_1 can be given as,

$$P_1 = 2{\cdot}3 + 3{\cdot}3 + 5{\cdot}2 + 11{\cdot}3 + 5{\cdot}3 + 2{\cdot}2 = 6 + 9 + 10 + 33 + 15 + 4 = 77$$

Weighted external path length of T_2 can be given as,

$$P_2 = 5{\cdot}2 + 7{\cdot}2 + 3{\cdot}3 + 4{\cdot}3 + 2{\cdot}2 = 10 + 14 + 9 + 12 + 4 = 49$$

Weighted external path length of T_3 can be given as,

$$P_3 = 2{\cdot}3 + 3{\cdot}3 + 5{\cdot}2 + 11{\cdot}1 = 6 + 9 + 10 + 11 = 36$$

Huffman'ın Ağacı

Teknik

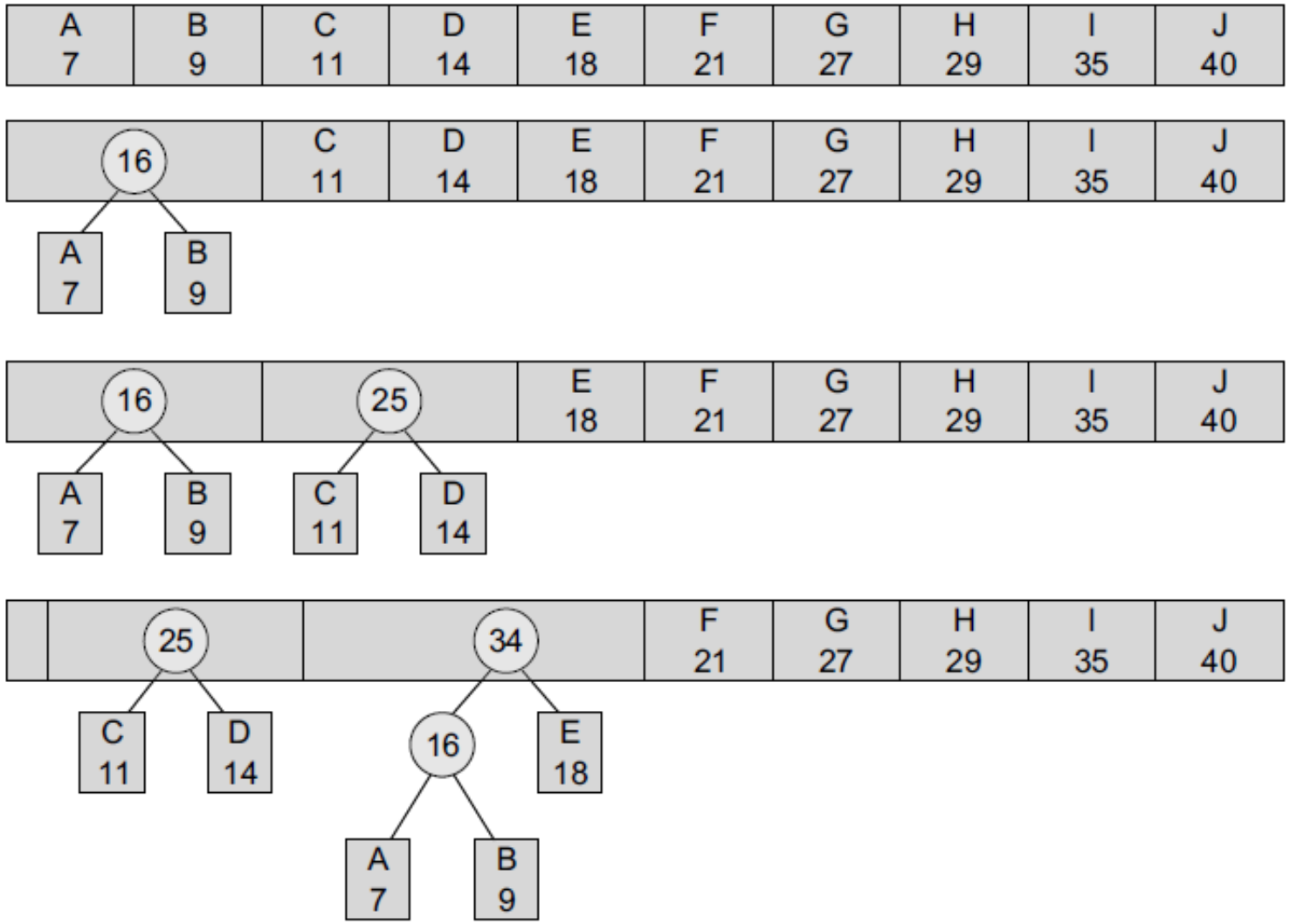
- Verilen n düğüm ve ağırlıkları için, minimum ağırlıklı yol uzunluğuna sahip bir ağaç bulmak için Huffman algoritması kullanılır.
- Süreç esas olarak, en küçük ağırlığa sahip iki düğümün çocukları olan yeni bir düğüm oluşturarak başlar, böylece yeni düğümün ağırlığı çocukların ağırlıklarının toplamına eşit olur.
- Yani, iki düğüm tek bir düğümde birleştirilir. Bu işlem, ağaçta yalnızca bir düğüm kalana kadar tekrarlanır. Yalnızca bir düğümü olan böyle bir ağaca Huffman ağacı denir.
- Huffman algoritması, tüm düğümlerin en düşük ağırlığa sahip düğüme en yüksek önceliğin verildiği bir öncelik sırası kullanılarak uygulanabilir.

```
Step 1: Create a leaf node for each character. Add the character and its w
        of occurrence to the priority queue.
Step 2: Repeat Steps 3 to 5 while the total number of nodes in the queue i
Step 3: Remove two nodes that have the lowest weight (or highest priority)
Step 4: Create a new internal node by merging these two nodes as children
        equal to the sum of the two nodes' weights.
Step 5: Add the newly created node to the queue.
```

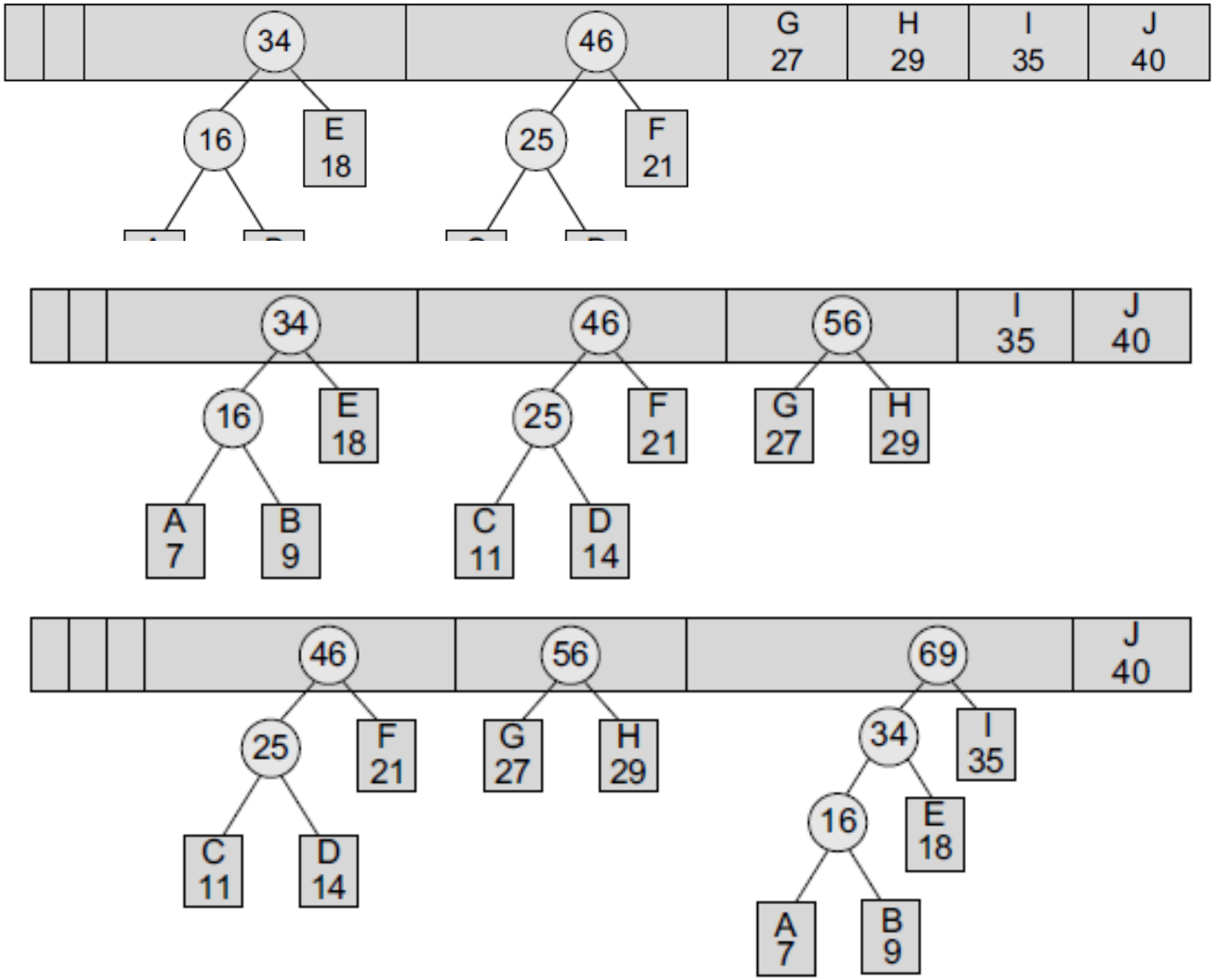
Figure 9.23 Huffman algorithm

Huffman'ın Ağacı

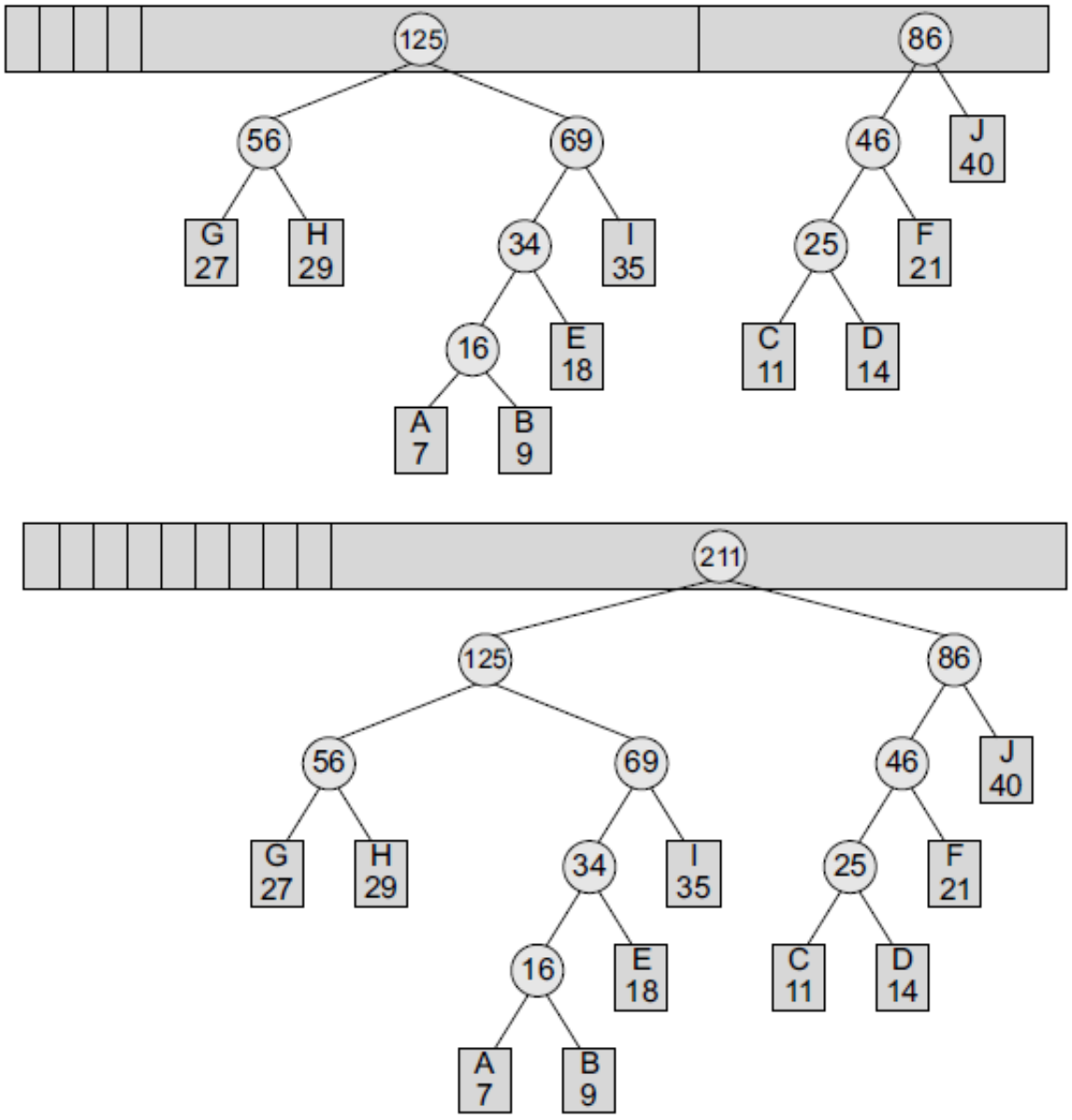
Example 9.10 Create a Huffman tree with the following nodes arranged in a priority queue.



Huffman'ın Ağacı



Huffman'ın Ağacı



Huffman'ın Ağacı

Veri Kodlama

- Verilerimizi (karakterlerimizi) bit kullanarak kodlamak istediğimizde, 2^r karakteri kodlamak için r bit kullanırız.
- Örneğin, r=1 ise, iki karakter kodlanabilir. Bu iki karakter A ve B ise, A 0 olarak kodlanabilir ve B 1 olarak kodlanabilir ve bunun tersi de geçerlidir.
- r=2 ve r=3 kullanılarak kodlanabilen karakter aralığını gösteren Tablo 9.1 ve 9.2'ye bakın.
- Şimdi, ABBBBBBAAAACDEFGGGGH veri dizisini kodlamamız gerekirse, karşılık gelen kod şu şekilde olacaktır:
  00000100100100100100100000000000010011100101110110110110111

Table 9.1 Range of characters that can be coded using r = 2

Code	Character
00	A
01	B
10	C
11	D

Table 9.2 Range of characters that can be coded using r = 3

Code	Character
000	A
001	B
010	C
011	D
100	E

Huffman'ın Ağacı

Veri Kodlama

- Bu kodlama şeması sabit uzunlukta bir koda sahiptir çünkü her karakter aynı sayıda bit kullanılarak kodlanmaktadır.
- Bu kodlama tekniği basit olmasına rağmen, değişken uzunlukta bir kod kullanılarak verilerin kodlanması daha verimli hale getirilebilir.
- İngilizce bir metin yazdığımızda, tüm karakterlerin sık kullanılmadığını fark etmiş olabilirsiniz. Örneğin, a, e, i ve r gibi karakterler w, x, y, z vb. karakterlerden daha sık kullanılır.
- Yani temel fikir, sık görülen karakterlere daha kısa, daha az görülen karakterlere ise daha uzun bir kod atamak.
- Aynı veriyi kodlamak için daha az sayıda bit gerektirdiğinden, sabit uzunluklu kodlamaya göre değişken uzunluklu kodlama tercih edilir.

Huffman'ın Ağacı

Veri Kodlama

- Değişken uzunluklu kodlama için önce bir Huffman ağacı oluşturuyoruz. İlk olarak, tüm karakterleri, en düşük sıklıkta görülen karakterin en yüksek önceliğe sahip olduğu bir öncelik sırasına yerleştiriyoruz.
- Daha sonra, önceki bölümde açıklandığı gibi bir Huffman ağacı oluşturun. Şekil 9.24, veri setini kodlamak için kullanılan bir Huffman ağacını göstermektedir.
- Huffman ağacında daireler, alt düğümlerinin kümülatif ağırlıklarını içerir.
- Her sol dal 0 ile, her sağ dal ise 1 ile kodlanmıştır. Dolayısıyla A, E, R, W, X, Y ve Z karakterleri Tablo 9.3'te gösterildiği şekilde kodlanmıştır.

Table 9.3 Characters with their codes

Character	Code
A	00
E	01
R	11
W	1010
X	1000
Y	1001
Z	1011

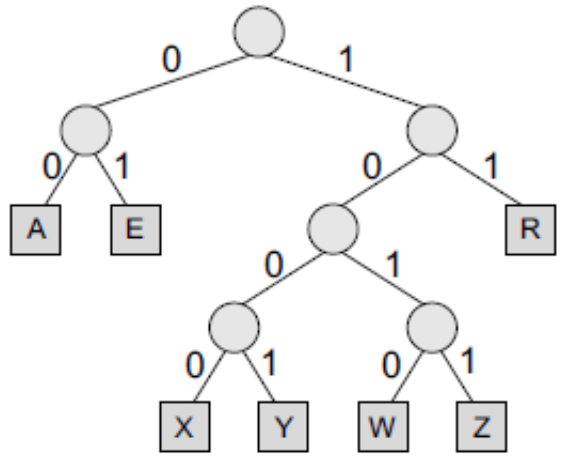


Figure 9.24 Huffman tree