

Ağaç Yapıları (Tree Structures)

İçerik



Temel Kavramlar



Ağaçlarda Dolaşım



İkili Ağaçlar (Binary Trees)



İkili Arama Ağacı (Binary Search Tree ve Temel İşlemler

Bilgisayar bilimlerinde yer alan önemli veri yapılarından bir tanesi de ağaçlardır. Ağaç yapılarını önemli yapan özellikler :



Esneklik : Tek bir amaç için en iyi çözüm ağaç yapısı olmasa da, çoklu çözümler için ağaç yapısının esnekliği bu yapının avantajı olarak sayılabilir.



Etkili Arama: Arama sırasında ağacın bazı dalları budanarak arama yapılması performans artışı sağlar ve bu da verimliliği artırır. Ağaç yapısı genellikle bilginin istenilen bölümüne konumlanmak için genellikle filtreleme aracı olarak kullanılır.



Doğal Temsil: Bilgiyi temsil etmenin doğal bir yoludur.

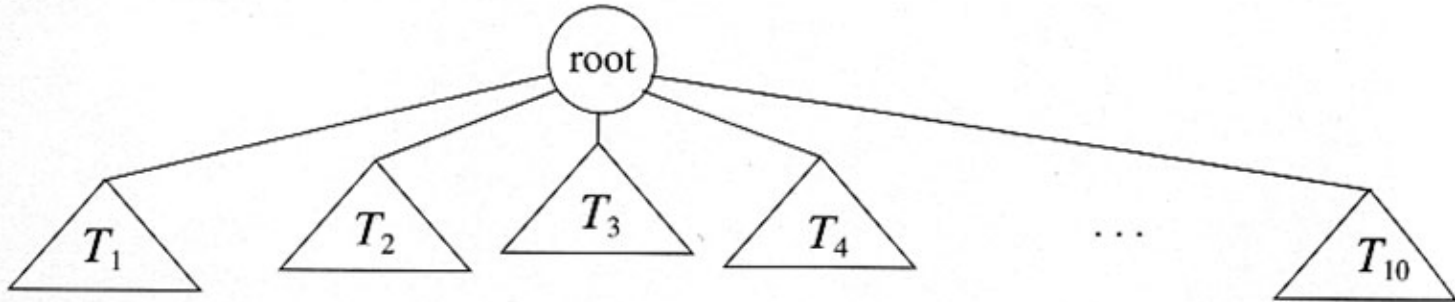


Ağaç yapısı bir uygulamayı bazen daha kolay uygulanabilir bir hale dönüştürür.

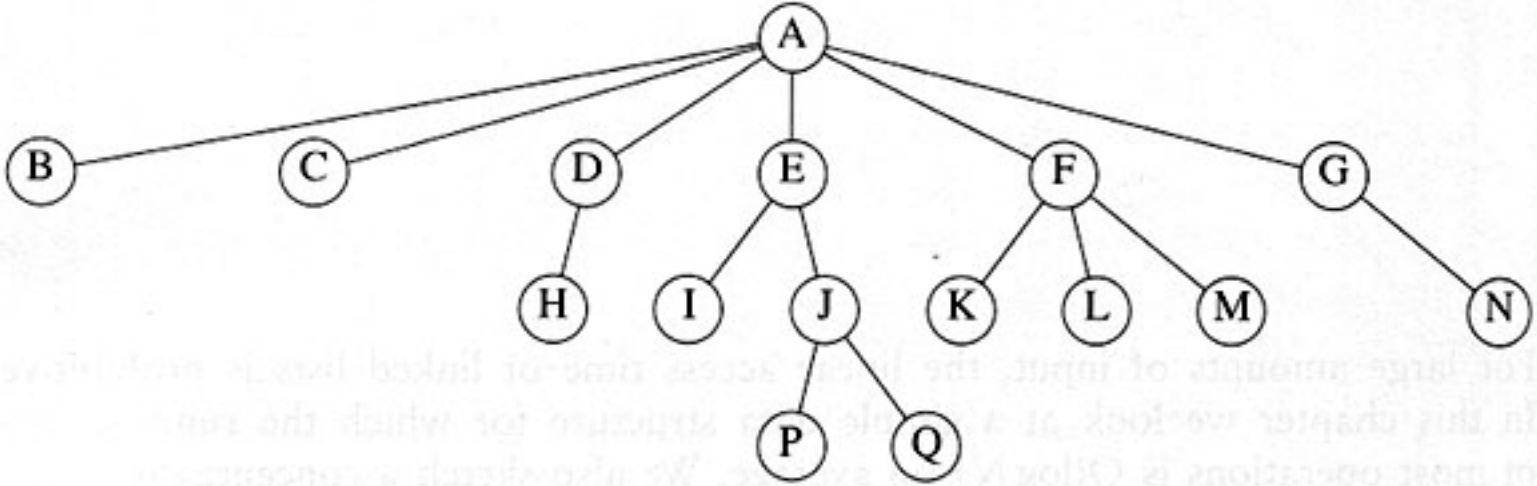
Ağaçlar (Trees)

✓ Ağaçlar düğümlerin koleksiyonudur.

✓ Ağaç, eğer boş değilse birbirinden farklı r tane düğüme, T_1 , T_2 ve T_n ile ifade edilebilecek olan alt ağaçlara sahiptir.



Bazı Temel Kavramlar



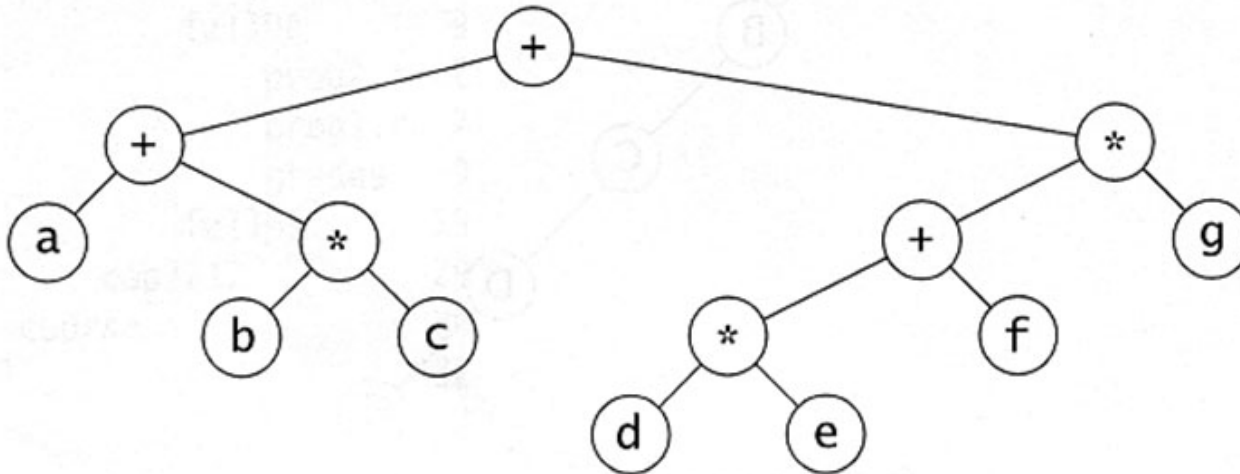
- ✓ **Çocuk ve Aile** : Kök (root) haricindeki her düğümün bir ailesi vardır.
- ✓ **Yaprak**: Herhangi bir çocuğu bulunmayan düğümlerdir.
- ✓ **Kardeş**: Aynı aileye sahip olan düğümlerdir.

Bazı Temel Kavramlar (Devam)

- ✓ **Yol(Path)** : Ardışık kenarlardır.
- ✓ **Yol Uzunluğu (Length of Path)** : Yol içerisindeki kenarların toplamıdır.
- ✓ **Bir düğümün derinliği (Depth of a node)** : Kökten düğüme kadar olan eşsiz yolun uzunluğudur.
- ✓ **Bir düğümün boyu (Height of a node)** : Bir düğümden yaprağa kadar olan en uzun yolun uzunluğudur. Yaprakların boyu 0'dır.

Bir ağacın boyutu = Kök düğümün yüksekliği
= En dipteki yaprağın derinliğidir.
- ✓ **Baba-Oğul (Ata-Torun)** : Eğer n_1 'den n_2 'ye doğru bir yol var ise, n_2 n_1 'in babası (atası); n_1 ise n_2 'nin oğulu veya torunudur.

Örnek



Expression tree for $(a + b * c) + ((d * e + f) * g)$



Bu yapıda operandlar yaprakları, operatörler ise dahili düğümleri oluşturmaktadır.

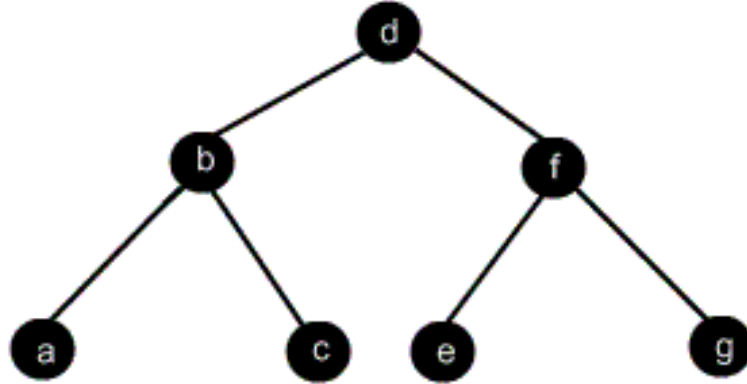
Ağaç İçerisinde Hareket (Traversal)



Ağaç yapısı üzerinde herhangi bir düğüme erişme sürecimize ağacı gezmek (traverse) denir. Bir ağacı en çok bilinen üç değişik yöntemle gezebiliriz :

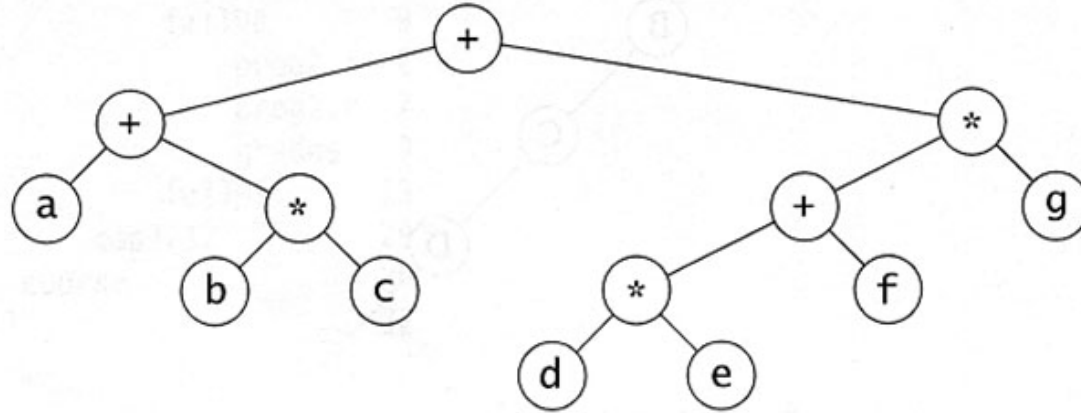
- i) Kökten başlayarak (Preorder)
- ii) Sondan başlayarak (Postorder)
- iii) Sıralı (Inorder)

Kökten Başlayarak Dolaşım (Preorder Traversal)



İlk adımda köke uğranır(d). Sol alt ağaç kökten başlayarak dolaşılır (b-a-c). Son adımda sol alt ağaç kökten başlayarak dolaşılır (f-e-g). Sonuçta ağaç sırasıyla d-b-a-c-f-e-g düğümlerine erişilerek gezilir.

Örnek

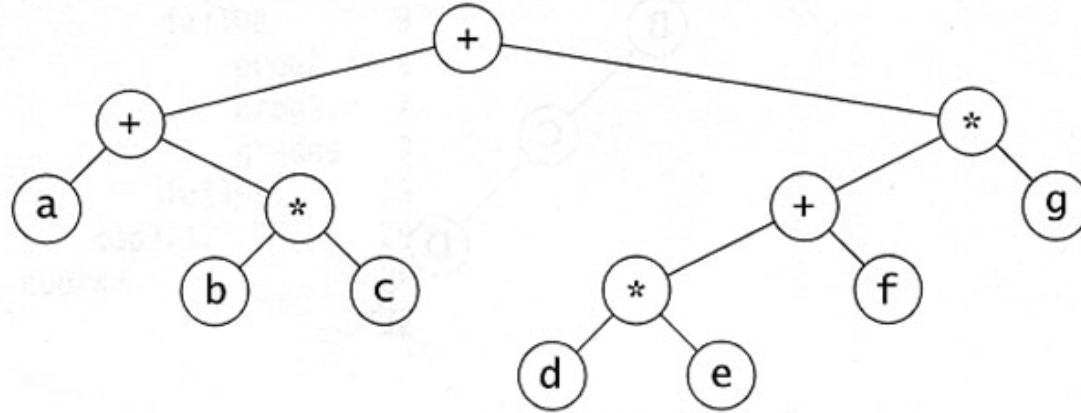


Expression tree for $(a + b * c) + ((d * e + f) * g)$



Yukarıda verilen ağaç yapısında kökten başlayarak (preorder) bir dolaşım izlenirse düğümler hangi sırayla erişilerek gezilmiş olur?

Örnek

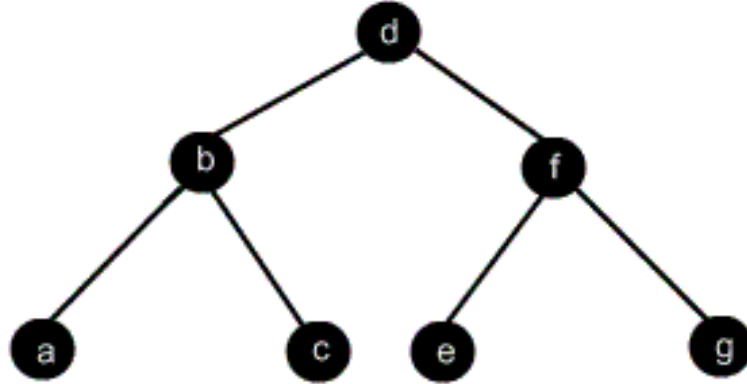


Expression tree for $(a + b * c) + ((d * e + f) * g)$



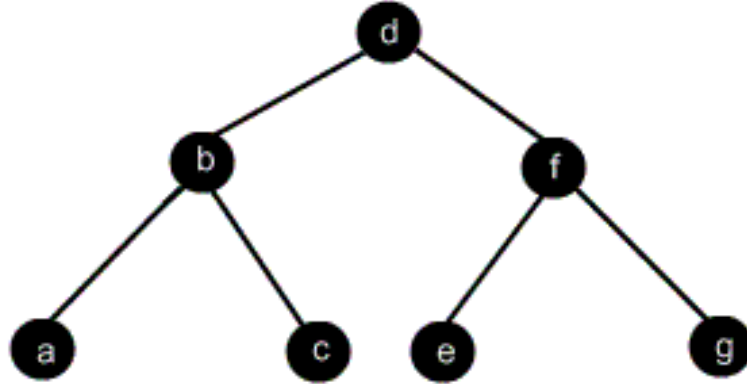
$++a*bc*+*defg$ şeklinde bir sıra izlenir.

Sondan Başlayarak Dolaşım (Postorder Traversal)



İlk adımda sol alt ağaç sondan başlayarak dolaşılır (a-c-b). İkinci adımda sağ alt ağaç sondan başlayarak dolaşılır (e-g-f). Son adımda ise köke uğranır(d). Sonuçta ağaç sırasıyla a-c-b-e-g-f-d düğümlerine erişilerek gezilir.

Sıralı Dolaşım (Inorder Traversal)

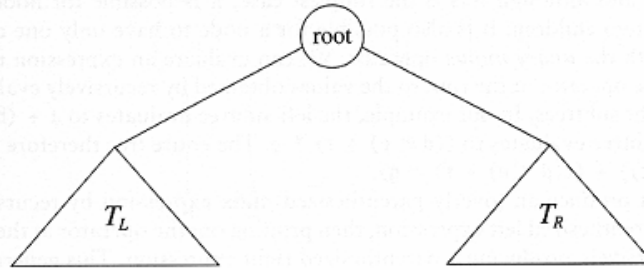


İlk adımda sol alt ağaç sıralı şekilde dolaşılır (a-b-c). Kök düğüme uğranır (d). Son adımda sağ alt ağaç sıralı şekilde dolaşılır (e-f-g). Sonuçta ağaç sırasıyla a-b-c-d-e-f-g düğümlerine erişilerek gezilir.

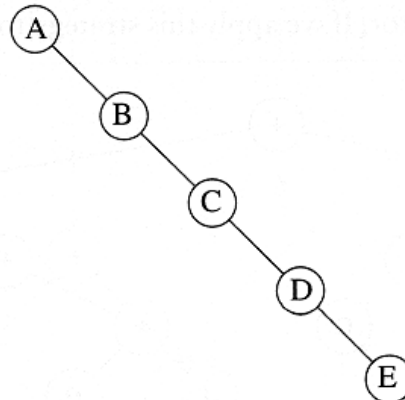
İkili Ağaç Yapıları (Binary Tree Structures)

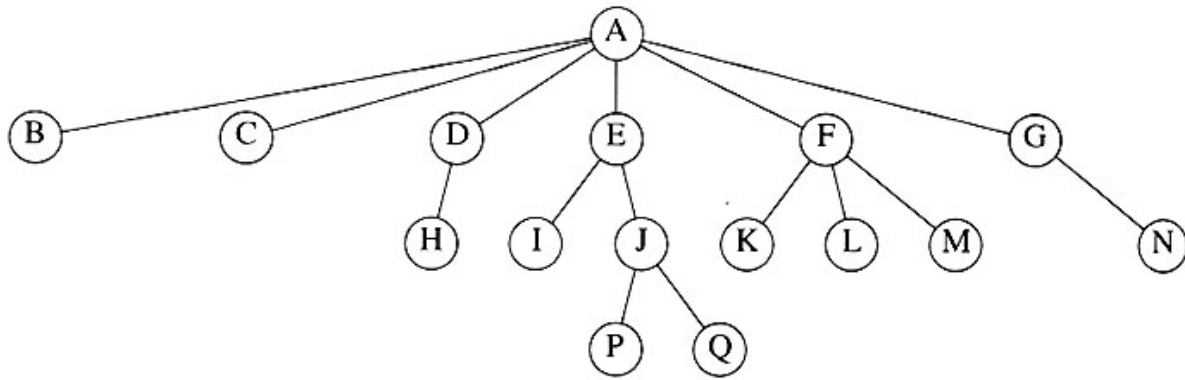
Binary Tree

- ✓ Düğümlerinin en fazla 2 çocuğa sahip olduğu ağaç yapısıdır.

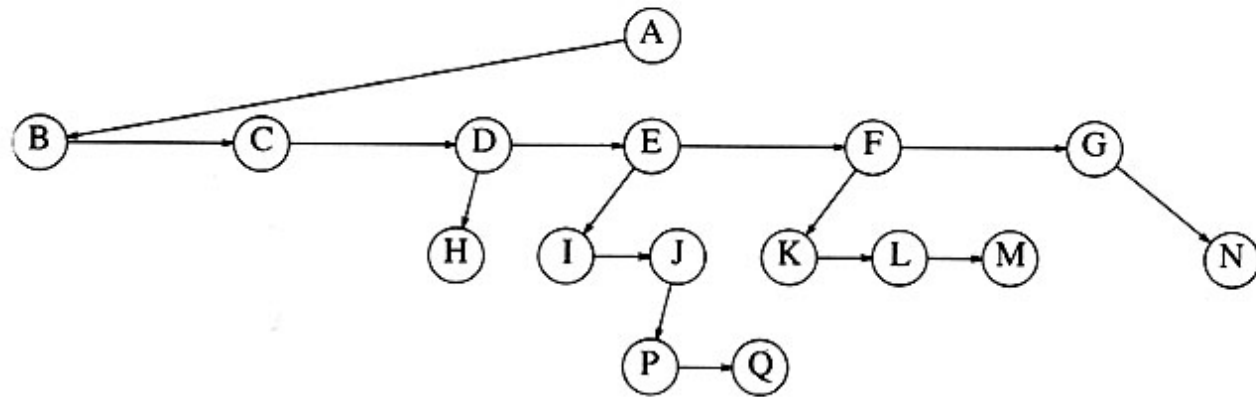


- ✓ Ortalama bir binary tree'nin derinliği genel olarak N'den küçüktür. Fakat en kötü durumda derinliği N-1 kadardır.



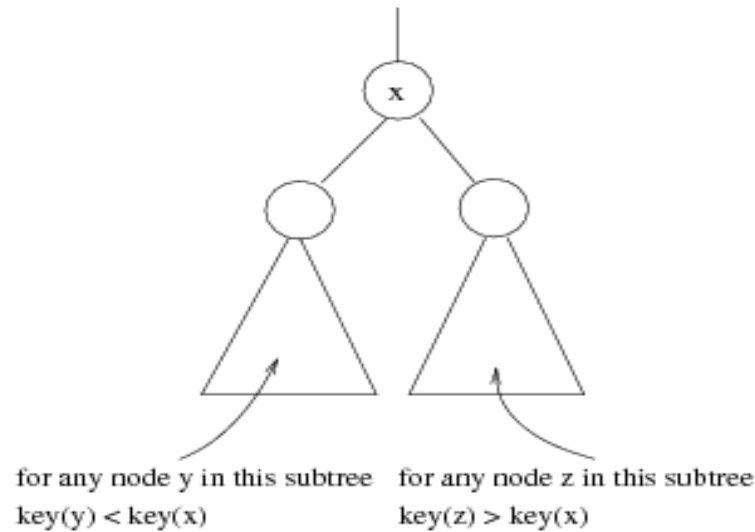


A tree



First child/next sibling representation of the tree

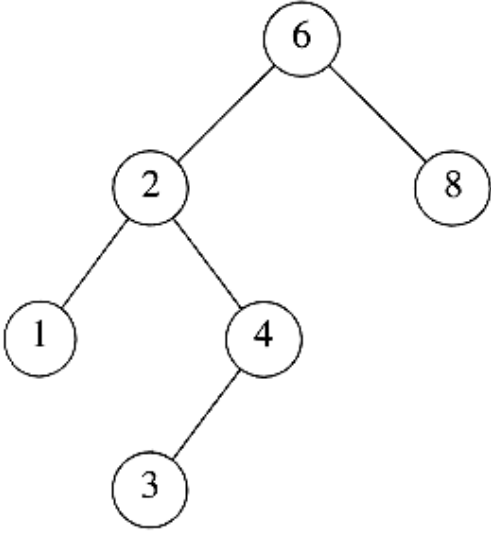
Binary Search Tree (BST)



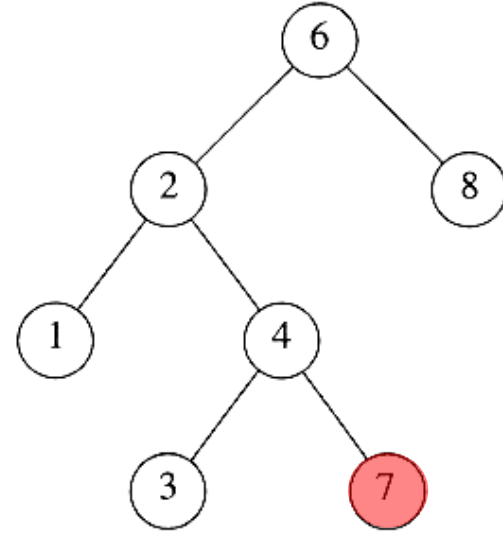
Her X düğümü için;

Sol taraftaki altağaçta yer alan anahtar değerleri her zaman için X 'in değerinden küçüktür.

Sağ taraftaki altağaçta yer alan anahtar değerleri her zaman için X 'in değerinden büyüktür.



a)



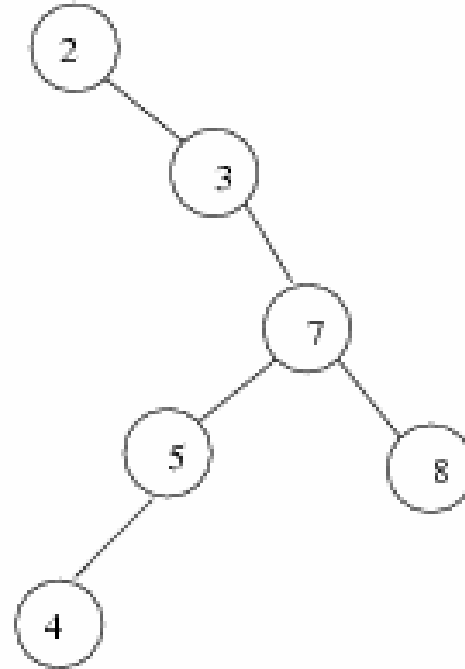
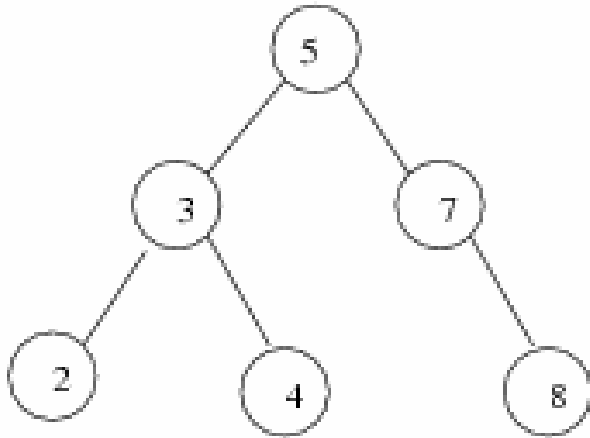
b)



a) Şeklinde yer alan ağaç bir ikili bir arama ağacı iken b) şeklinde yer alan ağaç ise binary tree değildir.

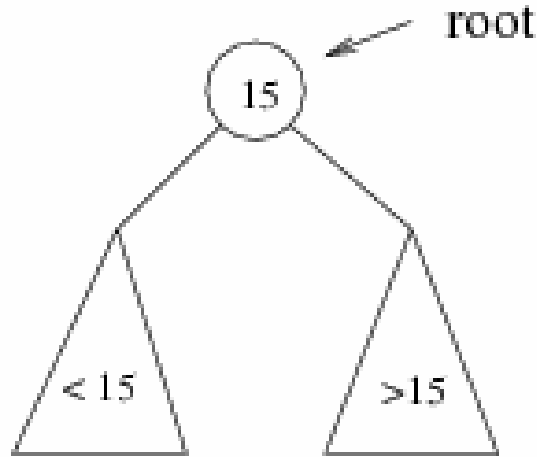


Aynı anahtar setleri farklı ikili ağaçlar oluşturabilir.



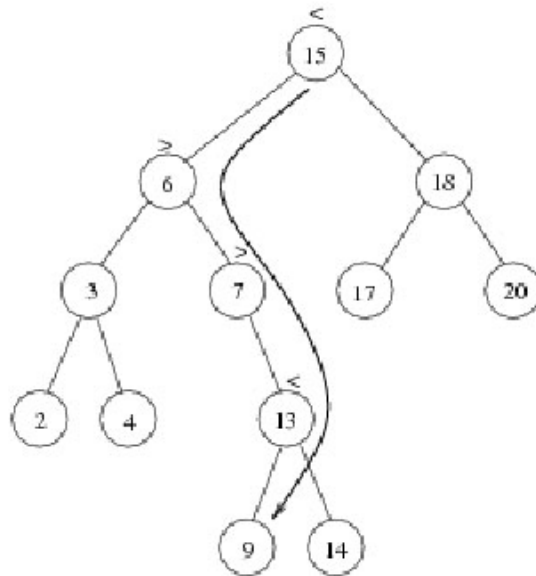
İkili Ağaçlarda Arama

- ✓ Eğer aranılan sayı 15 ise hemen erişilmiş olur.
- ✓ Eğer aranılan sayı 15'den küçük ise, sol taraftaki ağaca,
- ✓ Eğer aranılan sayı 15'den büyük ise, sağ taraftaki ağaca bakılır.



Örnek

Search for 9 ...



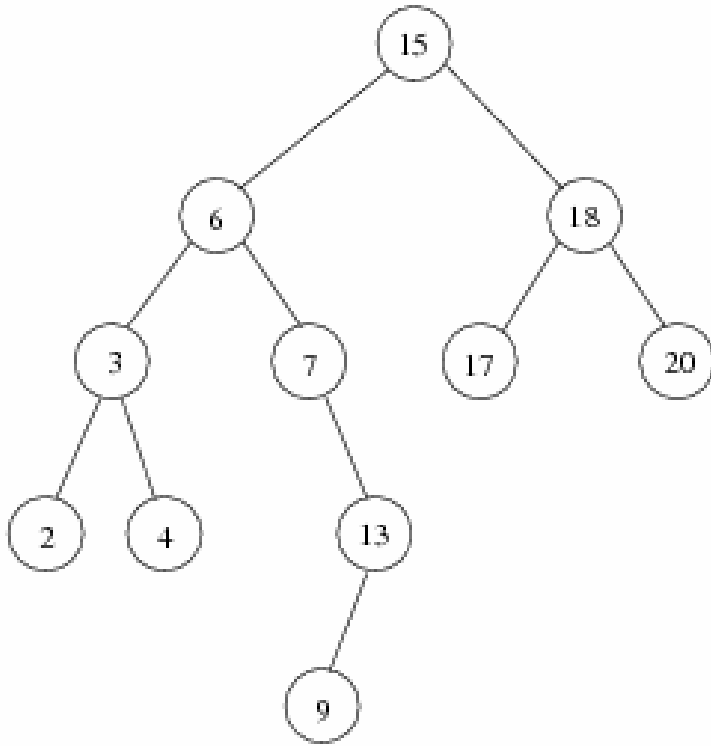
Search for 9:

1. compare 9:15(the root), go to left subtree;
2. compare 9:6, go to right subtree;
3. compare 9:7, go to right subtree;
4. compare 9:13, go to left subtree;
5. compare 9:9, found it!

BST'yi Sıralı Biçimde Dolaşmak



İkili arama ağacını sıralı bir biçimde dolaşmak anahtar değerlerinin sıralı bir şekilde elde edilmesini sağlar.

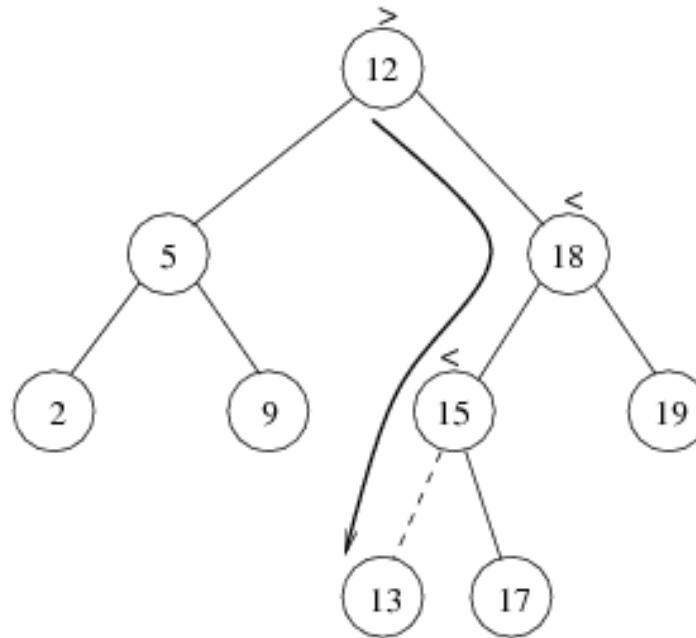


Inorder: 2, 3, 4, 6, 7, 9, 13, 15, 17, 18, 20

BST'ye Anahtar Ekleme



İkili arama ağacına yeni bir anahtar değeri ekleneceği zaman izlenecek adımlar herhangi bir anahtarı bulmak için izlenecek olan adımlar ile aynıdır. **Örn:** Aşağıdaki ağaca 13 değeri eklenmesi durumunda;

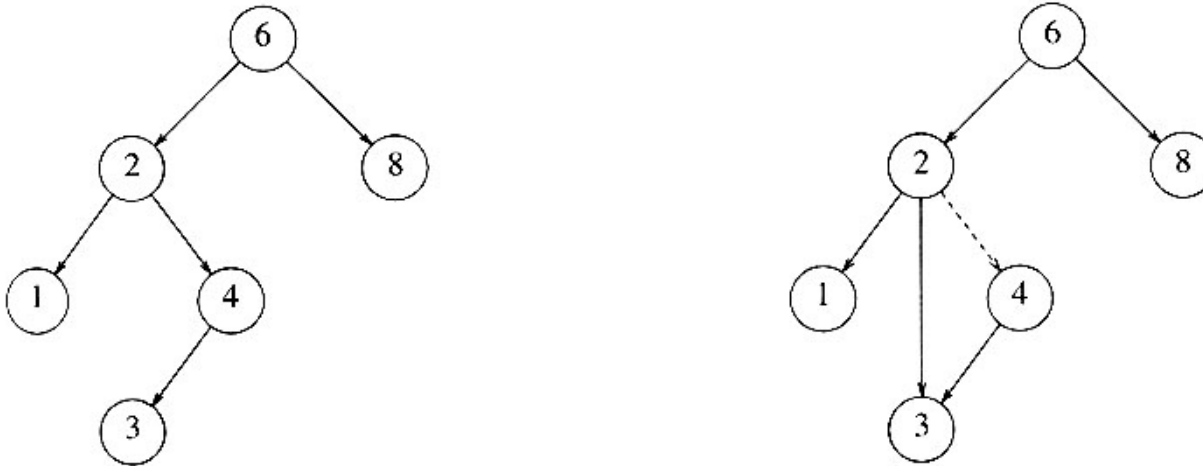


BST'ye Anahtar Silmek



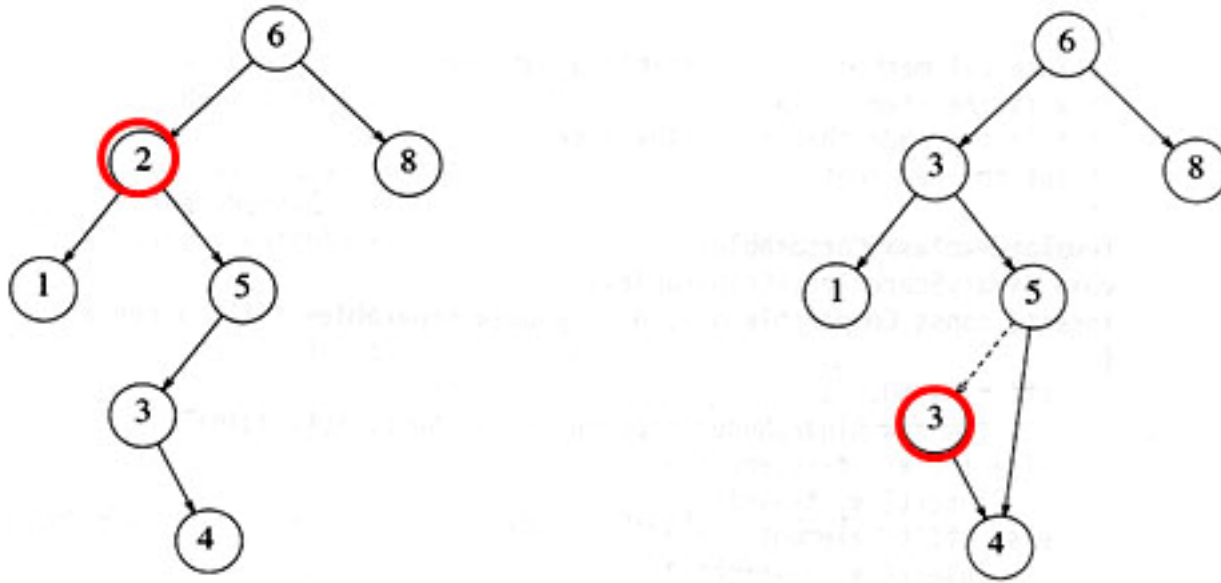
İkili arama ağacında herhangi bir düğüm silindiğinde bu düğüme bağlı olan çocukların dikkate alınması gerekmektedir. 3 farklı durum söz konusudur :

- i) Silinecek olan düğüm yaprak ise silme işlemi hemen gerçekleştirilir.
- ii) Silinecek olan düğüm eğer bir çocuğa sahip ise, o düğümün atasının pointer bilgisini alt düğüme (çocuğa) aktarmak gerekir.



Deletion of a node (4) with one child, before and after

- iii) Silinecek olan düğüm eğer iki çocuğa sahip ise, silinen düğümün yerine o düğümün sağ alt ağacındaki en küçük anahtar değeri getirilir. Daha sonra ise sağ alt ağaçtan taşınan anahtar değeri silinir.



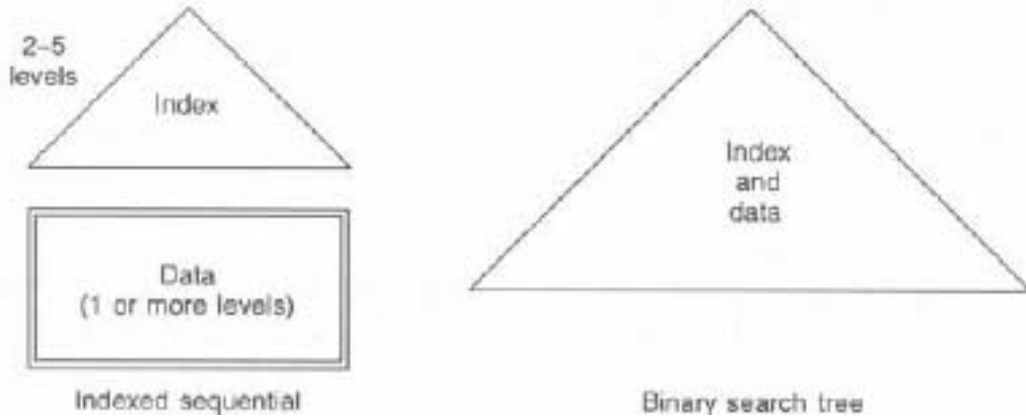
Deletion of a node (2) with two children, before and after

Dosya Yapılarında İkili Arama Ağaçlarının Kullanılması

✓ Indexed Sequential File Organization ağaç yapısını kullanmaktadır. Bu ağaç yapısı hem sıralı hem de rastgele olan erişimlerde kullanılmaktadır.

✓ **Index sequential dosyalarda ağaç sadece index amacı için kullanılır. Kayıtlar yapraklarda saklanır.**

✓ İkili arama ağaçlarında ise bilgi hem yapraklarda hem de dahili düğümlerde saklanmaktadır.



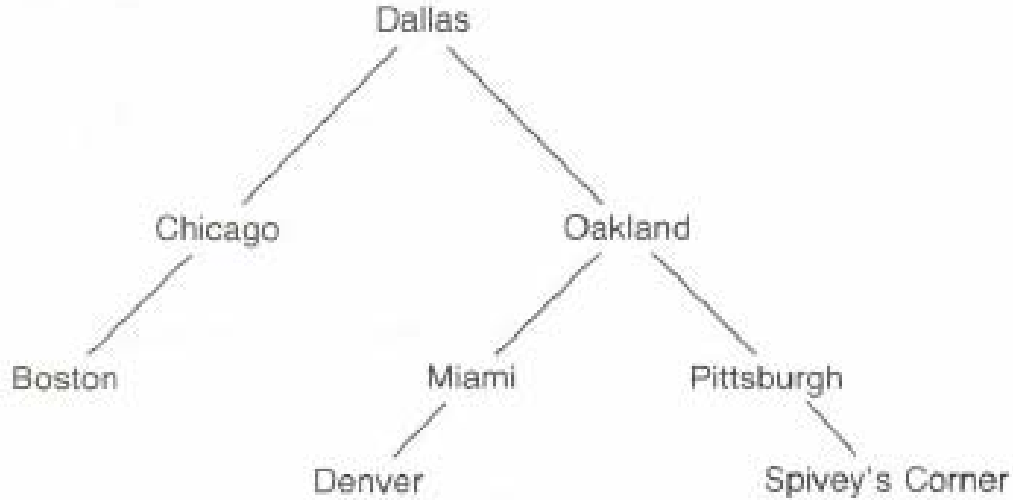
✓ Saklanan bilgi arttıkça ikili arama ağacının yüksekliği artar ve sonuç itibarıyla performansta düşüşler görülür.

✓ İkili arama ağaçlarında Branching Factor 2'dir. Index sequential dosyalarda ise branching factor daha fazla olduğu için çok sayıda bilginin saklandığı durumlarda sıralı ve doğrudan erişimlerde performans yüksektir.

Örnek



Örnek: Dallas, Oakland, Pittsburg, Miami, Chicago, Denver, Boston ve Spivey's Corner kayıtlarını ikili arama ağacında saklamak istiyelim.



Bu yapıda herhangi bir kayda ulaşmak için gerekli olan ortalama probe sayısı **2,75**'dir.

- ✓ İkili ağacın yapısı, bilgilerin eklenme sırasına bağlıdır. Alfabetik sırayla ekleme yapıldığında ağacın yapısı bağlı listeye dönüşür. Bu durumda ise eklenmesi sırasında sürekli olarak ağacın **dengelemesi** gerekir.



- ✓ Bu yapıda herhangi bir kayda ulaşmak için gerekli olan ortalama probe sayısı 4,5'dir.

- ✓ Dengeleme işlemi için geliştirilmiş bir diğer ikili ağaç yapısı ise AVL ağacıdır. Bu ağaç performansı arttırmak için sürekli olarak kendisini yeniden düzenlemektedir.

AVL Ağaçları (AVLTree Structures)

İkili Ağaç Yapılarında Denge

- ✓ **AVL (Adelson-Velskii-Landis)** ağacı, ikili arama ağacı olup bir node'un alt ağaçlarının yükseklikleri yaklaşık aynıdır. Bu yüzden aynı zamanda **height balanced tree** olarak da bilinirler.
- ✓ En kötü durumda ikili arama ağacının karmaşıklığı $O(N-1)$ 'dir.
- ✓ İkili arama ağacının boyunun olabildiğince kısa olması arzu edilir.

Dengeli Ağaç (Balanced Tree)

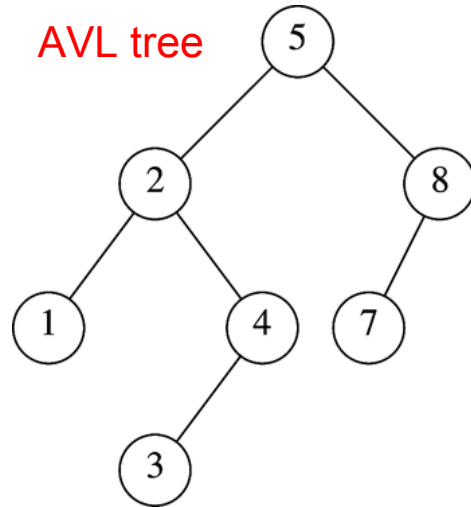
✓ Kök düğümün sağ ve sol altağaçları aynı boyda olmalıdır.

✓ Her bir düğümün sağ ve sol altağaçları aynı boyda olmalıdır.

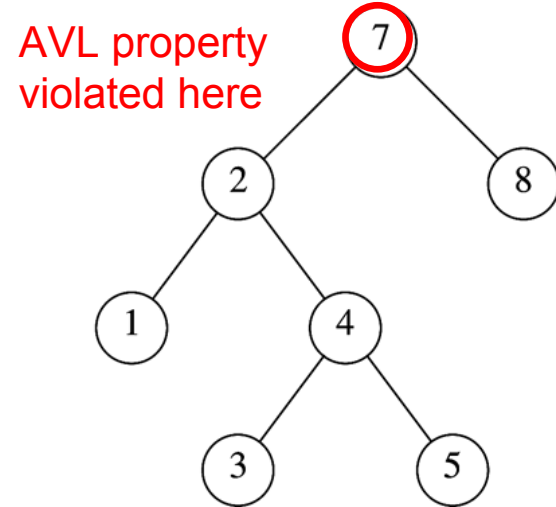
✓ AVL ağacında bilinmesi gereken bir kavram denge faktörüdür.
(Balance Factor)

$$\text{Balance Factor} = \text{Height}_{(\text{right subtree})} - \text{Height}_{(\text{left subtree})}$$

✓ Denge faktörünün -1,0,1 arasında değerler alabilir. (1'den büyük olması söz konusu değildir).



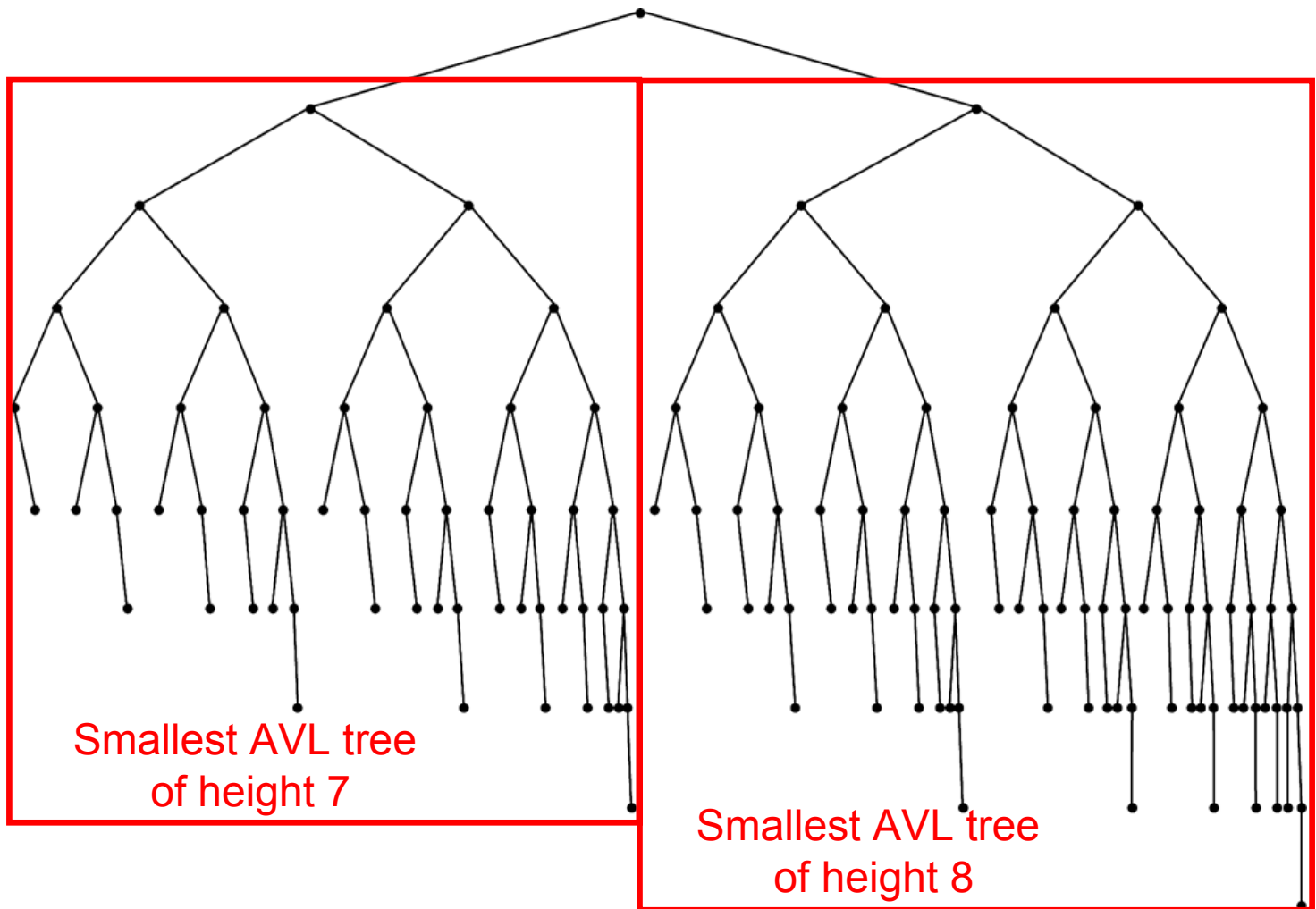
Balance Factor = 2-3 = |-1| = 1



Balance Factor = 1-3 = |-2| = 2

✓ Sol taraftaki ağaç dengeli bir yapı sergilerken (balance-factor=1); sağ kesimdeki ağaç ise dengesizdir (balance-factor =2)

✓ Dikkat edilmesi gereken önemli bir nokta herhangi bir düğümün denge faktörü hesaplanırken sol ve sağ ağaçların yüksekliklerinin belirlenmesidir. Yoksa herhangi bir kayıt için düğümlerin sayılması değildir.

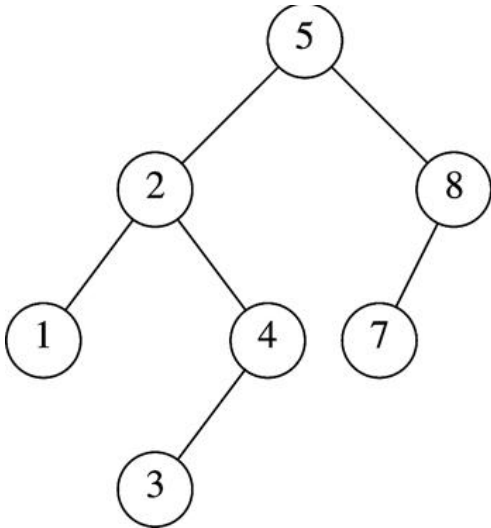


Smallest AVL tree of height 9

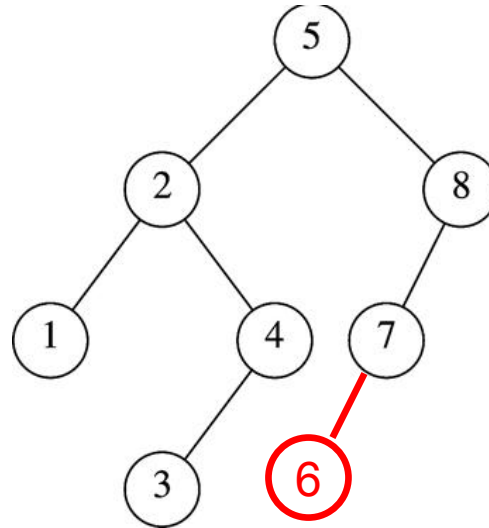
AVL Ağaçlarında Anahtar Ekleme

✓ Herhangi bir kaydın eklenmesi normal ikili ağaçta olduğu gibidir. Fakat, ekleme işleminde ekleme yapılan düğümden köke kadar olan yol üzerindeki bütün düğümlerin denge faktörleri hesaplanır.

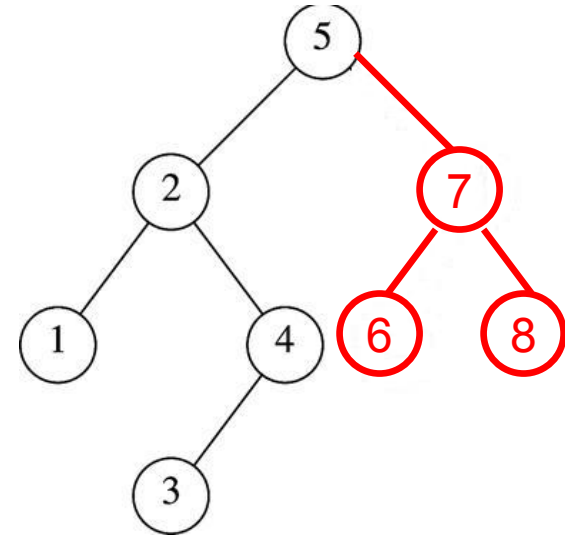
✓ Eğer sonuç ağaç dengeli bir yapıda değil ise, ağacı dengeli bir hale getirmek için 1 veya 2 döndürme (Rotation) gerekir.



Original AVL tree



Insert 6
Property violated



Restore AVL property

Dengeleme İşlemi





Dengelenecek olan düğüm x olarak gösterilsin. Bu durumda 4 farklı durum söz konusudur.

- i) Yeni kayıt, x düğümünün sol alt ağacının sol çocuğuna eklenebilir.
- ii) Yeni kayıt, x düğümünün sağ alt ağacının sol çocuğuna eklenebilir.
- iii) Yeni kayıt, x düğümünün sol alt ağacının sağ çocuğuna eklenebilir.
- iv) Yeni kayıt, x düğümünün sağ alt ağacının sağ çocuğuna eklenebilir.

Görüldüğü gibi 1. durum ile 3 durum ve 2. durum ile 4 durum birbirlerinin simetrikleridir.

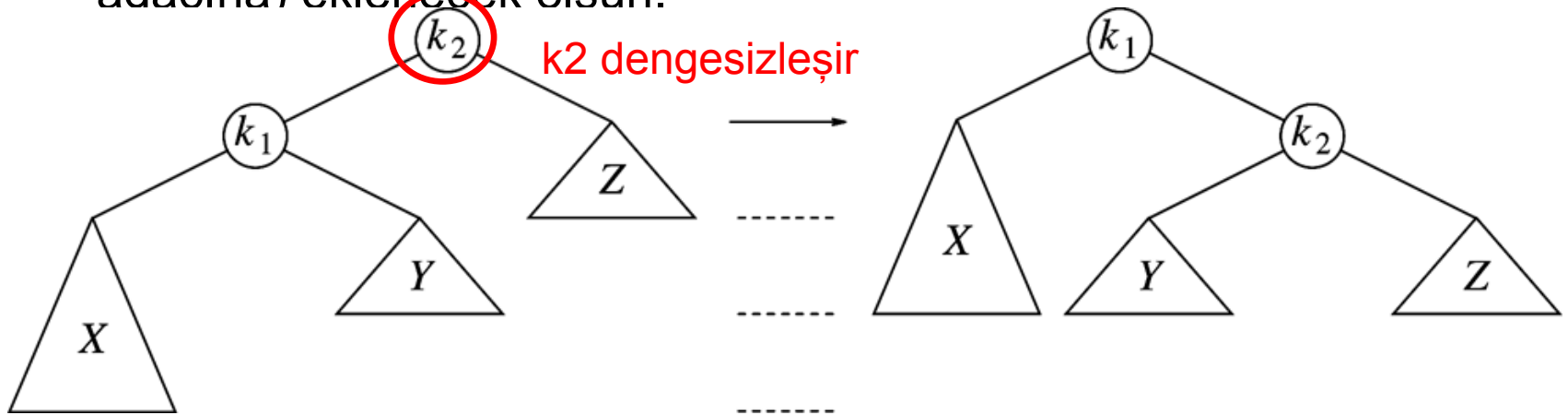
Döndürme İşlemi (Rotations)

-  Eğer ekleme işlemi ağacı dış kısmında oluyorsa (sağ ağacın sağına veya sol ağacın soluna gibi) **tek bir döndürme** yeterlidir.
-  Eğer ekleme işlemi ağacı iç kısmında oluyorsa (sağ ağacın soluna veya sol ağacın sağına gibi) **tek bir döndürme** yeterlidir.

I) Tek Sefer Döndürme (Single Right Rotation)



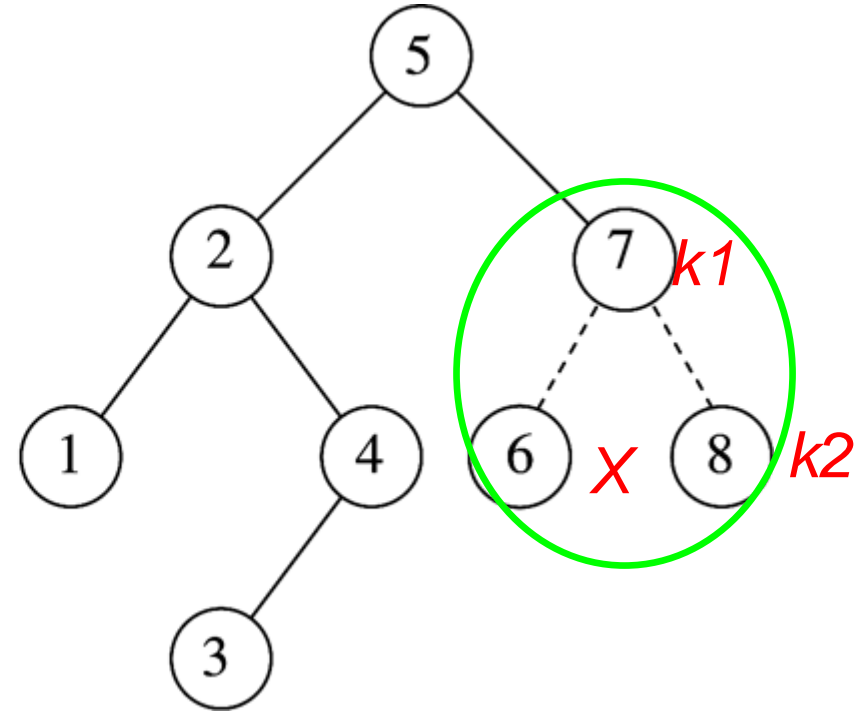
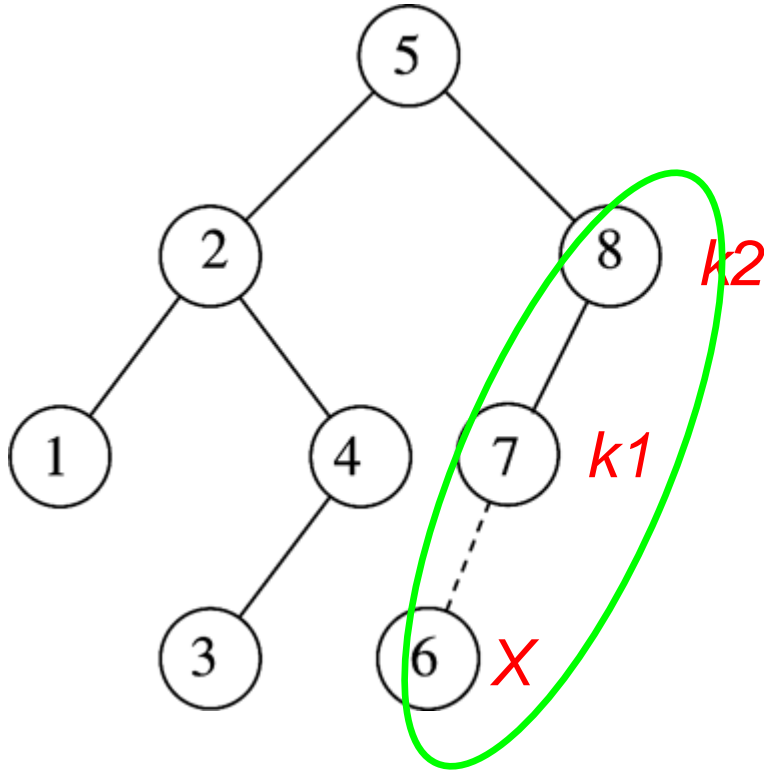
Örn: Eklenecek olan yeni kayıt sol alt ağacın sol kesimine (X alt ağacına) eklenecek olsun.



Bu durumda, sağ tarafa doğru tek bir döndürme işlemi gerçekleştirilir. (Single Right Rotation –SRR). Bu işlem daha önce bahsedilen durumlardan birincisine örnek olarak verilebilir.



Eklenecek olan yeni kayıt sol alt ağacın sol kesimine eklenerek kök düğümün dengesini bozduğu durumlarda (Denge faktörünün 1'den fazla olduğu durumlarda) sağa doğru 1 kere döndürme yapılarak ağaç dengelenir.

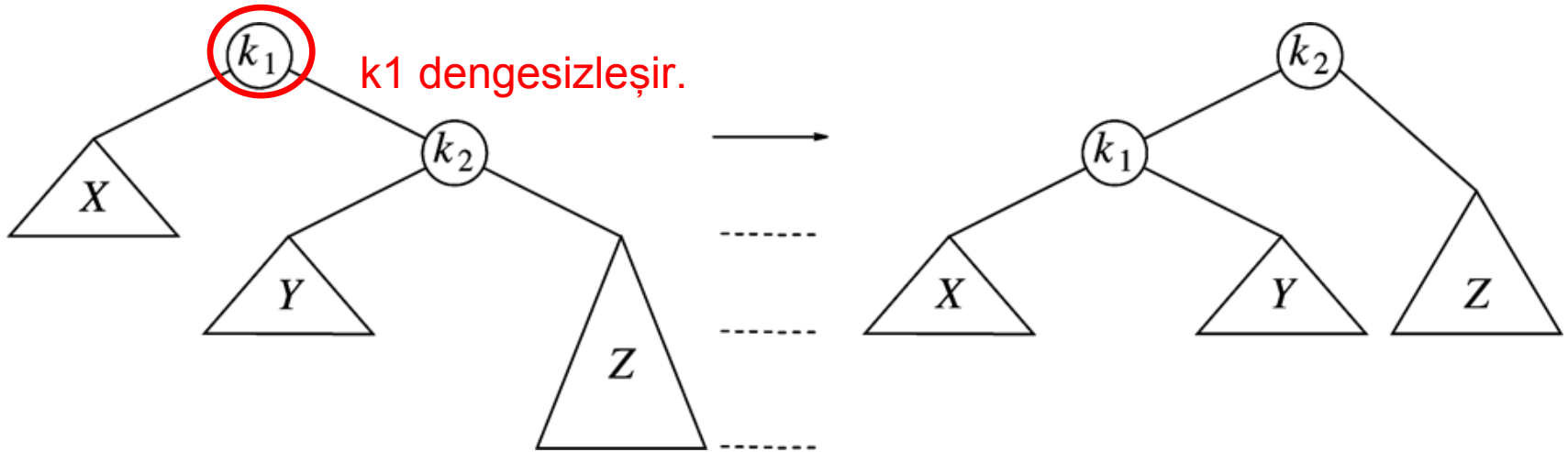


6 anahtarı eklendiğinde 8 anahtarının bulunduğu durum dengesizleştikten ağaç sağa doğru 1 defa döndürülür.

IV) Tek Sefer Döndürme (Single Left Rotation)



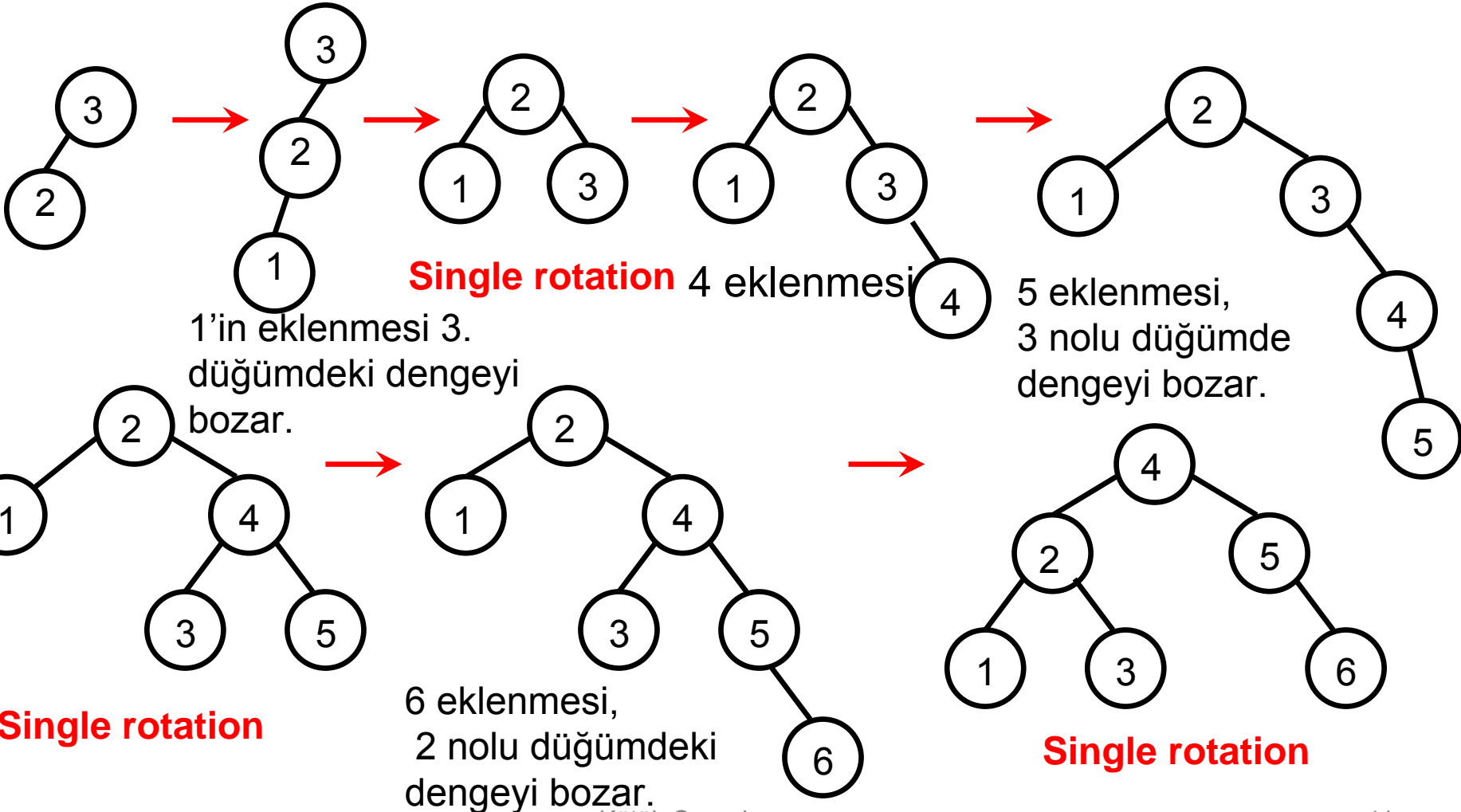
Örn: Eklenecek olan yeni kayıt sağ alt ağacın sağ kesimine (Z alt ağacına) eklenecek olsun.



Bu durumda, sol tarafa doğru tek bir döndürme işlemi gerçekleştirilir. (Single Left Rotation –SLR). Bu işlem daha önce bahsedilen durumlardan dördüncüsüne örnek olarak verilebilir.

Örnek

Aşağıdaki işlem adımları tek seferlik döndürme işlemine (SRR-SLR) örnektir. Örnekte 3, 2, 1, 4, 5, 6 anahtarlarını AVL ağacına ekleyelim.



Single rotation

Single rotation 4 eklenmesi

5 eklenmesi,
3 nolu düğümde
dengeyi bozar.

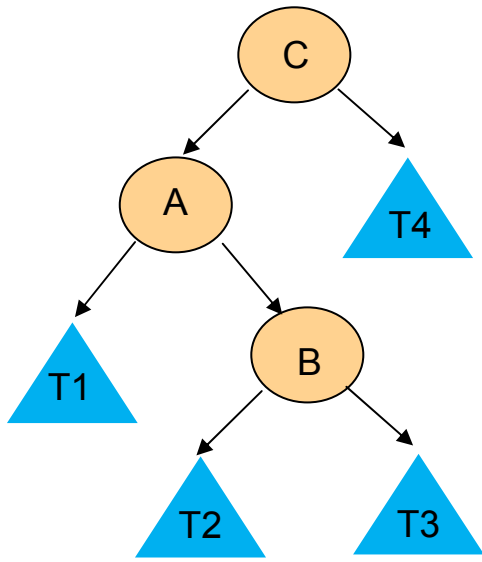
6 eklenmesi,
2 nolu düğümdeki
dengeyi bozar.

Single rotation

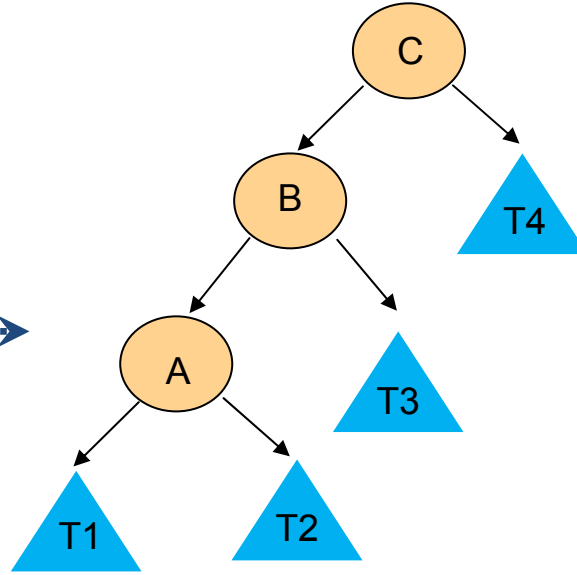
II) İki Sefer Döndürme



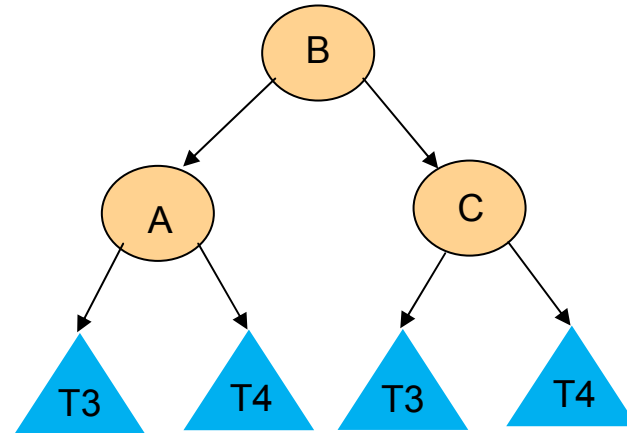
Eklenecek olan yeni kayıt, x düğümünün sağ alt ağacının sol çocuğuna eklendiğinde tek bir seferlik bir döndürme işlemi gerçekleştirmek ağacın dengeye gelmesini sağlamayacaktır. Bu yüzden 2 aşamalı bir döndürme işlemi gerçekleştirilmelidir.



SLR
>



SRR
↓

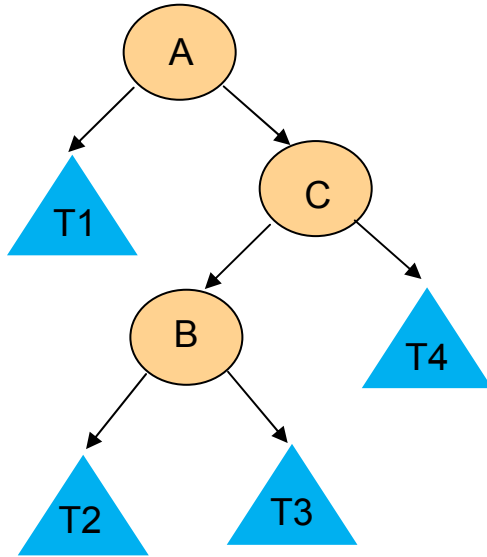


C düğümü sola doğru dengesiz bir durumdadır. A düğümünün sağ kesimi ise daha ağırdır. Ağacı dengeli AVL ağacına dönüştürmek için ard arda SLR ve SRR dönüşümleri yapılır.

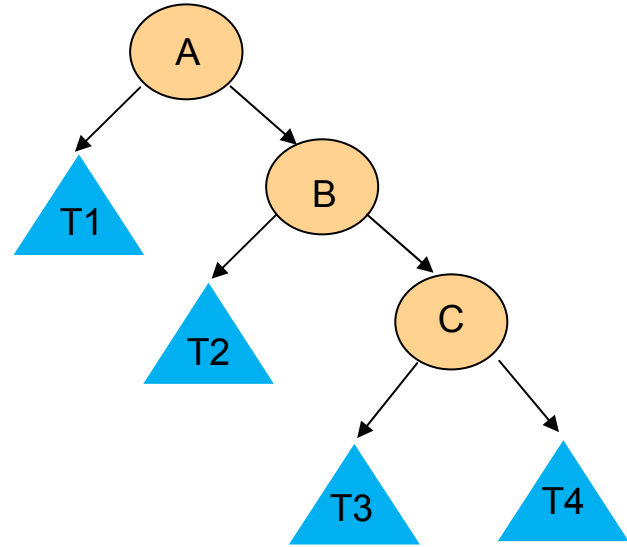
III) İki Sefer Döndürme



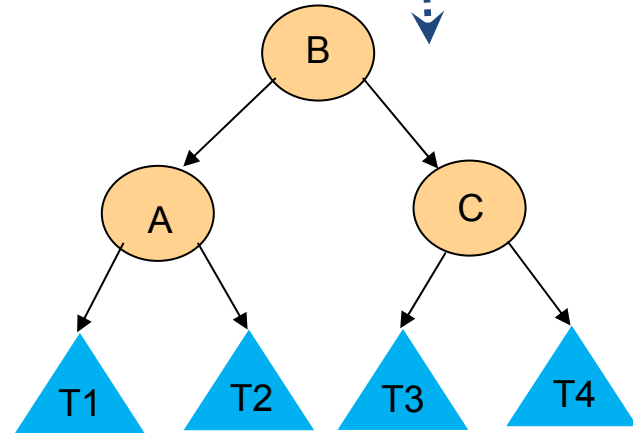
Eklenecek olan yeni kayıt, x düğümünün sol alt ağacının sağ çocuğuna eklendiğinde tek bir seferlik bir döndürme işlemi gerçekleştirmek ağacın dengeye gelmesini sağlamayacaktır. Bu yüzden II. durumda gerçekleştirilen döndürme işleminin simetriği gerçekleştirilir.



SRR
 ----->



SLR
 ----->

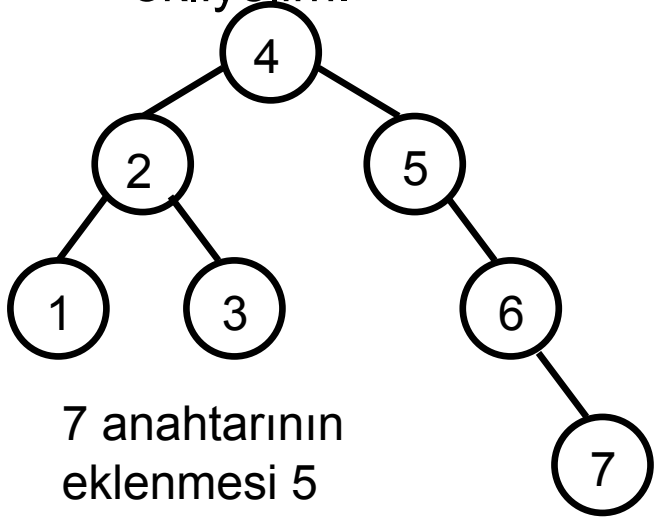


A düğümü sağa doğru dengesiz bir durumdadır. C düğümünün sol kesimi ise daha ağırdır. Ağacı dengeli AVL ağacına dönüştürmek için ard arda SRR ve SLR dönüşümleri yapılır.

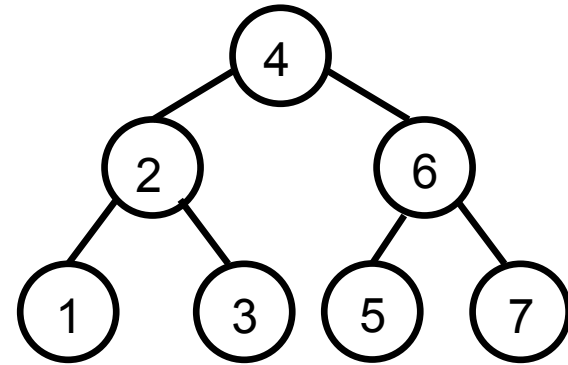
Örnek



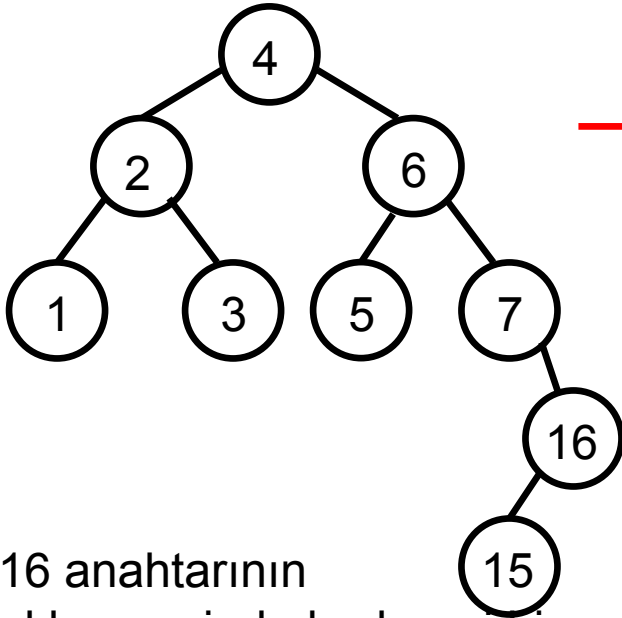
Bir önceki örnekte oluşturduğumuz ağaca yeni anahtarlar ekleyelim. Sırayla 7, 16, 15, 14, 13, 12, 11, 10, 8, 9 anahtarlarını ağacımıza ekliyelim.



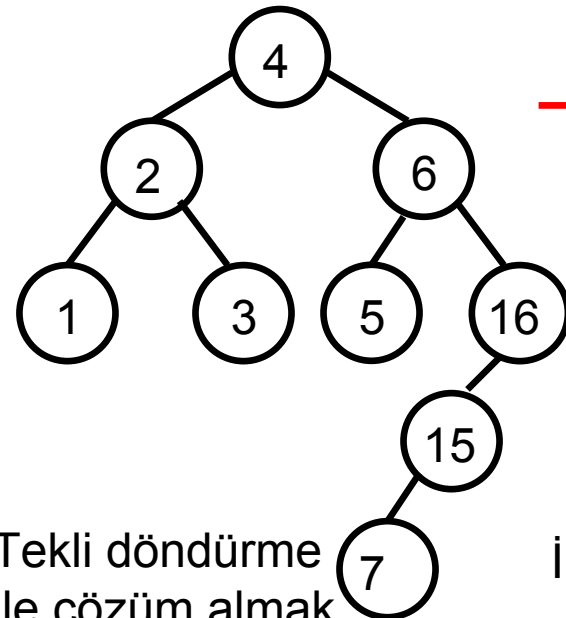
7 anahtarının eklenmesi 5 numaralı düğümde dengesizliğe neden olur.



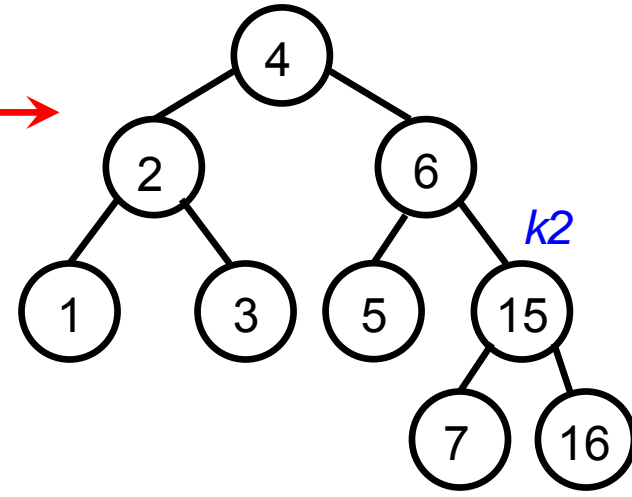
Single rotation



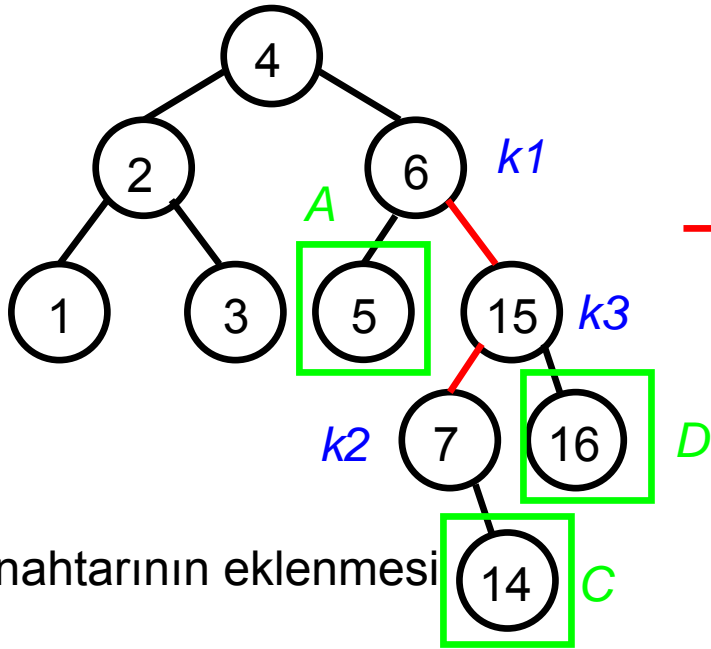
16 anahtarının eklenmesinde herhangi bir problem yok iken 15 anahtarının eklenmesiyle 7 numaralı düğümde dengesizlik meydana gelir.



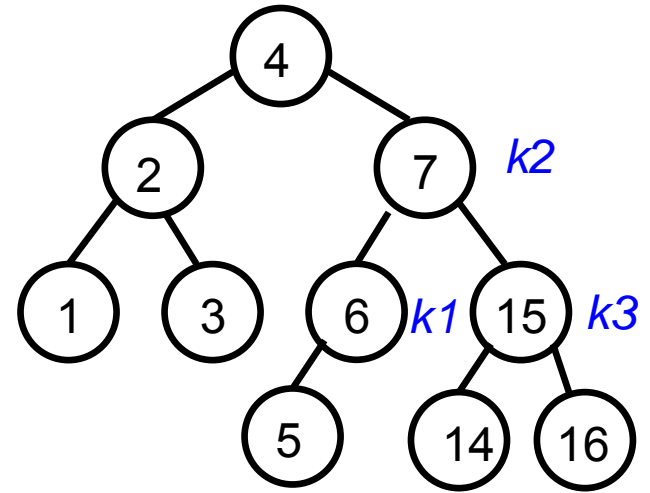
Tekli döndürme ile çözüm almak mümkün değildir. Hala dengesiz durum devam etmektedir.



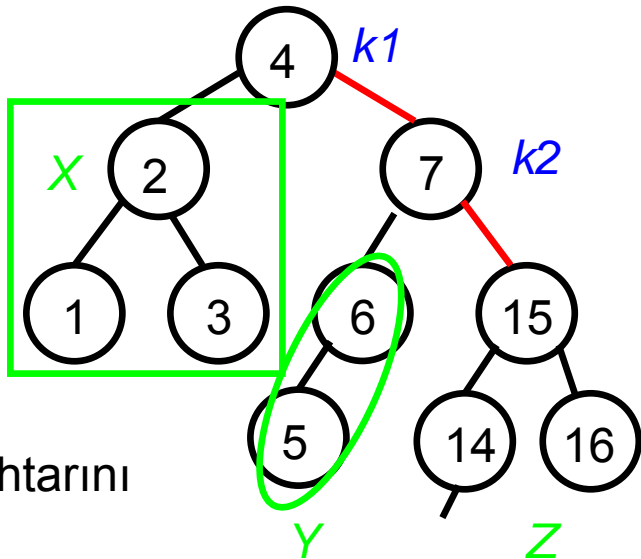
İkili Döndürme Yapılır ^{k1} ^{k3}



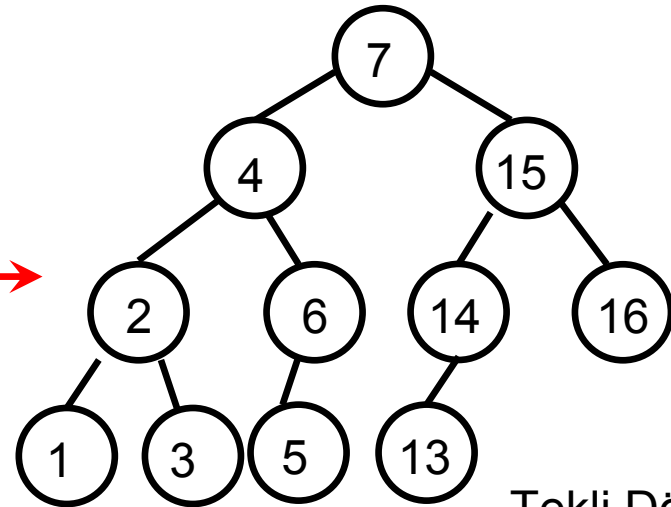
14 anahtarının eklenmesi



İkili Döndürme

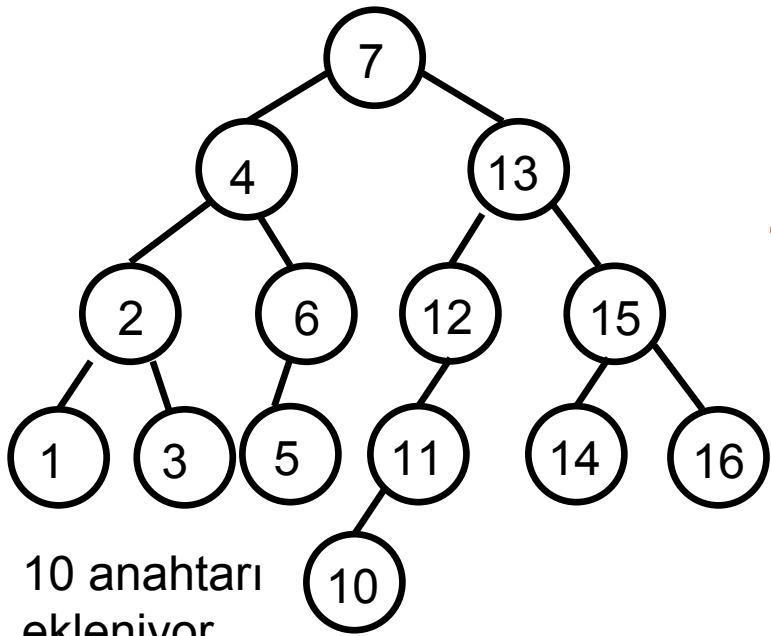


13 anahtarının eklenmesi

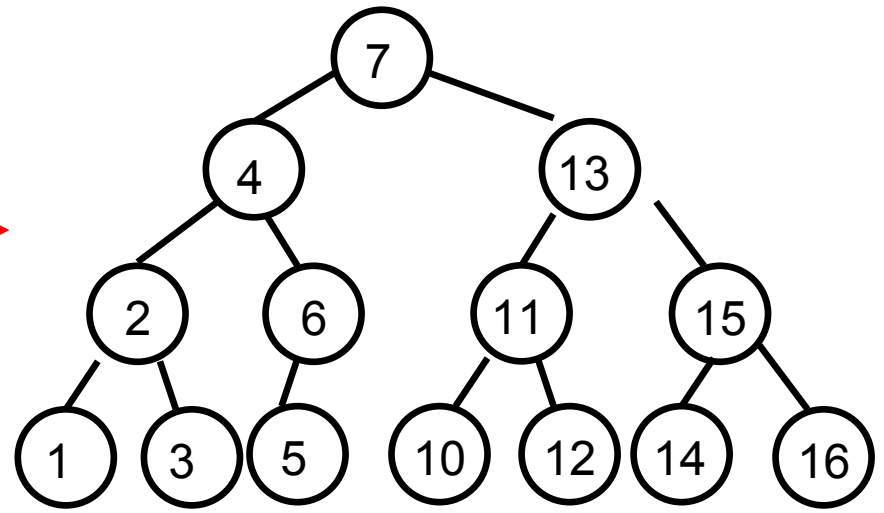


Tekli Döndürme

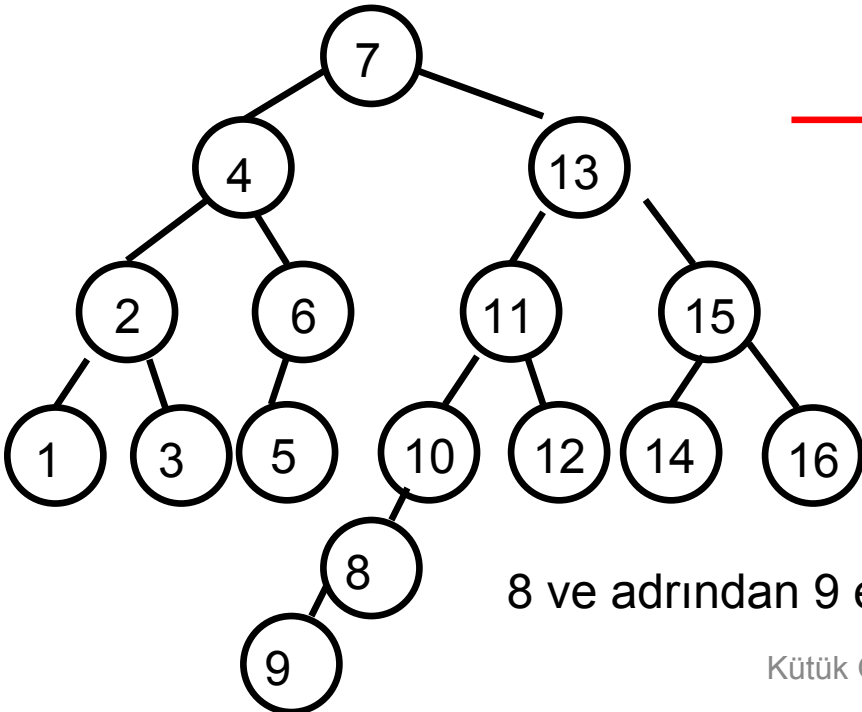
Kütük Organizasyonu



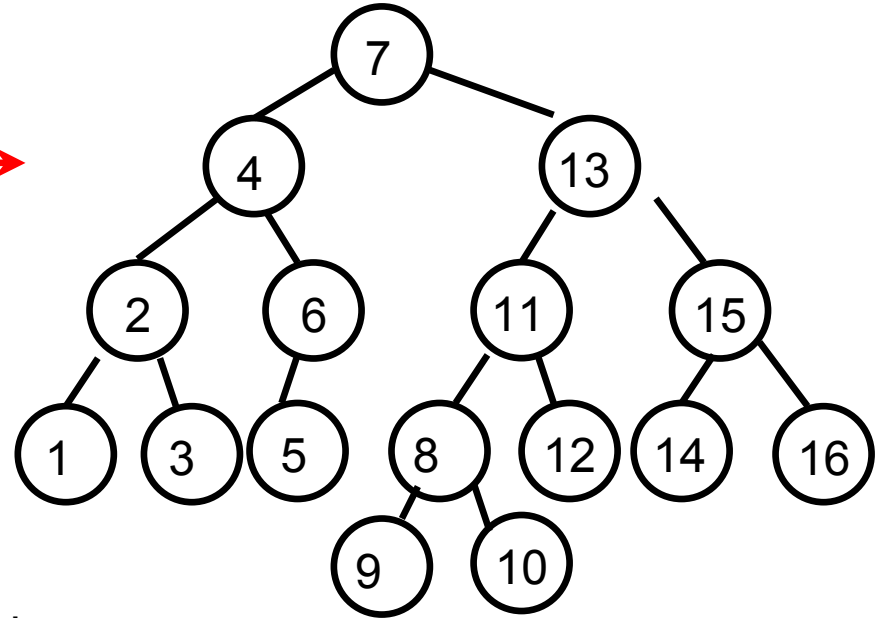
10 anahtarı ekleniyor



Tekli Döndürme



8 ve ardından 9 ekleniyor.



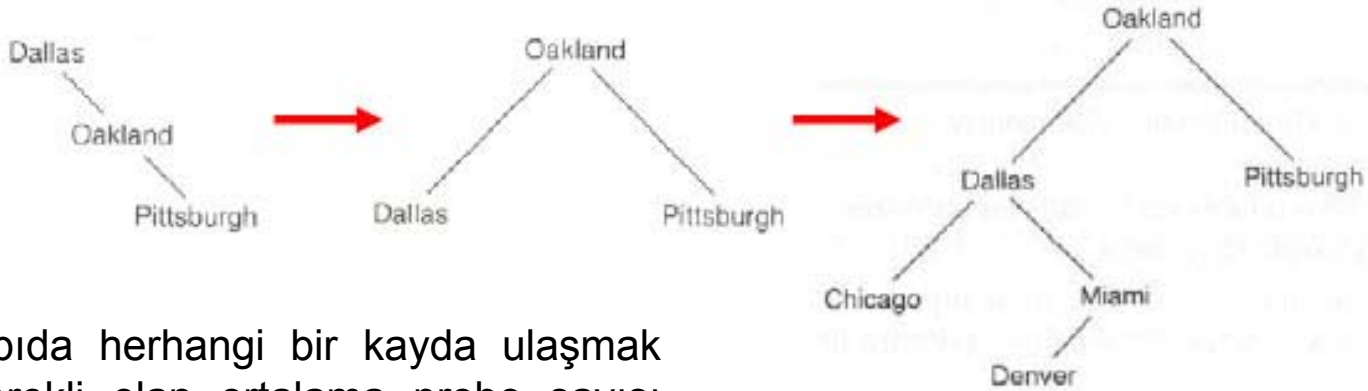
Tekli Döndürme

Dosya Yapılarında AVL Ağaçlarının Kullanılması

Örnek



Örnek: Dallas, Oakland, Pittsburg, Miami, Chicago, Denver, Boston ve Spivey's Corner kayıtlarını AVL ağacında saklamak istiyelim. Bu durumda ağaç oluşturulurken izlenecek adımlar sırasıyla aşağıdadır.



Bu yapıda herhangi bir kayda ulaşmak için gerekli olan ortalama probe sayısı **2,62**'dir.



Genel Olarak AVL Ağaçları

- ✓ Tüm kayıtların eklenmesi beklenmeden her eklemeden sonra bir veya iki döndürme yapılır.
- ✓ Dengesiz bir ağacın kök düğümünün denge faktörü $-/+ 2$ olur.
- ✓ Ekleme işlemi sırasında izlenen yol üzerinde $-/+1$ denge faktörü değerine sahip olan düğüm yoksa ağaç hala dengededir.
- ✓ Ekleme eklenen son yaprak kendi parent'ının ilk child'ı ise yol üzerindeki tüm düğümlerin denge faktör değeri yeniden düzenlenir.
- ✓ Eğer eklenen son yaprak kendi parent'ının ikinci child'ı ise sadece parent denge faktör değeri yeniden düzenlenir.