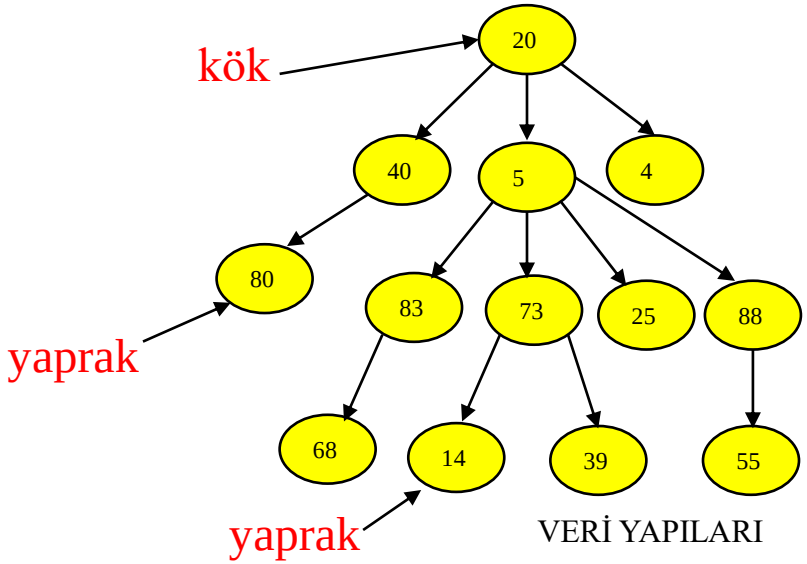
AĞAÇLAR

- Ağaçlar düğümlerden oluşur
- Her düğümün çocukları olabilir
- Bir düğümün çocuk sayısı ağacın cinsine göre değişir
  - ✓ Genel ağaçlarda bir düğümün n (n sıfır veya pozitif tam sayı) tane çocuğu olabilir.
  - ✓ İkili ağaçlarda bir düğümün 0, 1 veya 2 tane çocuğu olabilir
- Hiç çocuğu olmayan düğümlere yaprak düğüm denir
- Her çocuğun bir ebeveyni (parent) vardır.
- Birbirine bağlantılı düğümler arasında ebeveyn-çocuk (parent-child) ilişkisi vardır
- Ebeveyni olmayan düğüme kök (root düğüm) denir
- Yaprak olmayan düğümlere (root da dahil) iç düğüm denir

Örnek Ağaçlar

- Onlu ağaçlar
  - ✓ Bir düğümün en fazla on tane çocuğu var
- Dörtlü ağaçlar
  - ✓ Bir düğümüm en fazla 4 çocuğu var
- İkili ağaçlar
  - ✓ Bir düğümün en fazla 2 çocuğu var

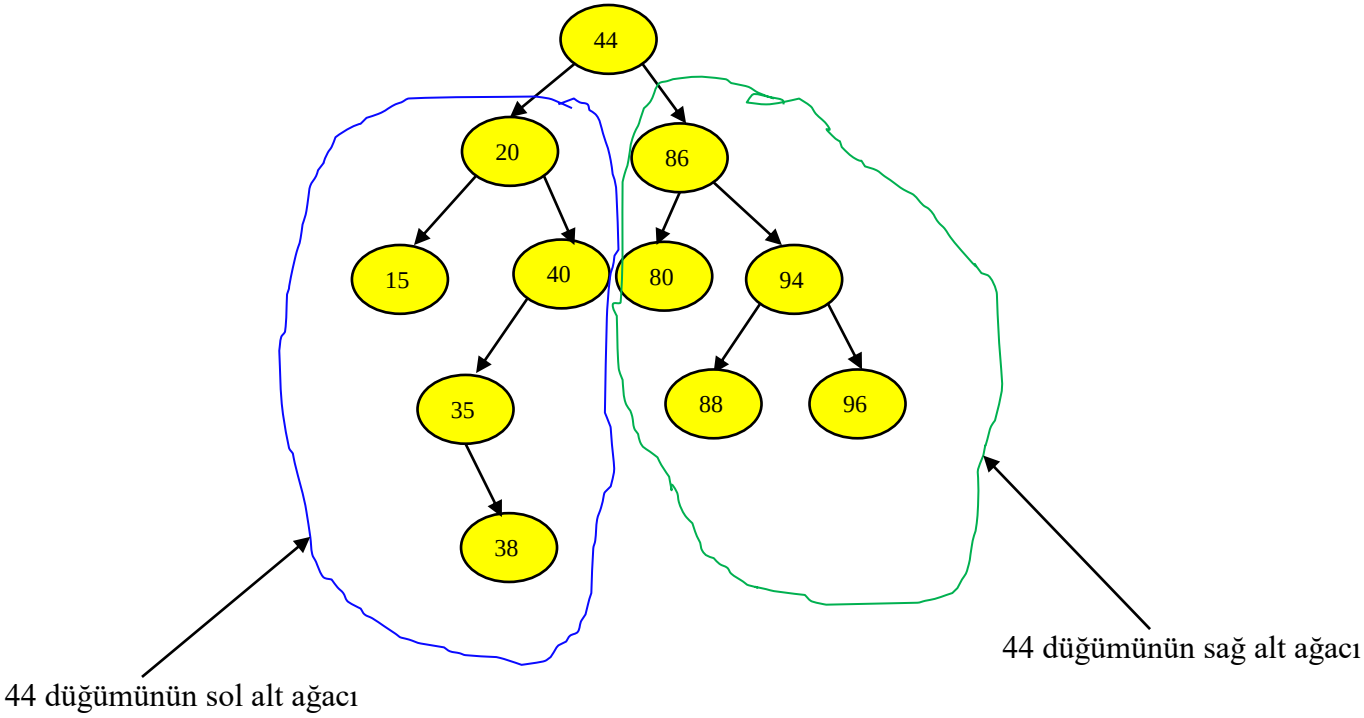


• Dörtlü ağaç
•id'si 20 olan düğüm kök düğümdür
  • Ebeveyni yok
• id'leri 80,68,14,39,20, 55 ve 4 olan düğümler yaprak düğümlerdir
  • Çocukları yok
• id'si 5 olan düğümün 4 tane çocuğu var (83, 73, 25 ve 88)
• Yaprak düğümler haricindeki düğümler (20, 40, 5, 83, 73 ve 88) iç düğümlerdir
• 4 düğümünün ebeveyni 20, 83'ün ebeveyni 5 dir; 20 (root) nin ise parenti yoktur

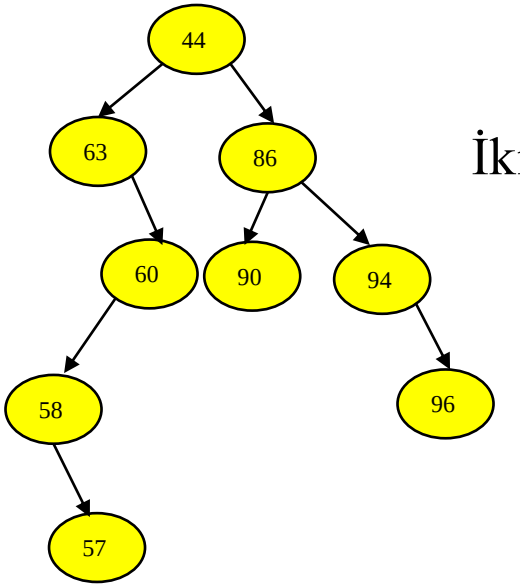
İkili Arama Ağacı

- Biz bu derste ikili arama ağacına odaklanacağız
- İkili ağacın özel bir halidir
- Hızlı aramalarda yaygın olarak kullanılır
  - ✓ Bir düğümün en fazla 2 çocuğu olabilir
  - ✓ Herhangi bir düğümün id'si o düğümün sol alt ağacındaki bütün düğümlerin id'sinden büyük
  - ✓ Herhangi bir düğümün id'si o düğümün sağ alt ağacındaki bütün düğümlerin id'sinden büyüktür

İkili Arama Ağacı



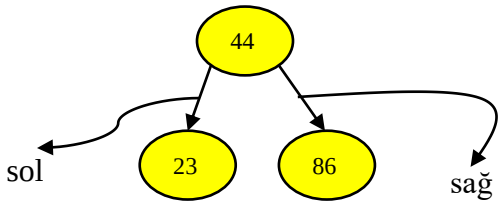
İkili Arama Ağacı



İkili ağaç, ancak ikili arama ağacı değil

İki Arama Ağacı

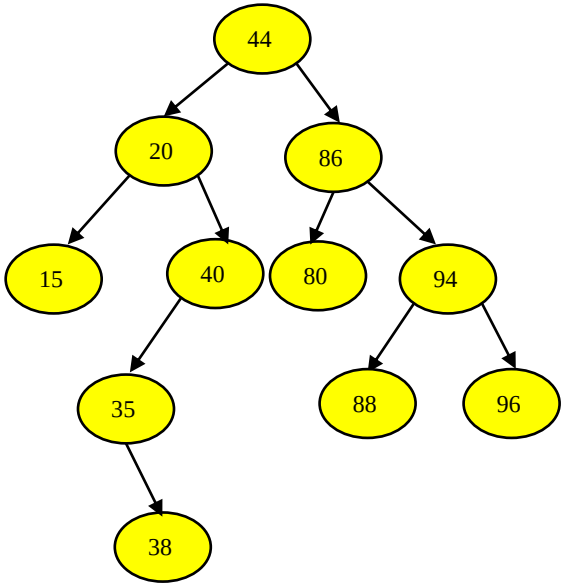
```
struct iaa_dugum {
    int icerik;
    struct iaa_dugum *sol;
    struct iaa_dugum *sag;
};
```

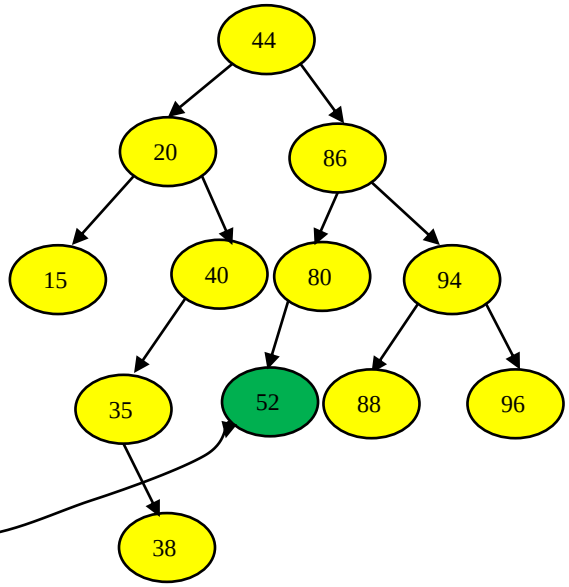


Düğüm Ekleme

- Ağacın kökünden başlayarak, eklenecek düğümün id'si (anahtarı) ağacın düğümün id'si ile kıyaslanır.
  - ✓ Eğer ağacın düğümünün id'si eklenecek düğümün id'sinden büyükse düğümün soluna git
  - ✓ Eğer ağacın düğümünün id'si eklenecek düğümün id'sinden küçükse düğümün sağına git
- Bu işleme devam edilerek eklenecek olan düğüm ağaca daima yeni bir yaprak olarak eklenir.

Düğüm Ekleme





id'si 52 olan düğüm, ağaca yaprak düğüm olarak eklendi

Düğüm Ekleme

➢ Düğüm Oluşturma

```
struct dugum* dugum_olustur(int icerik){
   struct dugum *d = (struct dugum*)malloc(sizeof(struct dugum));
   if(d==NULL){
      printf("Heapte gerekli yer ayrilamadi... exit ...\n");
      exit(1);
   }
   d->icerik = icerik; //(*d).icerik=icerik;
   d->sol=d->sag=NULL;
   return d;
}
```

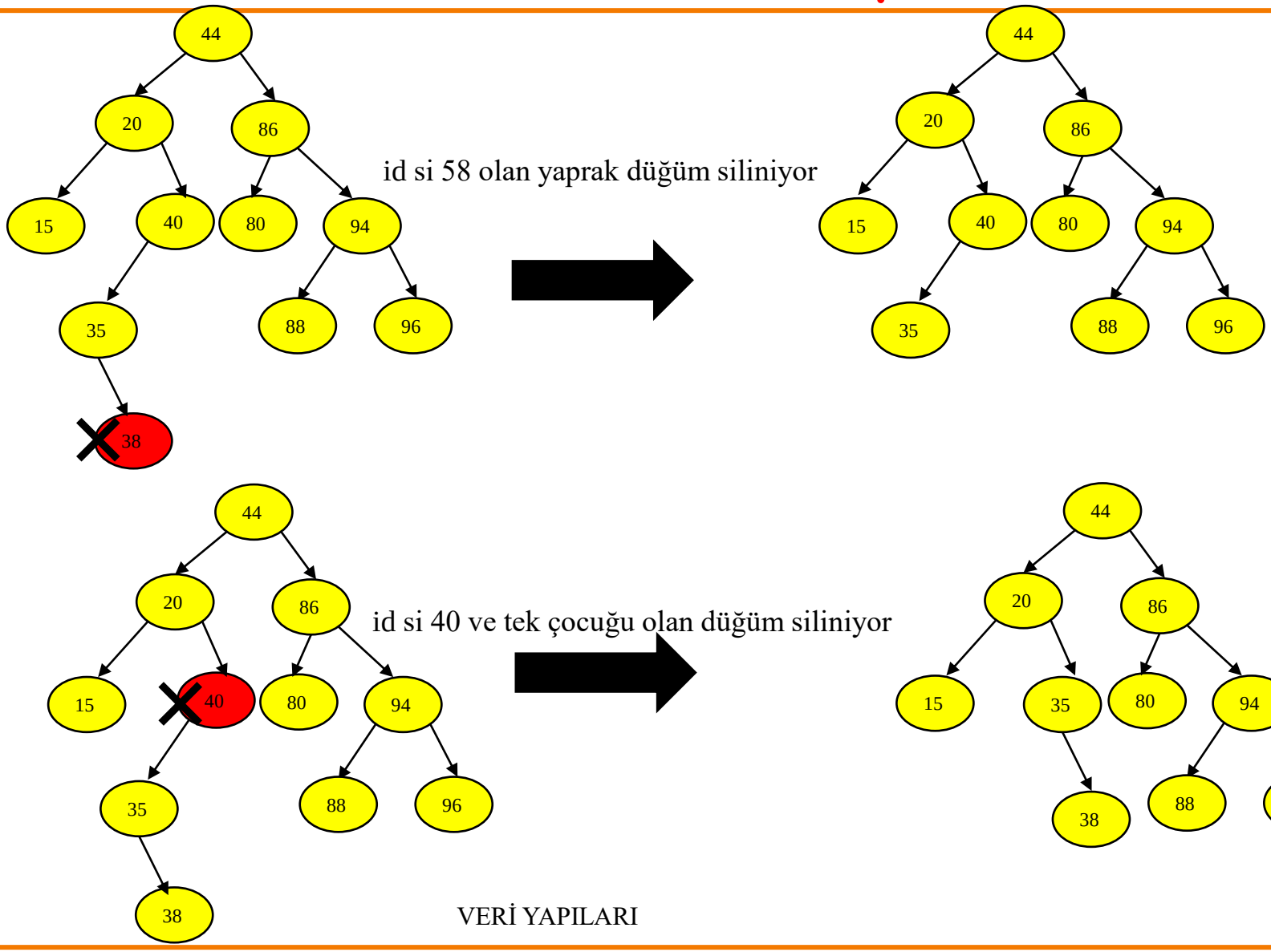
➢ Düğüm Ekleme

```
void ekle(struct ikili_arama_agaci *agac,int icerik){
   struct dugum *dugum;
   struct dugum *d;
   struct dugum *geri;

   d=agac->kok;
   while(d!=NULL){
      geri=d;
      if(icerik < d->icerik) d=d->sol;
      else if(icerik > d->icerik) d= d->sag;
      else return;
   }
   dugum=dugum_olustur(icerik);
   if(agac->kok==NULL){
      agac->kok = dugum;
      return;
   }
   if(icerik < geri->icerik) geri->sol = dugum;
   else geri->sag = dugum;

}
```

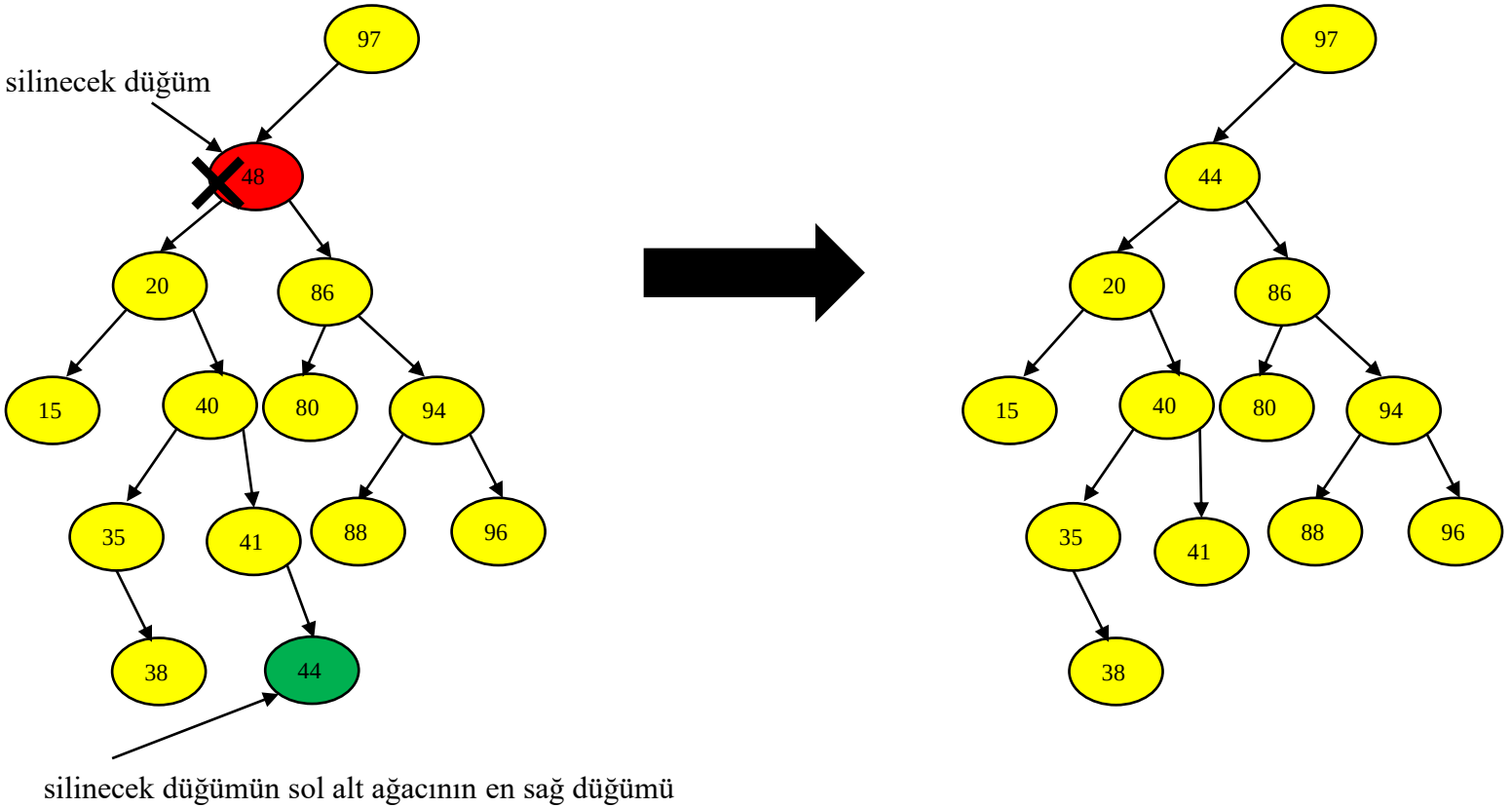
Düğüm Silme

- Ağacın kökünden başlamak suretiyle ağaç dolaşılarak silinecek düğüm tespit edilir
- Eğer silinecek düğüm yaprak veya sadece tek bir çocuğu varsa direkt silinir
- Aksi halde, silinecek düğüm (iç düğüm) belirlendikten sonra iki eylemden biri gerçekleştirilir
  - ✓ Ya silinen düğümüm sol alt ağacının en sağ düğümü belirlenip silinen düğümün yerini alması sağlanır
  - ✓ Ya da silinen düğümün sağ alt ağacının en sol düğümü belirlenip silinen düğümün yerini alması sağlanır

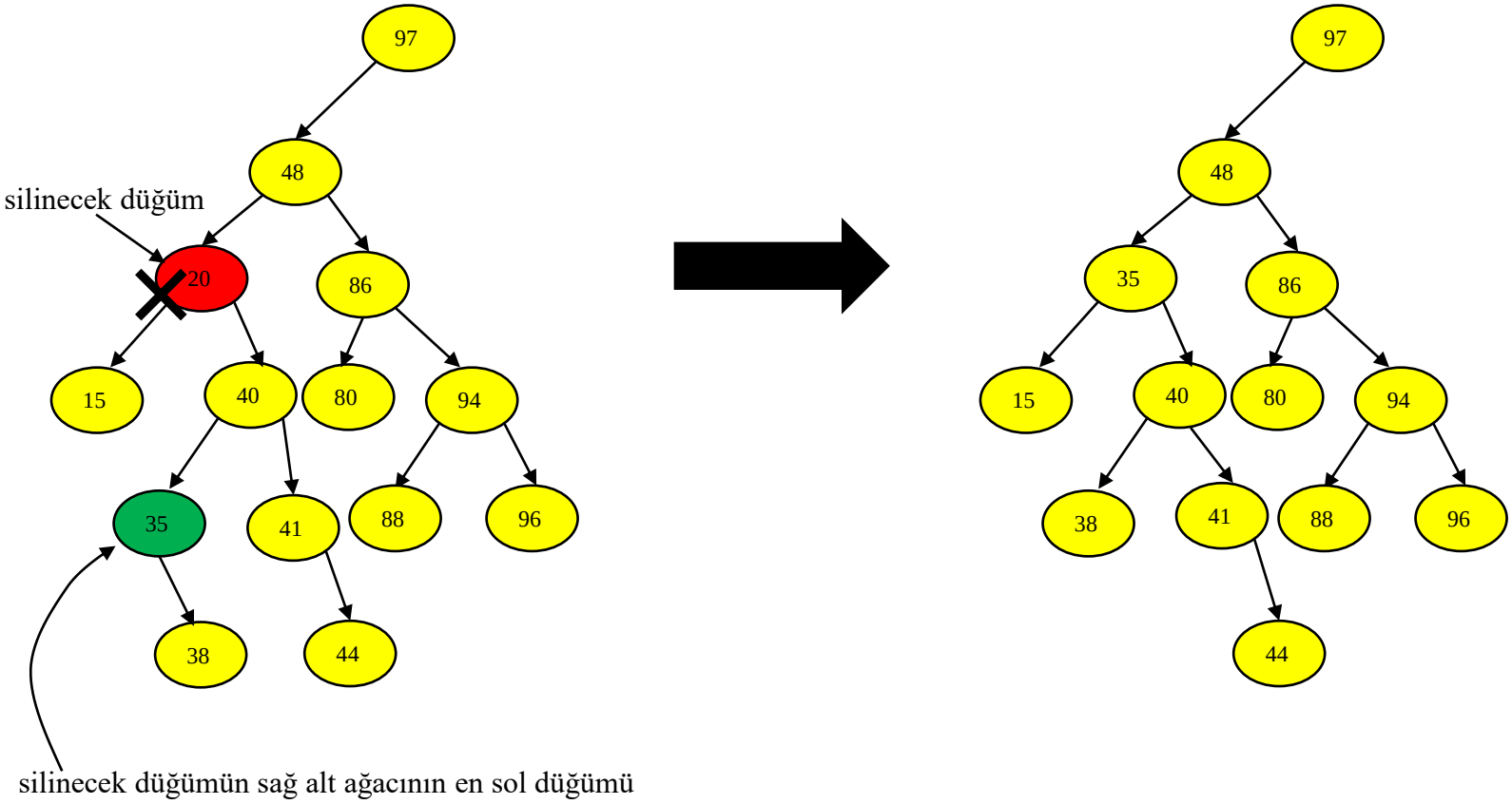
Silinen Düğümün Yaprak veya Tek Çocuğu Olan Düğüm Olması Durumu (Kolay)



İç Düğüm Silme (Silinen düğümün sol alt ağacının en sağ düğümü belirlenerek silme işlemi)



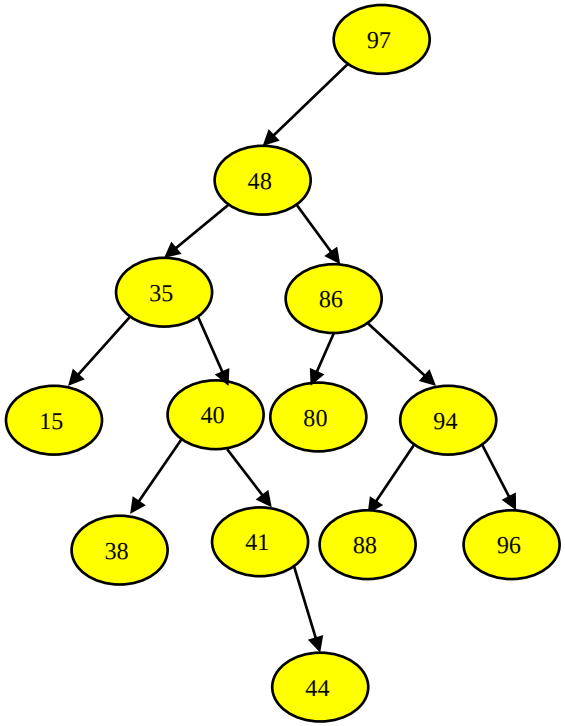
İç Düğüm Silme (Silinen düğümün sağ alt ağacının en sol düğümü belirlenerek silme işlemi)



İkili Arama Ağacı Dolaşımları

- pre-Order Dolaşım
  - ✓ Önce düğümün kendisi, daha sonra düğümün sol alt ağacı, en sonunda düğümün sağ alt ağacı dolaşılır
- in-Order Dolaşım
  - ✓ Önce düğümün kendisi, daha sonra düğümün sol alt ağacı, en sonunda ise düğümün sağ alt ağacı dolaşılır
- post-Order Dolaşım
  - ✓ Önce düğümün sol alt ağacı, sonra düğümün sağ alt ağacı, en sonunda ise düğümün kendisi dolaşılır
- Dolaşımları yenilemeli (recursive) fonksiyon çağrıları ile gerçekleştirmek iteratif yaklaşımından çok daha kolay

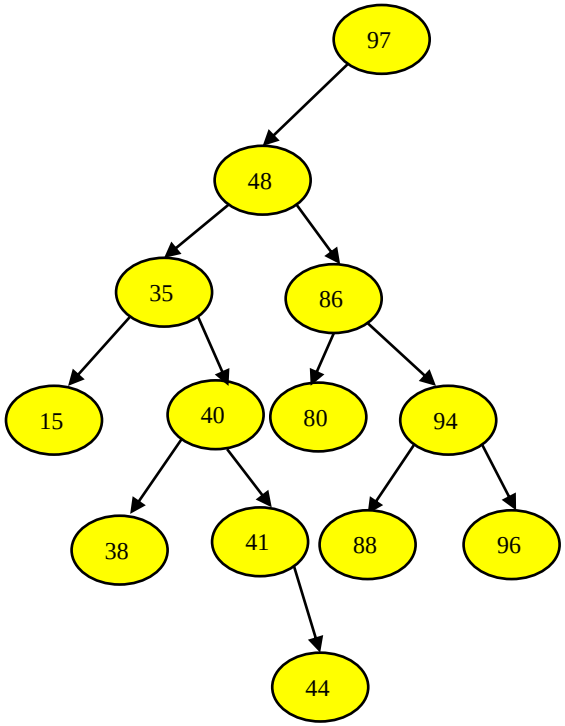
pre-Order Dolaşım




```
preOrder(struct dugum *kok){
    if(kok==NULL) return;
    printf("%4d ",kok->icerik);
    preOrder(kok->sol);
    preOrder(kok->sag);
}
```

97, 48, 35, 15, 40, 38, 41, 44, 86, 80, 94, 88, 96

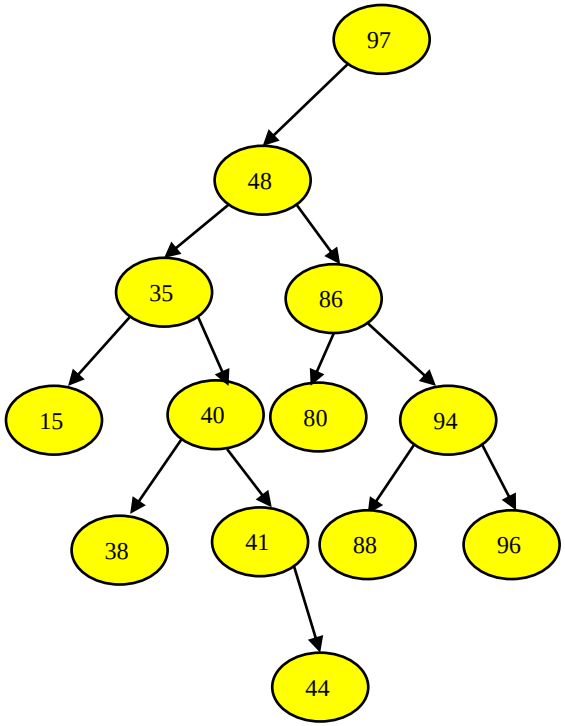
in-Order Dolaşım




```
inOrder(struct dugum *kok){
    if(kok==NULL) return;
    inOrder(kok->sol);
    printf("%4d ",kok->icerik);
    inOrder(kok->sag);
}
```

15, 35, 38, 40, 41, 44, 48, 80, 6, 88, 94, 96,97

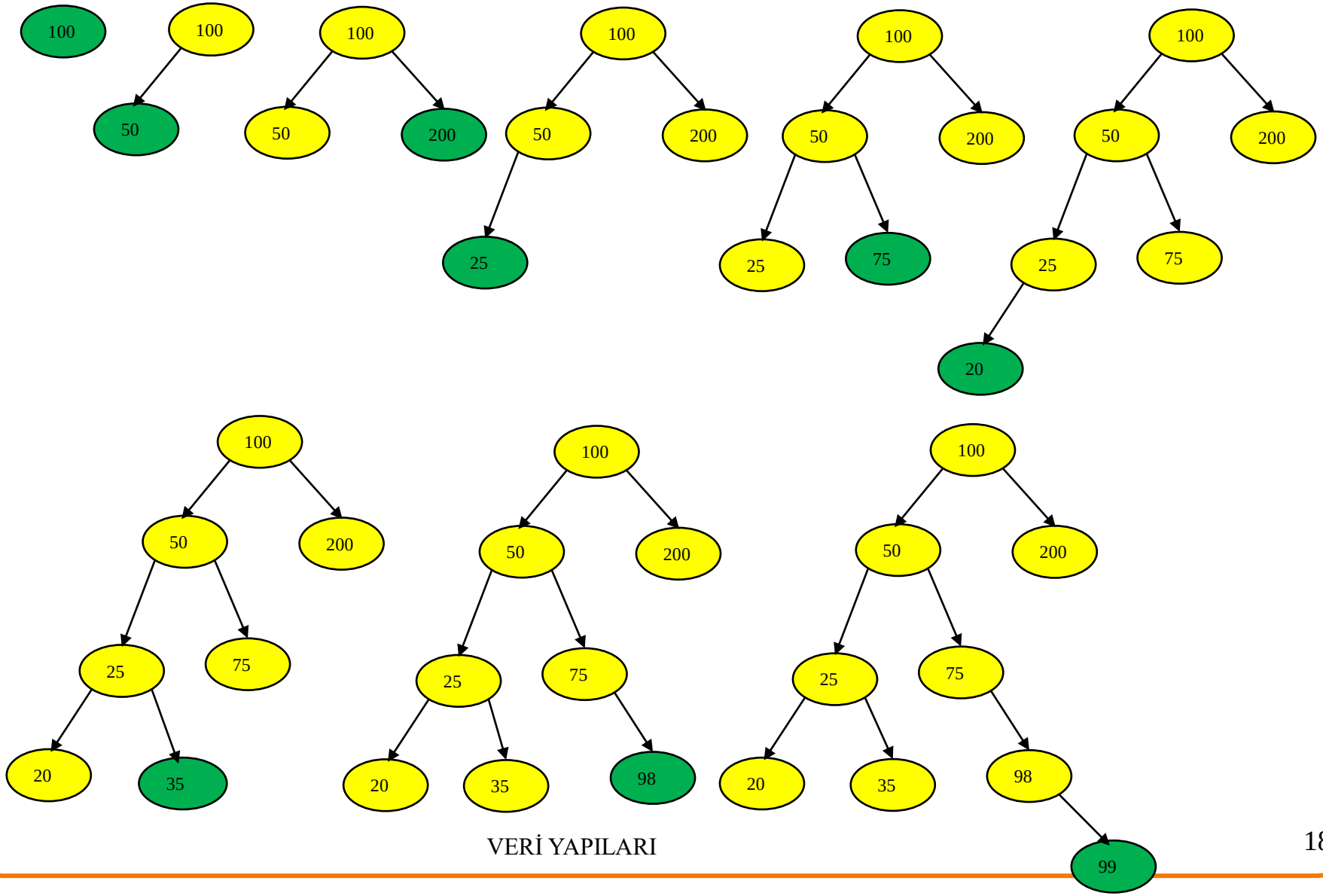
post-Order Dolaşım



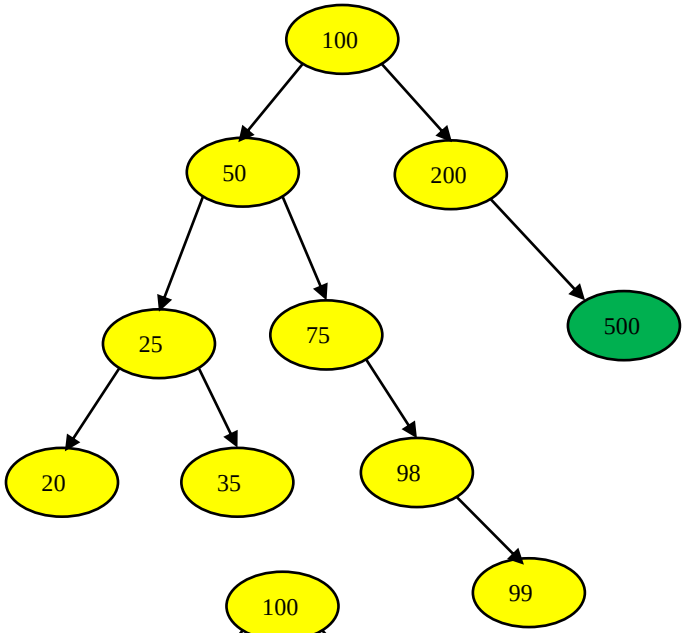

```
postOrder(struct dugum *kok){
    if(kok==NULL) return;
    postOrder(kok->sol);
    postOrder(kok->sag);
    printf("%4d ",kok->icerik);
}
```

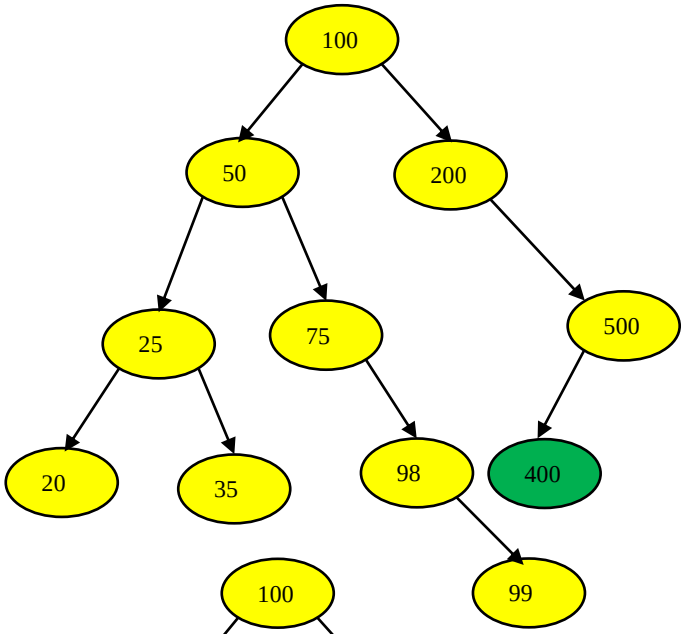
15, 38, 44, 41, 40, 35, 80, 88, 96, 94, 86, 48, 97

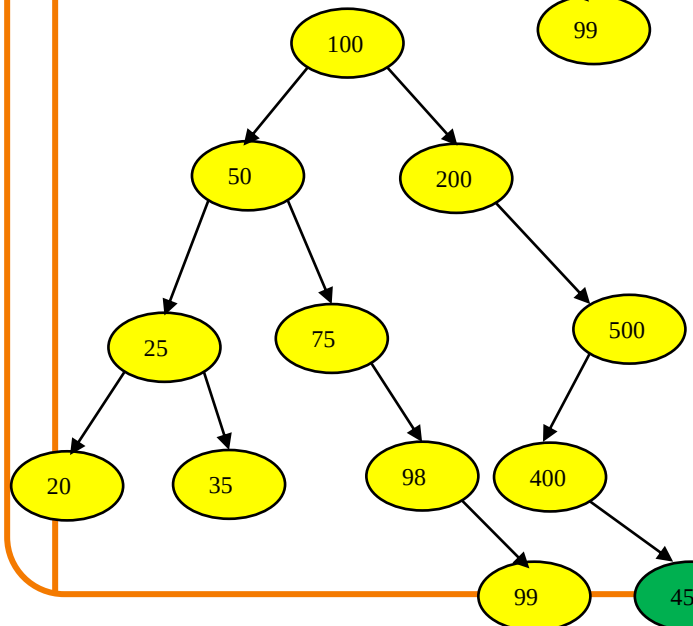
Örnek (Düğüm Ekleme)

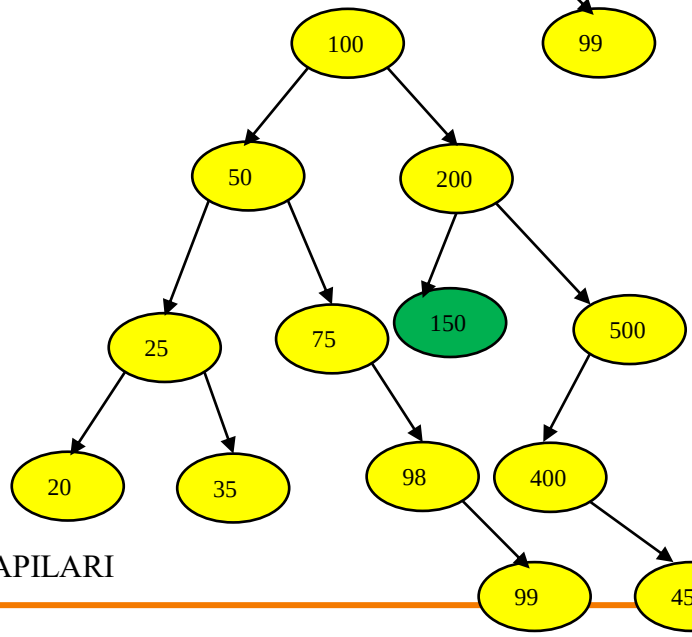


Örnek devam

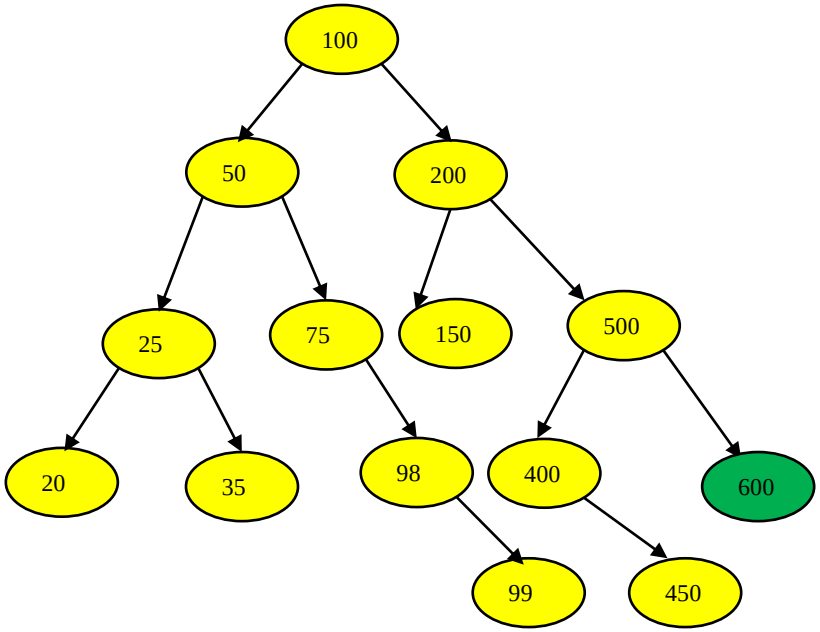

100
50
200
25
75
500
20
35
98
99


100
50
200
25
75
500
20
35
98
400
99

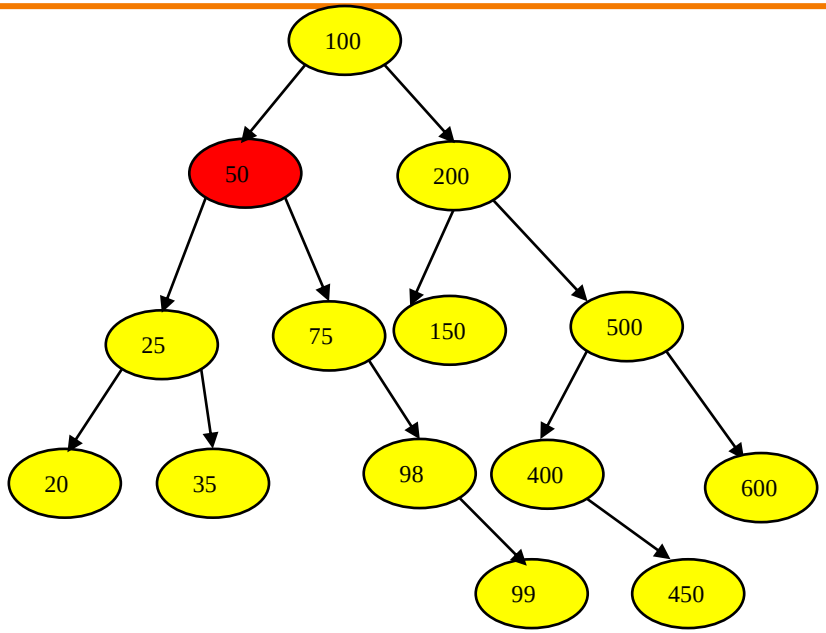

100
50
200
25
75
500
20
35
98
400
99
450

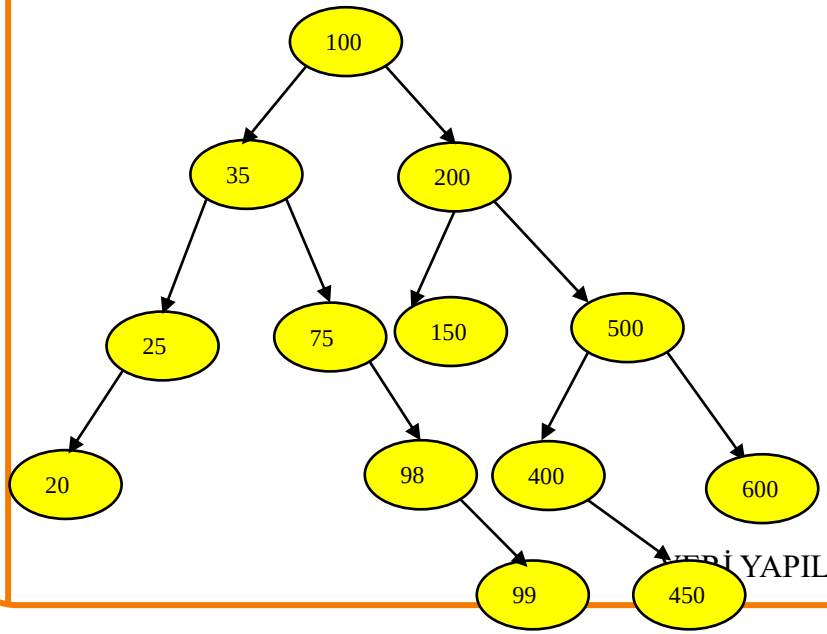

100
50
200
25
75
150
500
20
35
98
400
99
450

Örnek devam

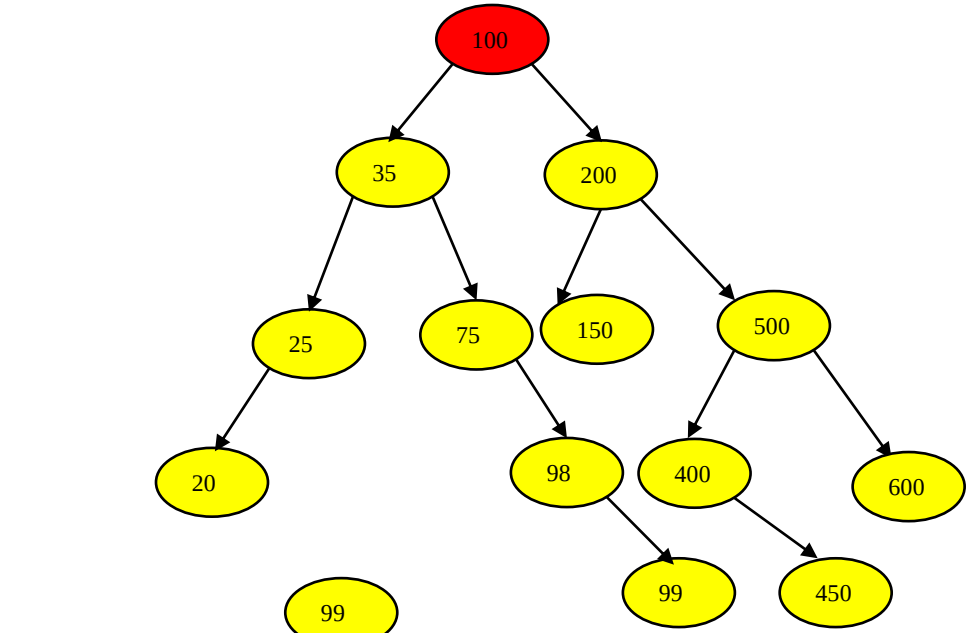


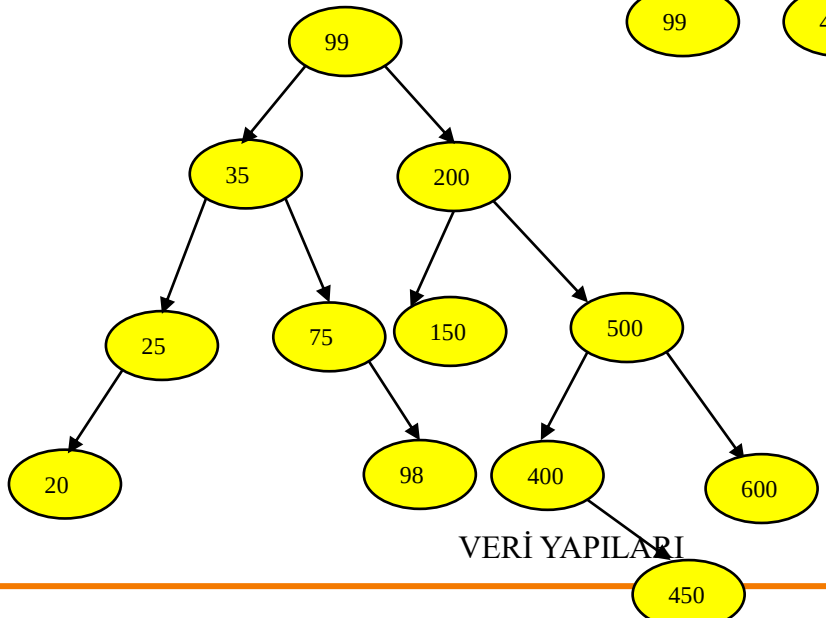
Örnek (Düğüm Silme)



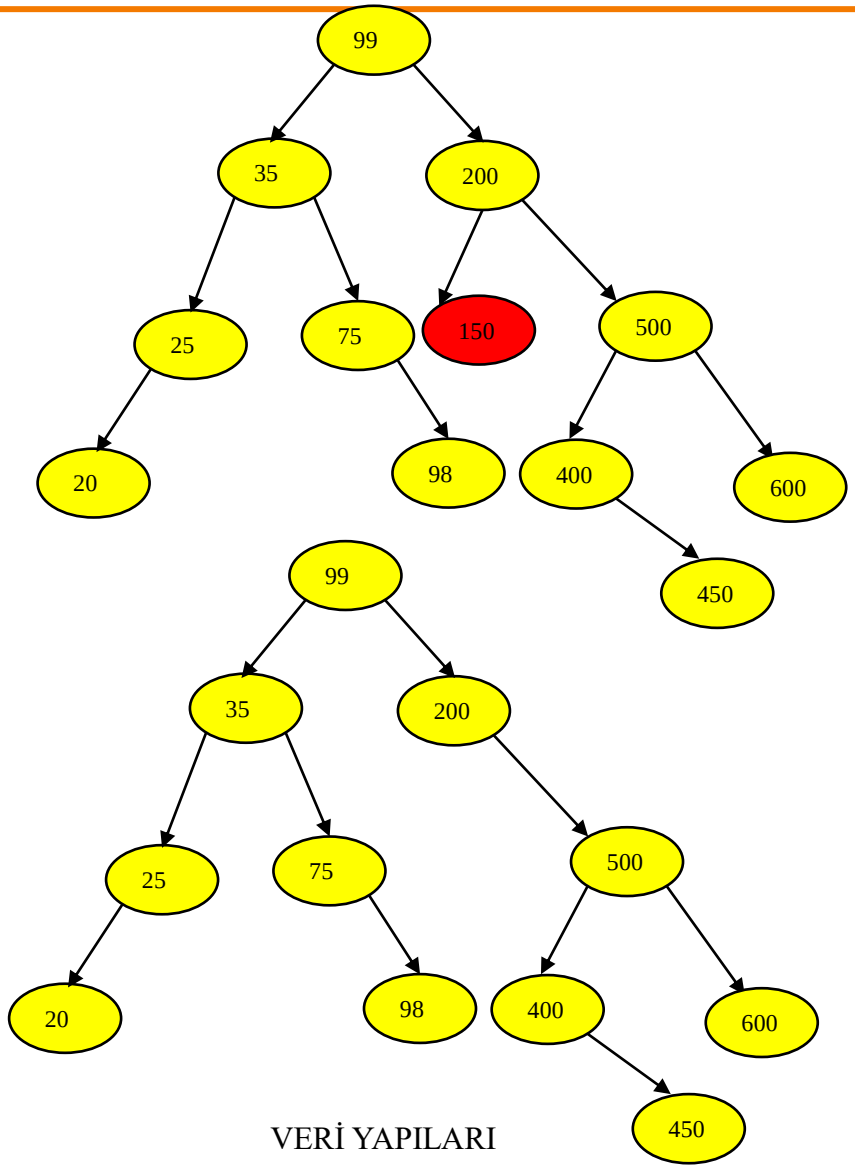


Örnek devam

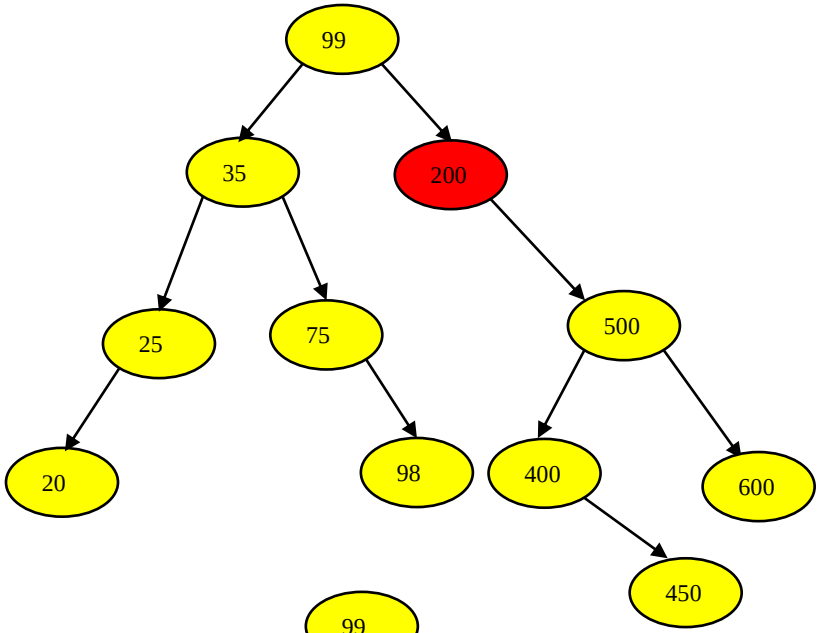

100
35
200
25
75
150
500
20
98
400
600
99
450

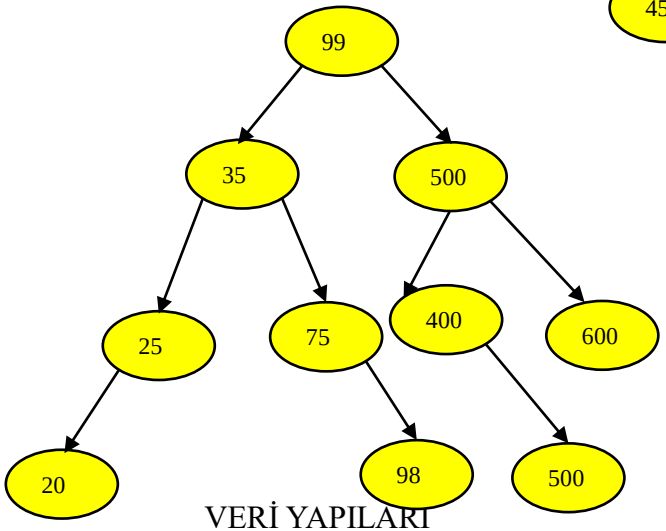

99
35
200
25
75
150
500
20
98
400
600
450

Örnek devam


99
35
200
25
75
150
500
20
98
400
600
450
99
35
200
25
75
500
20
98
400
600
450

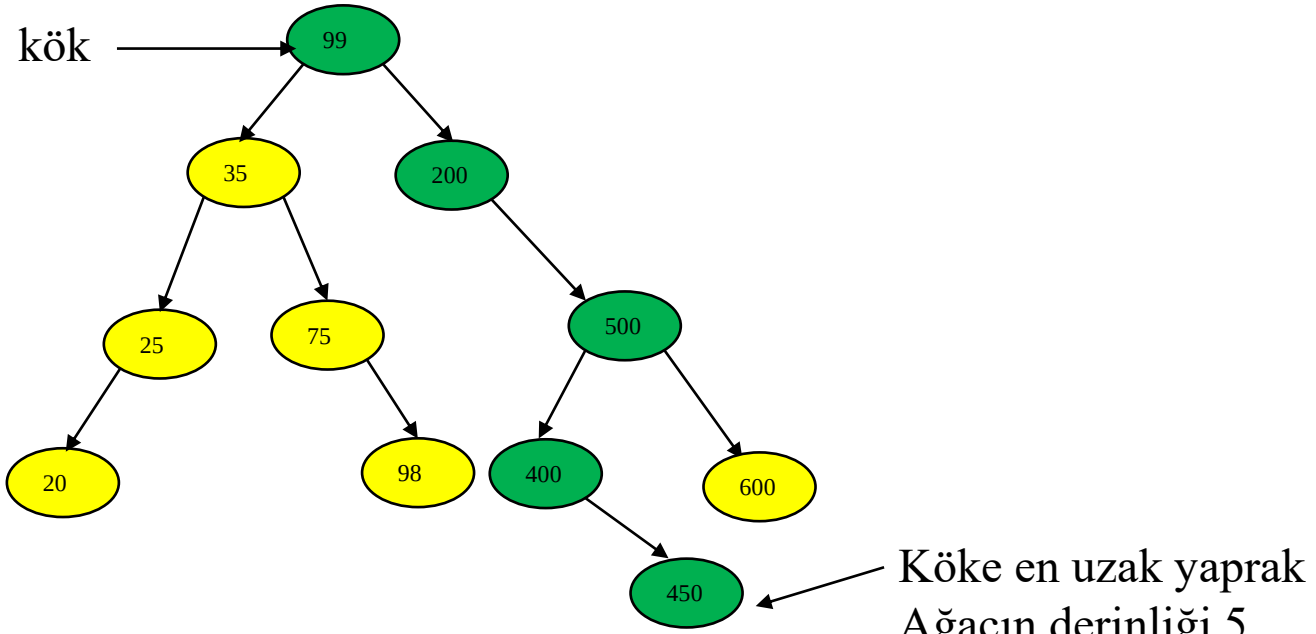
Örnek devam


99
35
200
25
75
500
20
98
400
600
450


99
35
500
25
75
400
600
20
98
500

İkili (İkili Arama) Ağacının Yüksekliği (Derinliği)

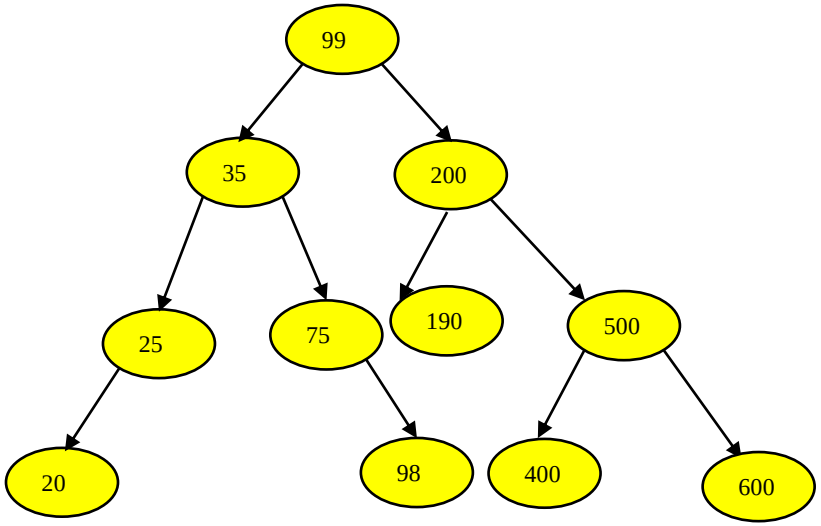
- Ağacın köküne en uzaktaki yaprağa olan yol üzerindeki düğümlerin sayısına ağacın yüksekliği (derinliği) denir

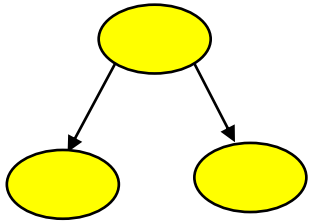


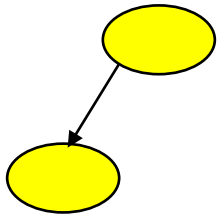
Dengeli İkili Ağaç

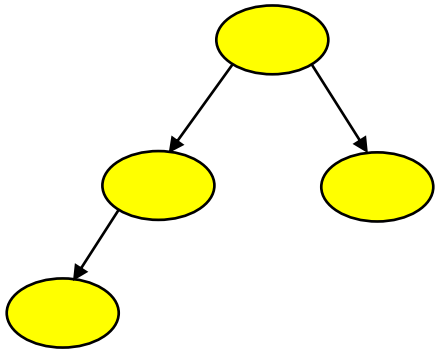
- Ağaçtaki her bir düğüme ait sol alt ağaç ile sağ alt ağaç arasındaki yükseklik farkı en fazla 1 ise bu ağaca dengeli ağaç denir
- Ağaç dengeli ise en kötü durumlardaki arama, ekleme ve silme işlemleri hızlı gerçekleştirilir

Dengeli İkili Ağaç Örnekleri

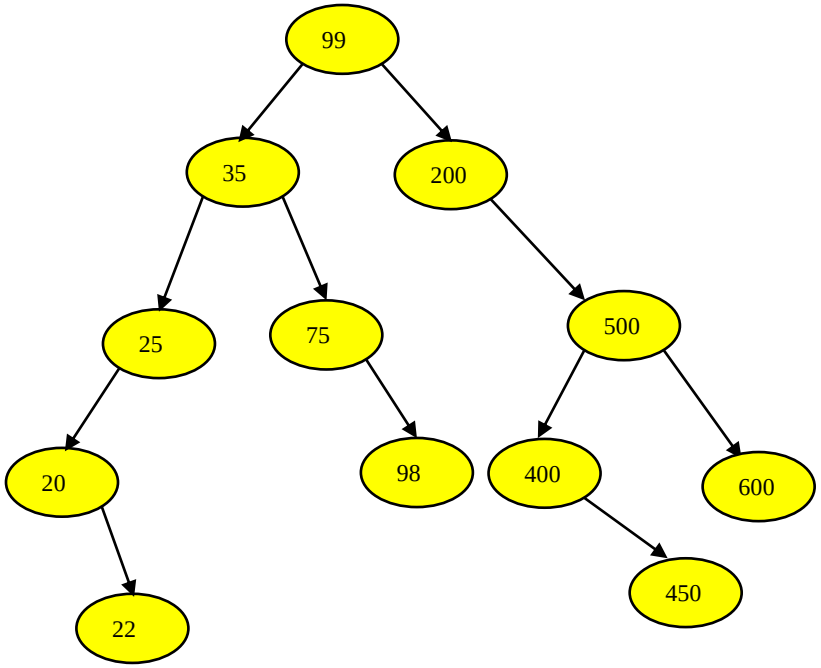


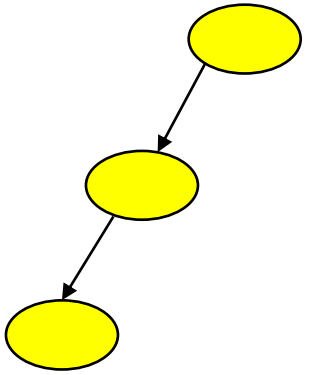


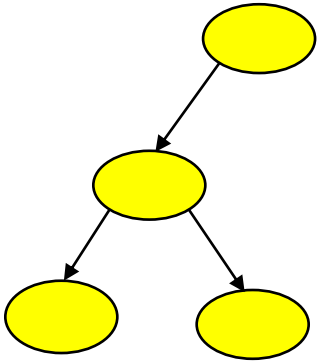




Dengeli Olmayan İkili Ağaç Örnekleri

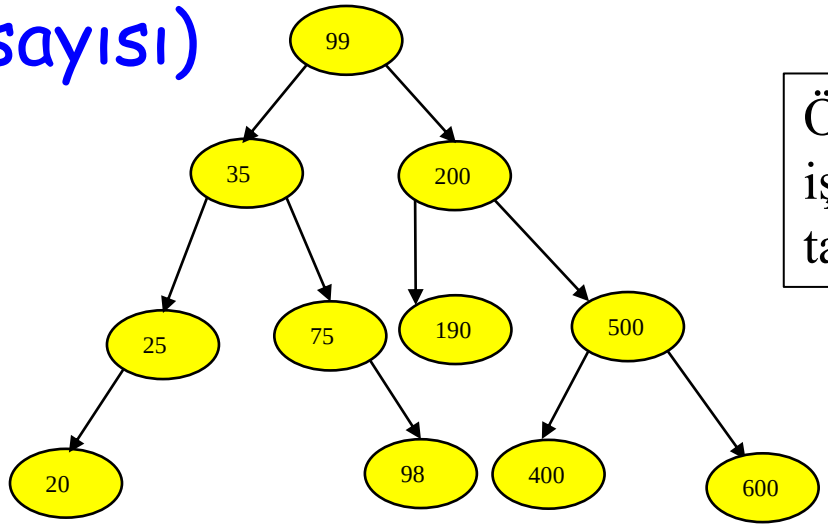






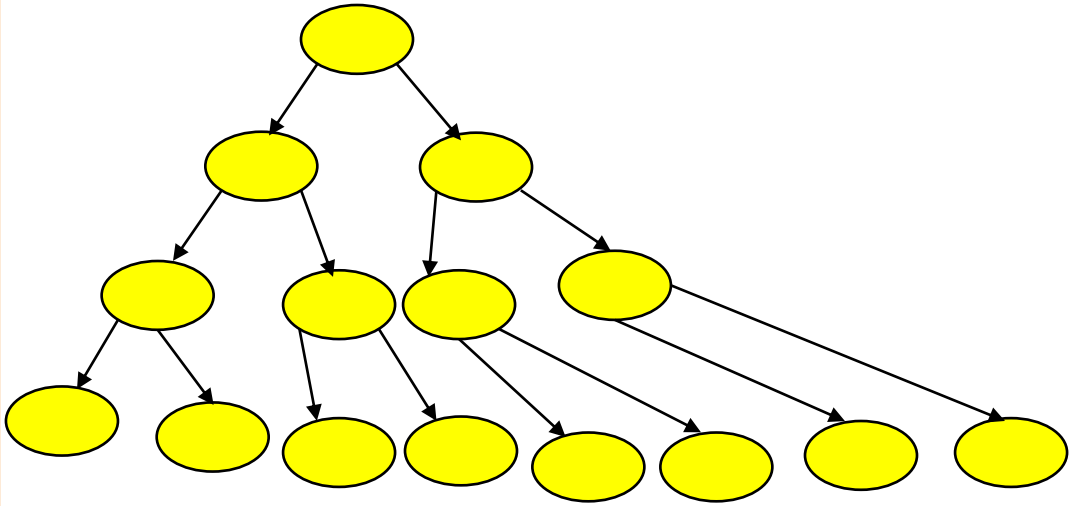
Dengeli İkili Ağaçlar

- Ağaç işlemlerinin hızlı olması açısından ağaç dengeli olmalıdır
- Eğer ağaç dengeli ise düğüm ekleme, silme ve arama işlemleri ağacın yüksekliği olan O(logn) sürede gerçekleşebilmektedir (n ağaçtaki düğüm sayısı)



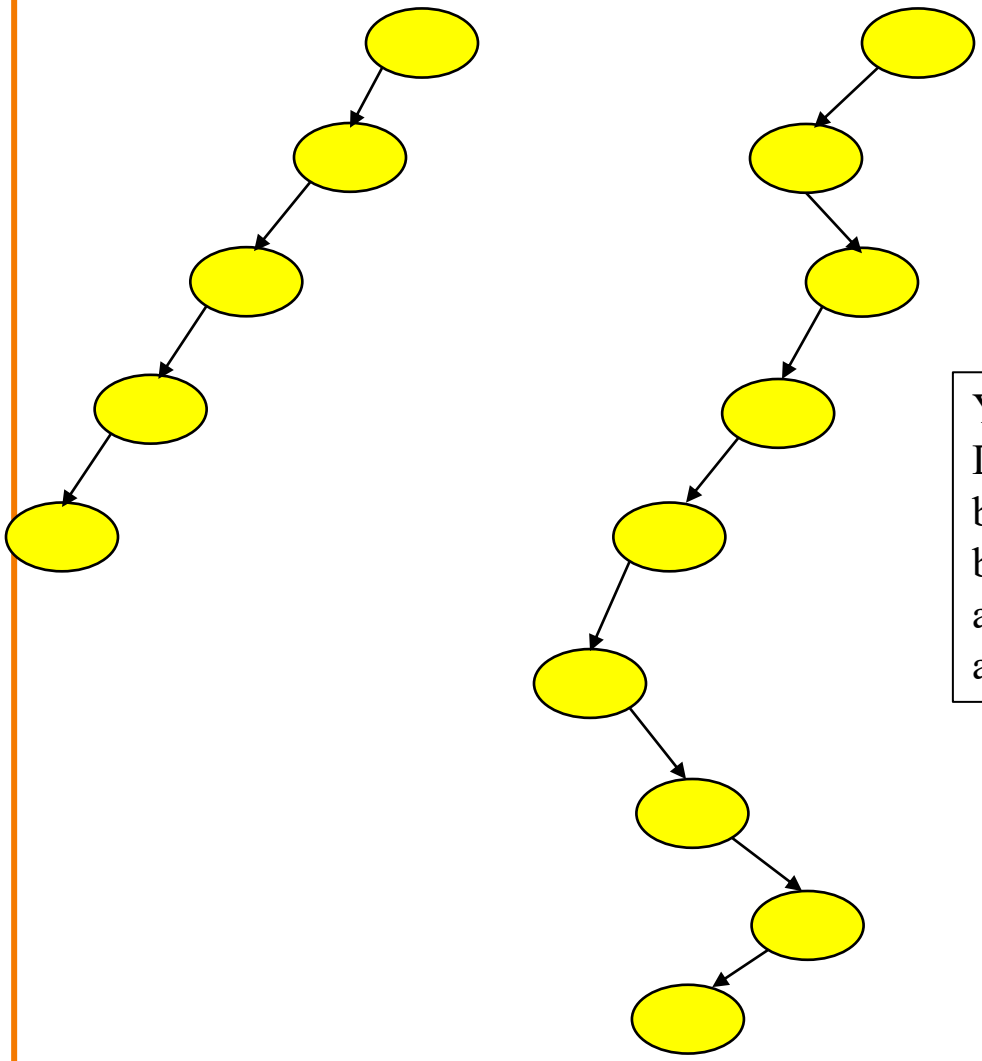
Örnek ağaçta n= 11. Dolayısıyla tüm işlemler O(logn) = 4 işlem süresinde tamamlanır.

Mükemmel Dengeli İkili Ağaç



n=15, dolayısıyla her bir işlem O(logn) = 4 işlem süresinde tamamlanabilir

Denge Açısından En Kötü Ağaç



Yandaki ağaçta düğüm sayısı n=9 dur. Denge bakımından en kötü ağaçlardan biridir. En kötü durumda işlemler O(n) =9 birim işlem süresinde tamamlanır. Bu tip ağaçlara doğrusal (linear) ağaç denir ve bu tip ağaçlar üzerindeki işlemler en uzun zaman alır.