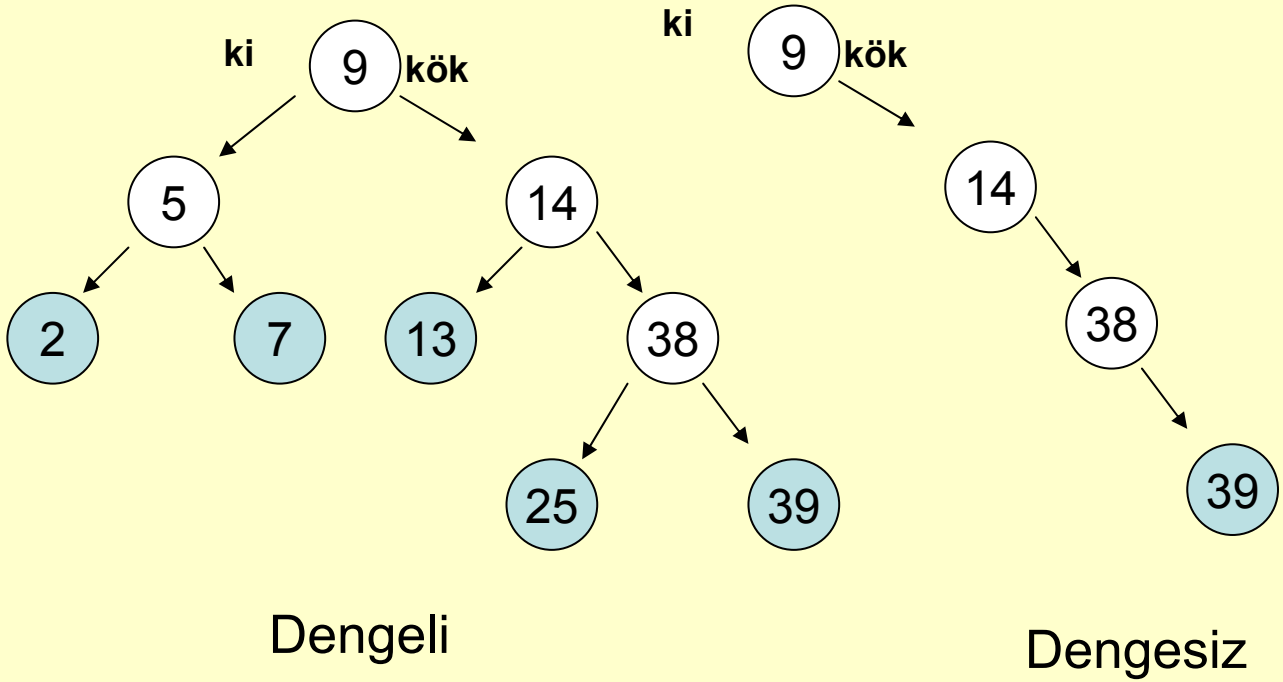
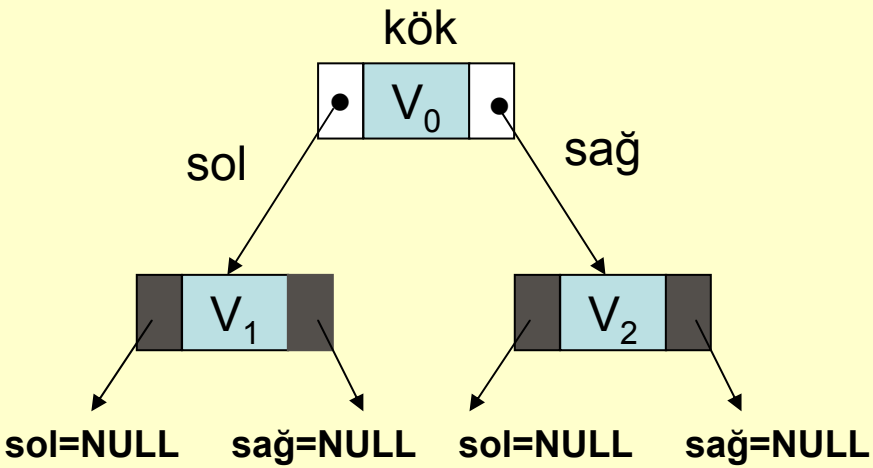
Veri Yapıları

İkili Ağaç;



Veri Yapıları

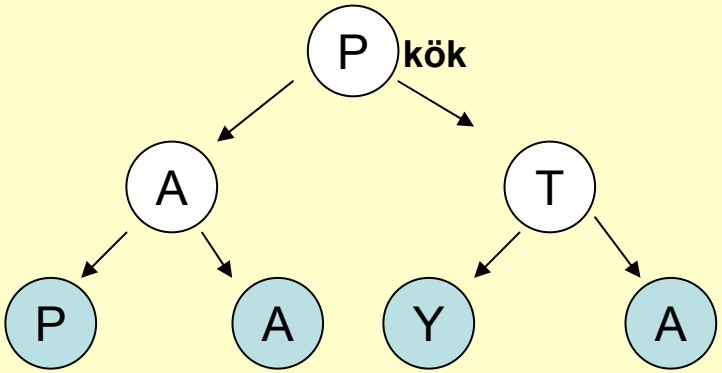
İkili Ağaç veri yapısı;



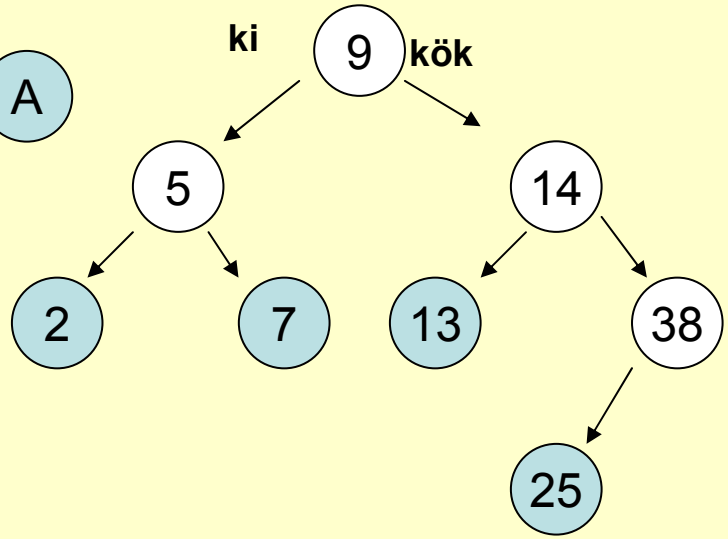

```
struct DUGUM2{

        int veri;
        struct DUGUM2 *sol;
        struct DUGUM2 *sag;
};
```

Veri Yapıları

İkili Ağaç üzerinde dolaşma;



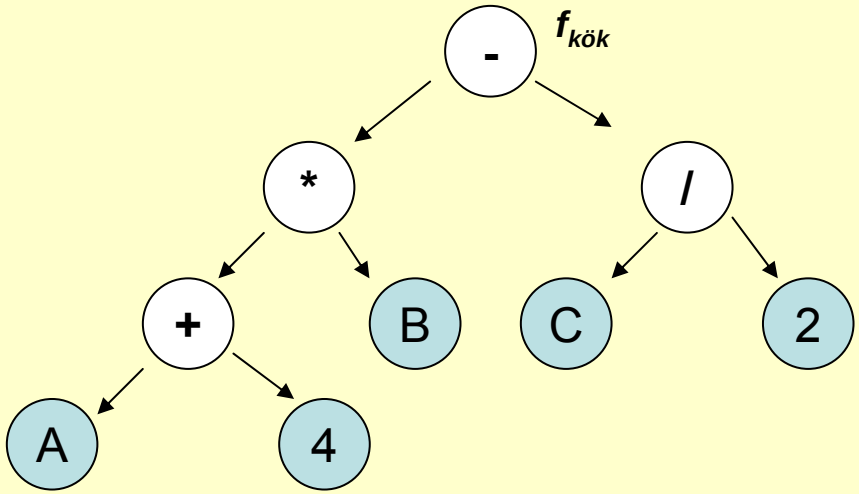
•Önce-kök : PAPATYA
•Ortada-kök : PAAPYTA
•Sonra-kök : PAAYATP



•Önce-kök : 9 5 2 7 14 13 38 25
•Ortada-kök : 2 5 7 9 13 14 25 38
•Sonra-kök : 2 7 5 13 25 38 14 9

Veri Yapıları

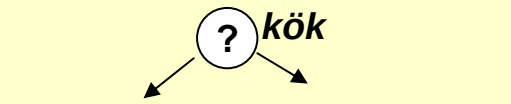
Bağıntı Ağaç yapısı;



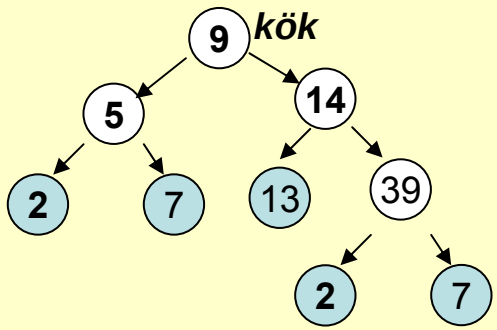
$f(x) = (A + 4) * B - C / 2$

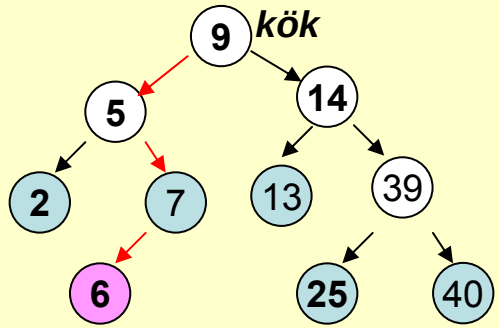
Veri Yapıları

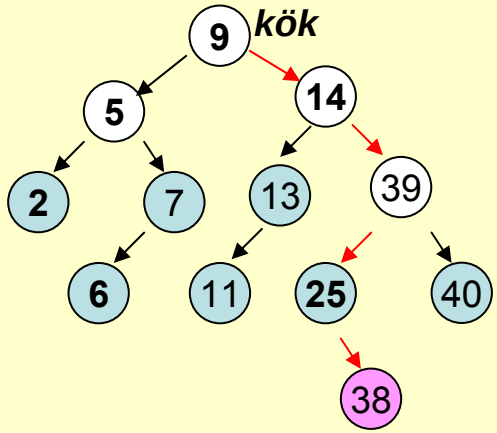
İkili Ağaça düğüm ekleme;

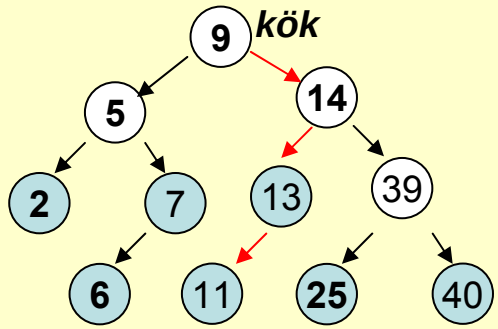



```
if(kök boş ağacı gösteriyorsa)
  düğümü köke ekle;
else {
  if(eklenen kökten küçükse)
    sol alt ağaca dallan;
    basa dön;
  else
    sağ altağaca dallan;
    basa dön;
}
```









Veri Yapıları

İkili Ağaçta dolaşma;

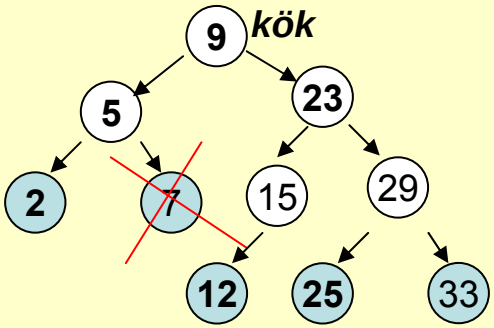
```
if (kök boş değilse; düğüm varsa) {
  düğüm üzerindeki bilgiyi kullan;
  fonksiyonu kökün sol altağacı için rekürsif olarak tekrar çağır.
  fonksiyonu kökün sağ altağacı için rekürsif olarak tekrar çağır.
}
```

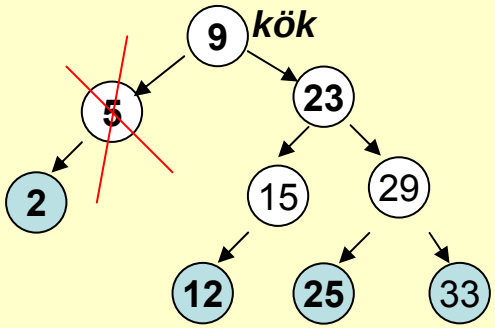
İkili Ağaçta arama;

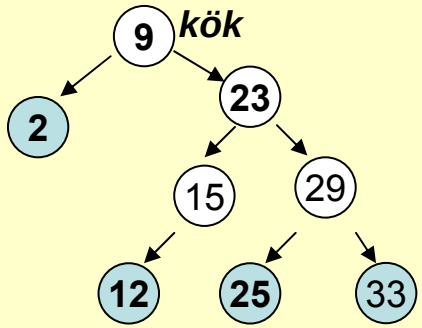
```
while (kök boş değilse ve aranan kökte değilse) {
    if(aranan kökteki bilgiden küçükse)
       kök=kökün sol çocuğu adresi;
    else
       kök=kökün sağ çocuğu adresi;
}
 return kök;
}
```

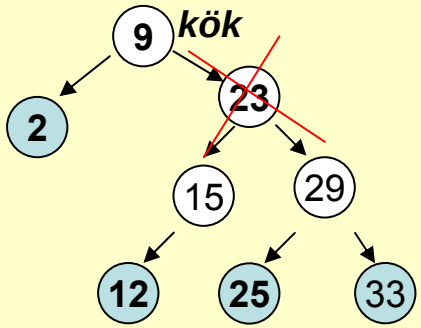
Veri Yapıları

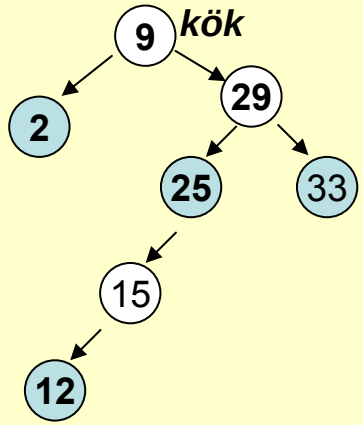
İkili Ağaçta silme;



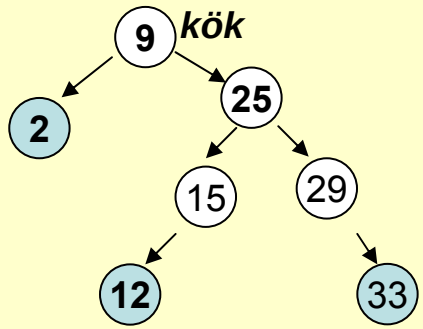








Dengesiz durum



Dengeli durum

Veri Yapıları

İkili Ağaçta silme;

```
while (silinecek düğümün ve ailesinin adresini bulana kadar) {
    q<- silinecek düğümün adresi, qa<-ailesinin adresi;
    if(silinmek istenen bulunamadı ise)
       yapacak bir şey yok dön;
    if(silinecek düğümün iki alt çocuğuda varsa)
       sol altağacın en büyük değerli düğümünü bul;
       bu düğümdeki bilgiyi silinmek istenen düğüme aktar;
    bu aşamada en fazla bir çocuğu olan düğümü sil;
    silinen düğümün işgal ettiği bellek alanını serbest bırak;
}
```