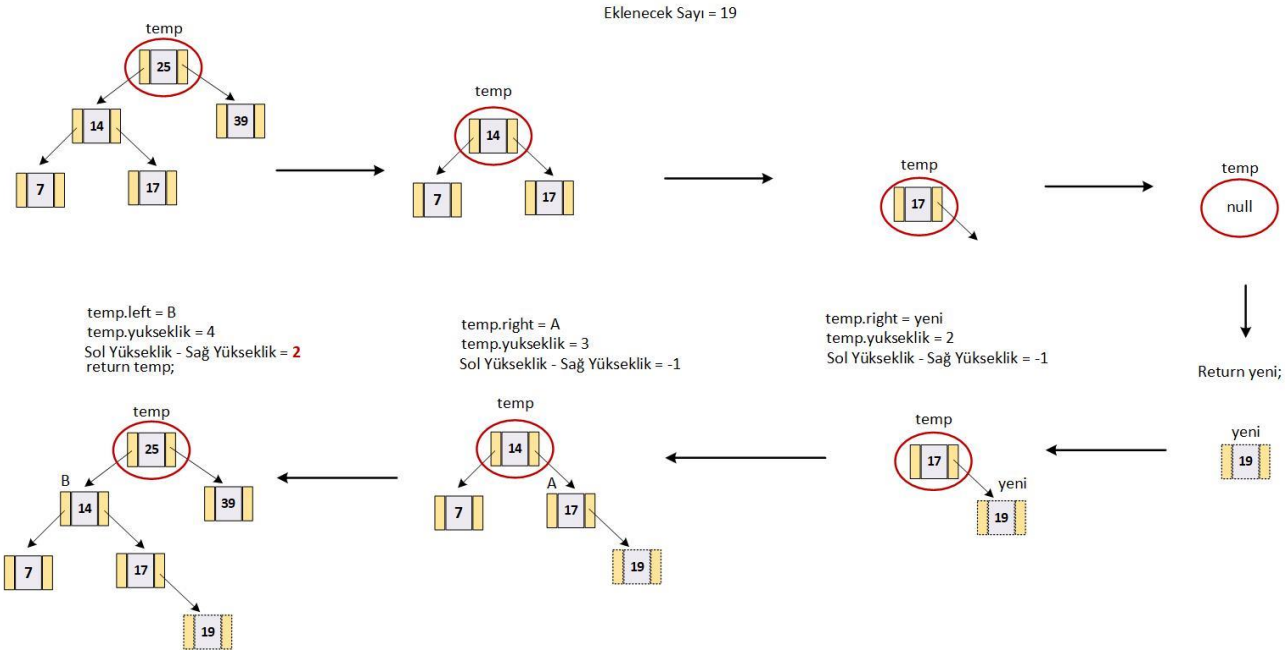
AVL AĞAÇLARI

AVL ağaçları normal ikili ağaçlar gibi olup tek farkları sol-sağ alt düğüm yükseklikleri farkı 1'den fazla olma durumlarında ağacı dengelemek için kırılım işleminin gerçekleştirilmesidir. AVL ağaçlarının oluşturulması için gereken Node yapısı aşağıda verilmiştir.

```
class Node {
    int deger;
    int yukseklik;
    Node sol;
    Node sag;

    Node(int deger){
        this.deger = deger;
    }
}
```

Bu Node yapısından yararlanarak bir düğüm ekleme işlemi şekil 1'de verilmiştir. Şekil 1'de var olan ağaca 19 elemanı eklenmek istenmektedir. Bu durumda rekürsif olarak yaprak düğüme kadar ilerleme işlemi gerçekleştirilir. Her düğüm için eklenecek değer, düğüm değerinden küçükse sola, büyükse sağa dallanma işlemi gerçekleştirilir. En son null değerine ulaştığında yeni eklenecek eleman return edilir. Rekürsif yapı sondan başa doğru gittiğinde yükseklik değerleri güncellenir. Bu sırada sol ve sağ çocuğun yükseklik farkları hesaplanır. Eğer 1'den büyük veya -1'den küçükse dengesizlik var demektir. Ona göre uygun kırılım işlemi gerçekleştirilir.

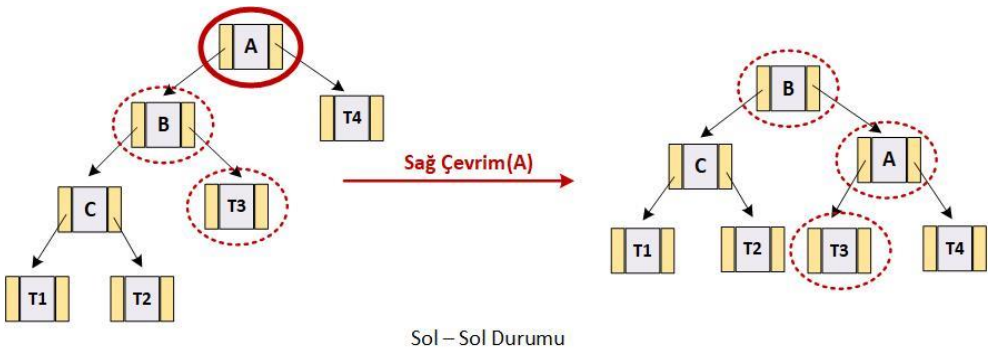


Şekil 1. AVL ağacı ekleme işlemi

AVL ağaçları için ekleme kodu aşağıda verilmiştir.

```
Node ekle(Node temp, Node yeni) {
    if(temp == null) {
        return yeni;
    }else{
        if(temp.deger > yeni.deger) {
            temp.sol = ekle(temp.sol, yeni);
        }else {
            temp.sag = ekle(temp.sag, yeni);
        }
    }

    temp.yukseklik = max(yukseklik(temp.sol), yukseklik(temp.sag)) + 1;

    int fark = yukseklik(temp.sol) - yukseklik(temp.sag);

    if(fark > 1 && yeni.deger < temp.sol.deger) {
        return sagaKir(temp);
    }

    if(fark > 1 && yeni.deger > temp.sol.deger) {
        temp.sol = solaKir(temp.sol);
        return sagaKir(temp);
    }

    if(fark < -1 && yeni.deger > temp.sag.deger) {
        return solaKir(temp);
    }

    if(fark < -1 && yeni.deger < temp.sol.deger) {
        temp.sol = sagaKir(temp.sag);
        return solaKir(temp);
    }

    return temp;
}
```

Dengesizlik durumlarında 4 farklı durum meydana gelir. Bunlar sol-sol, sağ-sağ, sol-sağ ve sağ-sol'dur. Sol-sol durumu şekil 2'de verilmiştir.

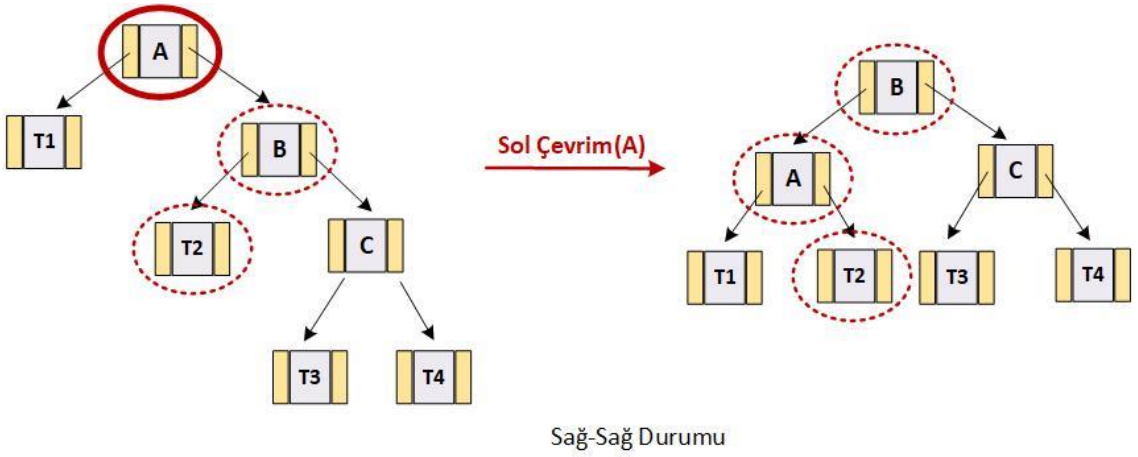


Şekil 2. Sol-sol durumu ve sağ kırılımın gerçekleştirilmesi

Şekil 2'deki bu işlem için gereken kod parçası aşağıda verilmiştir.

```
Node sagaKir(Node A) {
    Node B = A.sol;
    Node T3 = B.sag;

    A.sol = T3;
    B.sag = A;

    A.yukseklik = max(yukseklik(A.sol),yukseklik(A.sag)) + 1;
    B.yukseklik = max(yukseklik(B.sol),yukseklik(B.sag)) + 1;

    return B;
}
```

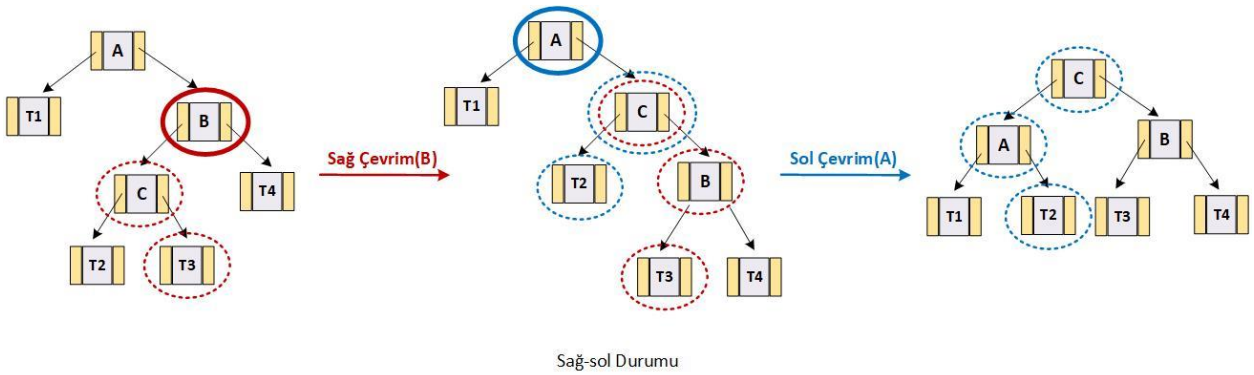
Sağ-sağ durumunda ise sola kırılım gerçekleştirilmelidir. Bu durum şekil 3'te gösterilmiş ve sola kırılım için gereken kod parçası hemen aşağısında verilmiştir. Burada devamlı bordo halka ile kırılımın uygulandığı düğüm gösterilmiştir. Kesikli halkalar ise bağlantısı değişen düğümleri göstermektedir.



Şekil 3. Sağ-sağ durumu ve sol kırılımın gerçekleştirilmesi

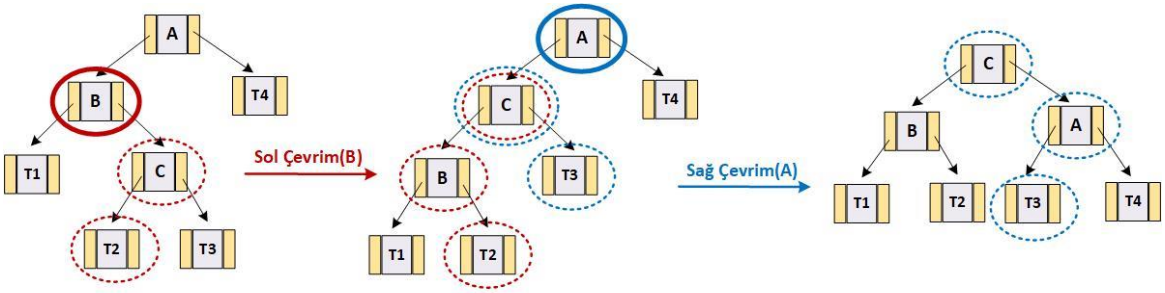
```
Node solaKir(Node A) {
    Node B = A.sag;
    Node T2 = B.sol;

    B.sol = A;
    A.sag = T2;

    A.yukseklik = max(yukseklik(A.sol),yukseklik(A.sag)) + 1;
    B.yukseklik = max(yukseklik(B.sol),yukseklik(B.sag)) + 1;

    return B;
}
```

Sağ-sol ve sol-sağ kırılımlarının gerçekleştirilmesi şekil 4 ve şekil 5 ile verilmiştir. İlk adımdaki kırılımda baz alınan düğüm kesintisiz bordo halka ile gösterilmiştir. Bordo halkalar ilk çevrimden etkilenen düğümleri ifade etmektedir. İkinci kırılımın baz alındığı düğüm ise mavi kesintisiz halka ile gösterilmiştir. Benzer olarak mavi halkalar ile 2. kırılımdan etkilenen düğümler gösterilmektedir. Şekil 4'teki işlemlerin gerçekleştirilmesi için sagaKir(B) ardından da solaKir(A) fonksiyonu çağrılmalıdır. Şekil 5 için de benzer durum söz konusudur.



Sağ-sol Durumu

Şekil 4. Sağ-sol durumu ve sırası ile sağ - sol kırılımının gerçekleştirilmesi



Sol – Sağ Durumu

Şekil 5. Sol-sağ durumu ve sırası ile sol - sağ kırılımının gerçekleştirilmesi