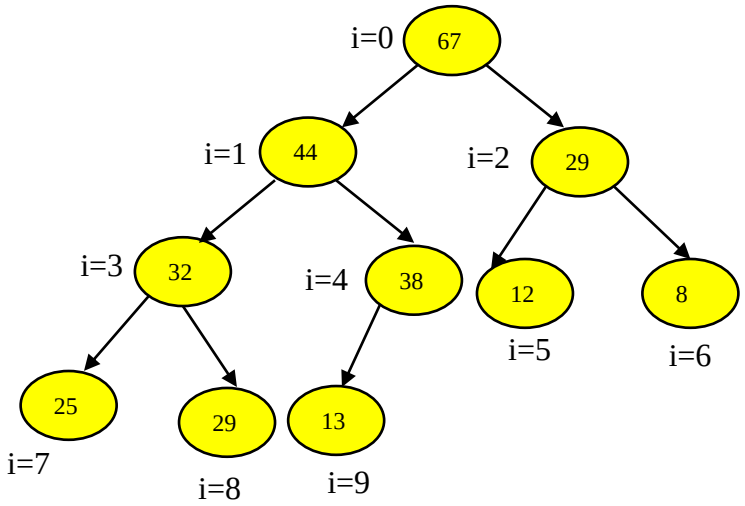
Heap Veri Yapısı

- Özel bir dengeli ikili ağaç
  - ✓ İkili ağaç dizi kullanılarak gerçekleştirilir
- Bu ağaçta düğümler ağacın seviyelerine yerleştirilirken en soldan başlanılır
  - ✓ Bir düğümün sol çocuğu yoksa sağ çocuğu da yoktur
- İki farklı çeşidi var
  - ✓ Max Heap: Her bir düğümün anahtarı sol ve sağ çocuklarının anahtarlarından büyük
  - ✓ Min Heap: Her bir düğümün anahtarı sol ve sağ çocukların anahtarlarından küçük
- Biz Max Heap üzerine odaklanacağız
- Min Heap işlemleri Max Heap işlemlerinin benzeridir

Max Heap



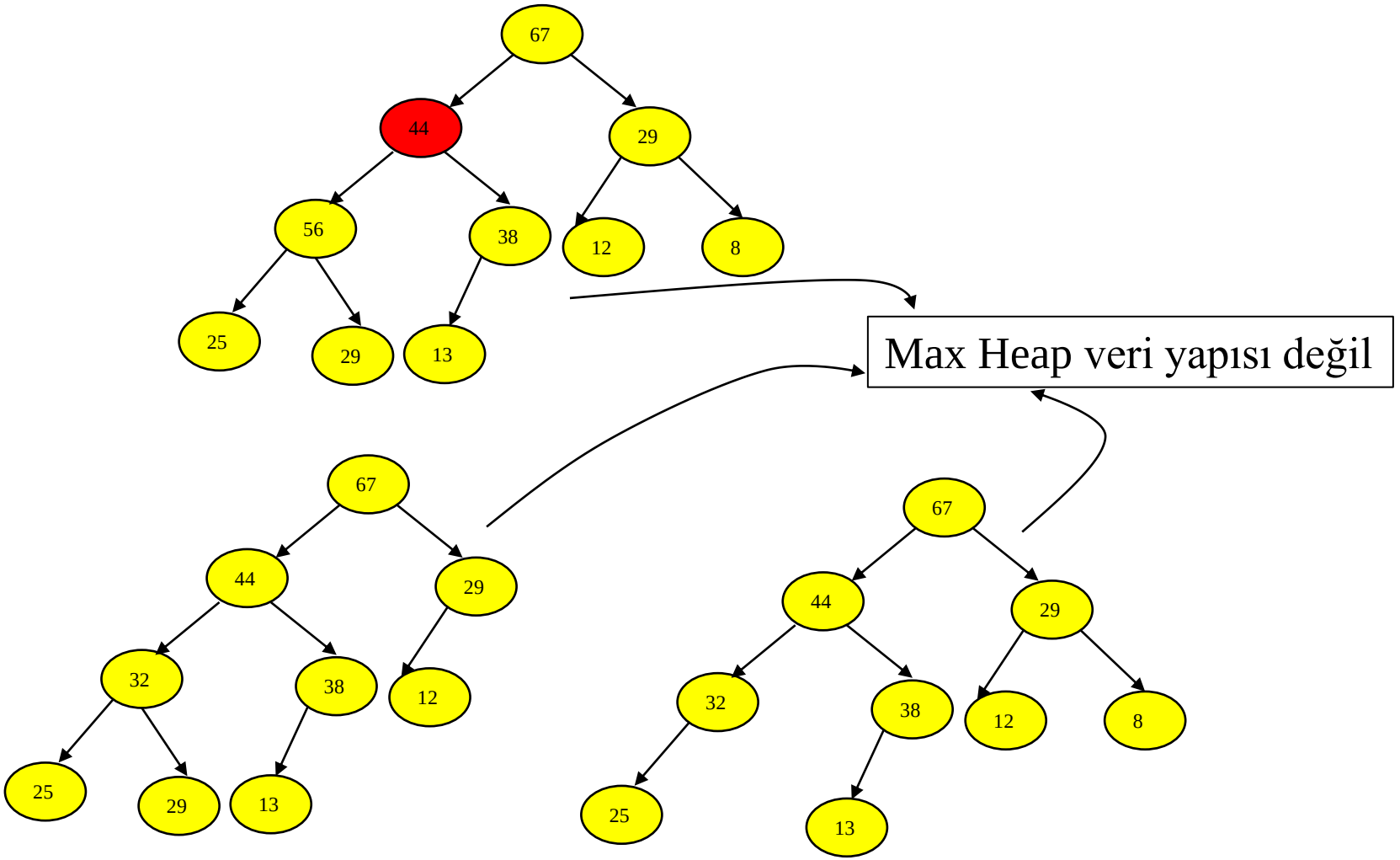
Max Heap veri yapısının 2'li ağaç şeklinde gösterimi

67	44	29	32	38	12	8	25	29	13

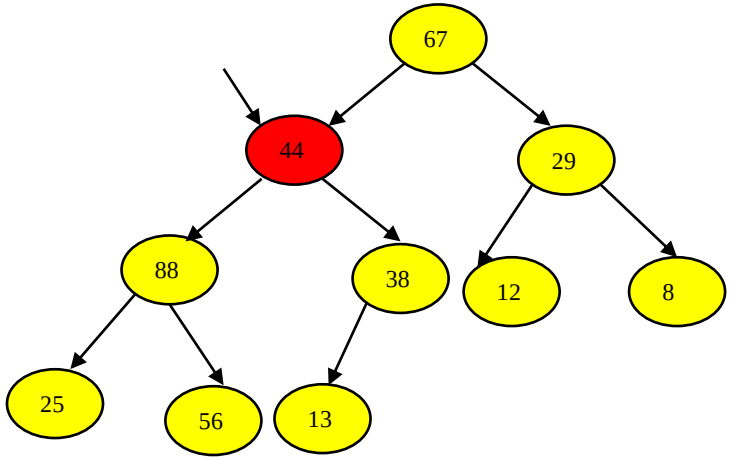
Max Heap veri yapısının dizi şeklinde şeklinde gösterimi

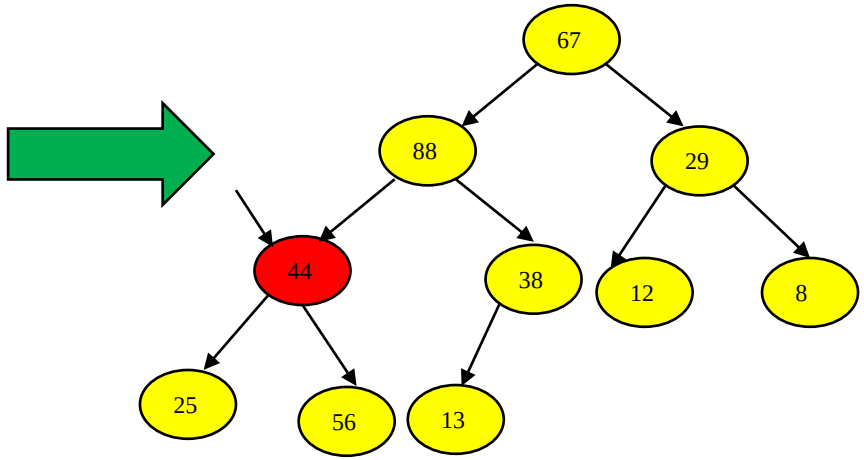
Kökün indeksi 0 olmak üzere, n indeksli bir düğümüm sol çocuğunun 2 x n +1, sağ çocuğunun indeksi ise 2 x n + 2 şeklinde hesaplanır

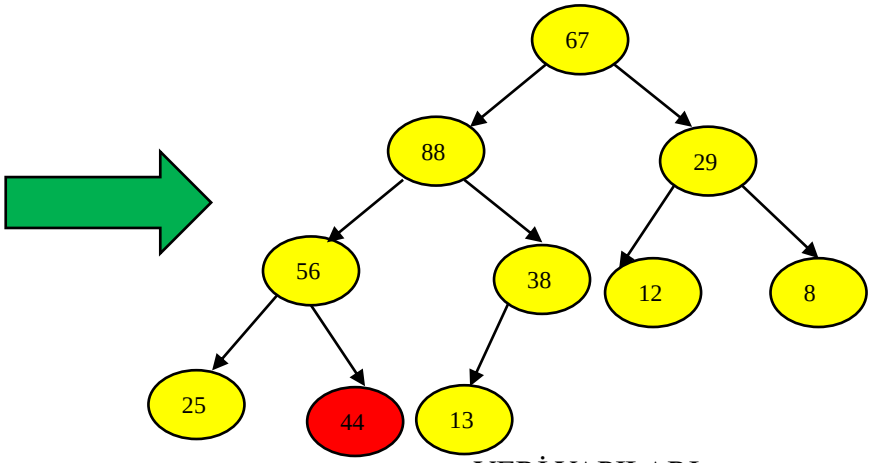
Max Heap Özelliğini Sağlamama



buble_down algoritması







Sol ve sağ çocuklardan anahtarı büyük olan çocuğun anahtarı parentin anahtarından büyükse, anahtarları değiş tokuş et. Bunu heap özelliği sağlanana kadar devam ettir

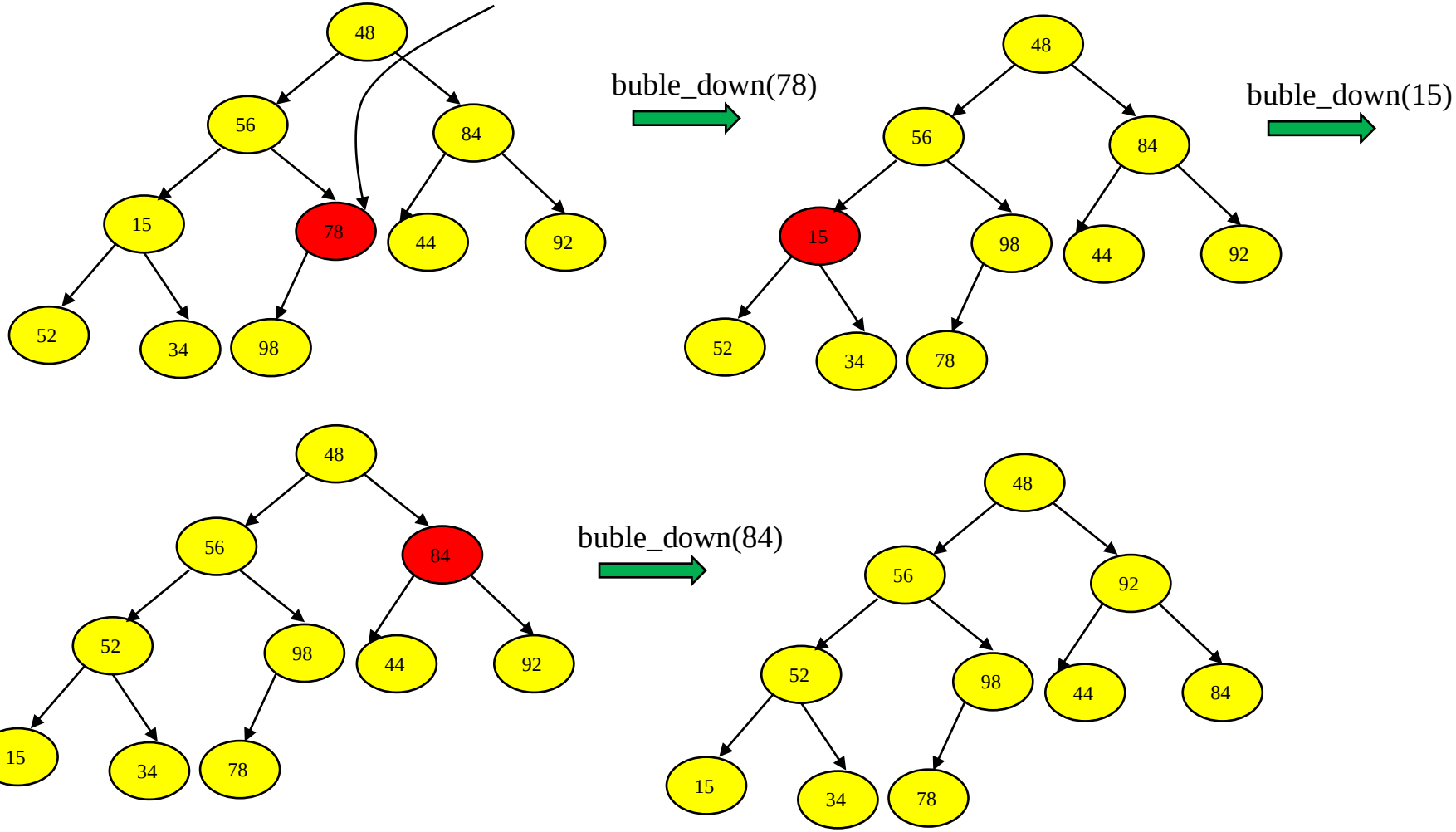
heapify algoritması

- Verilen bir diziyi Max Heap veri yapısına dönüştürür
- Yaprak olmayan en büyük indekse sahip düğümden başlamak üzere en son kök düğüme kadar buble_down algoritması uygulanır

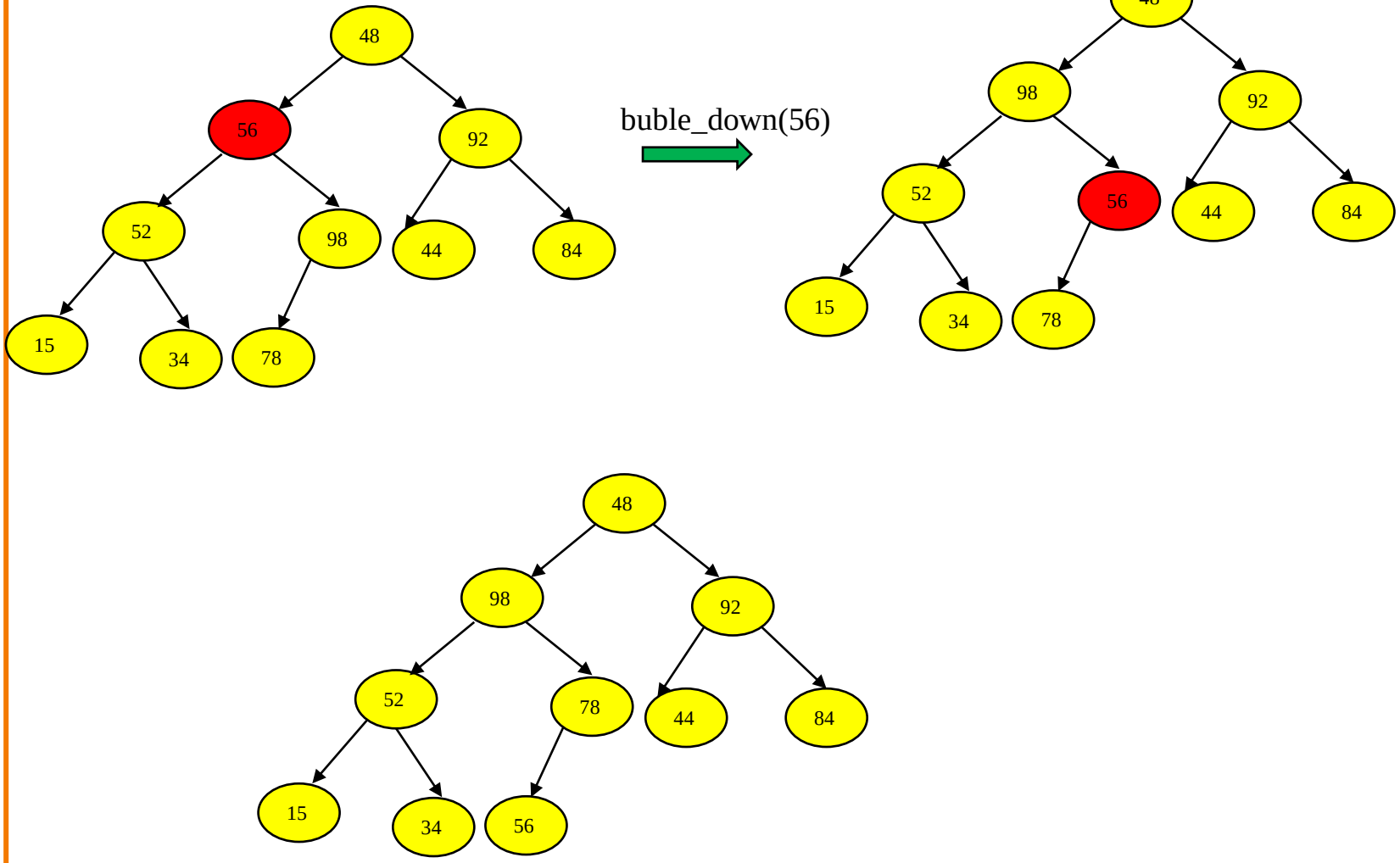
```
for(i=heap->eleman_sayisi/2-1; i>=0; i--) buble_down(heap,i)
```

Heapify algoritması

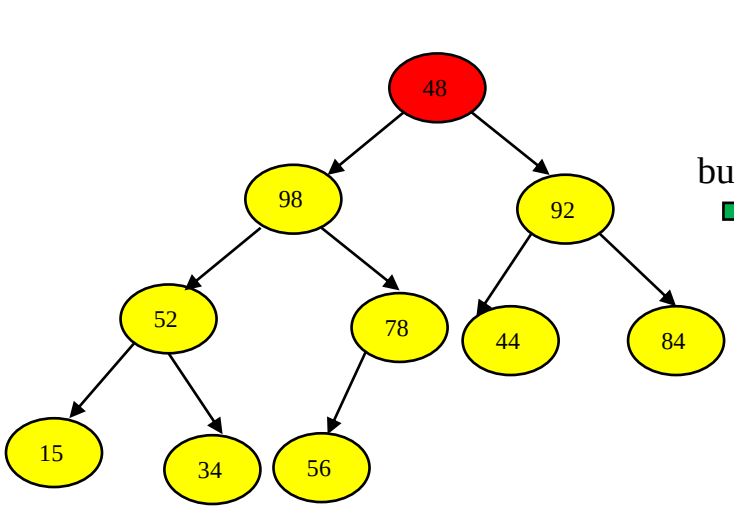
Yaprak olmayıp, en büyük indekse sahip düğüm



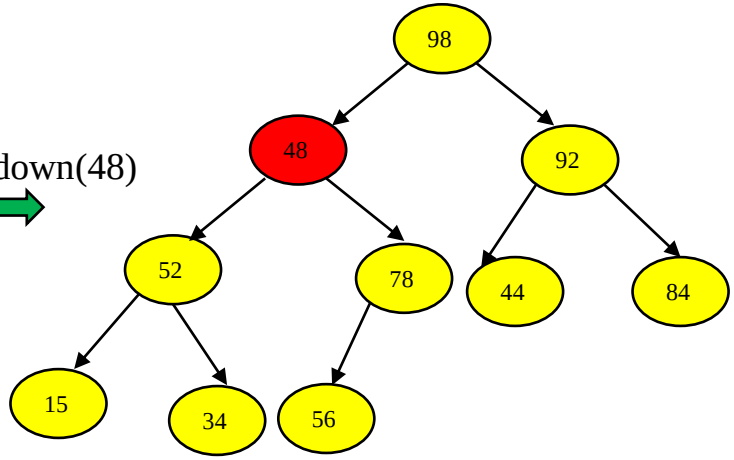
heapify algoritması devam

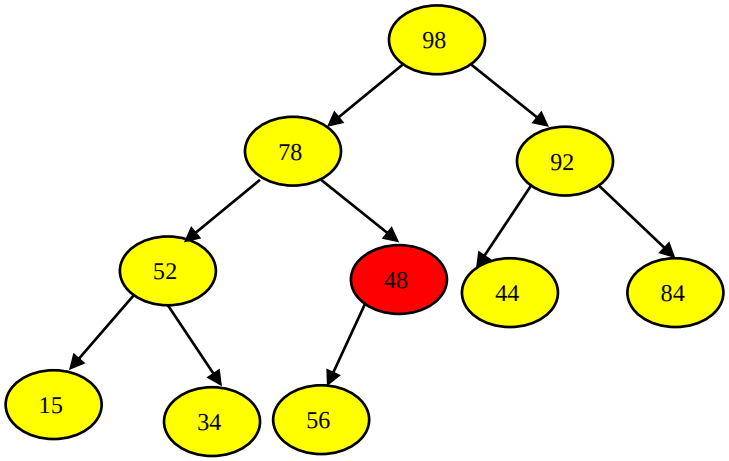


heapify algoritması devam

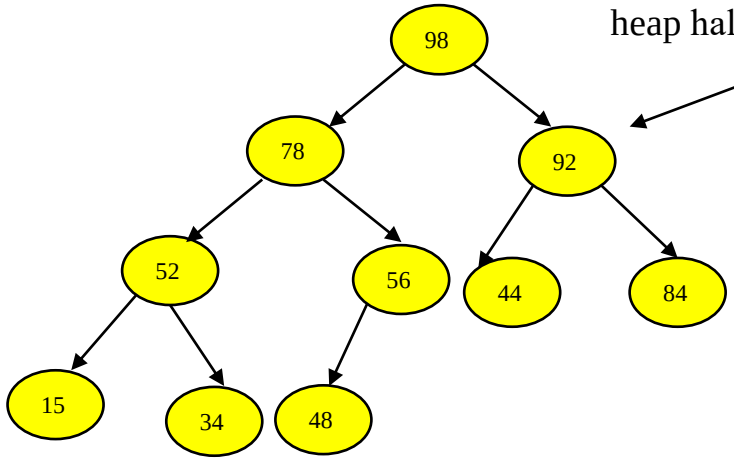


buble_down(48)





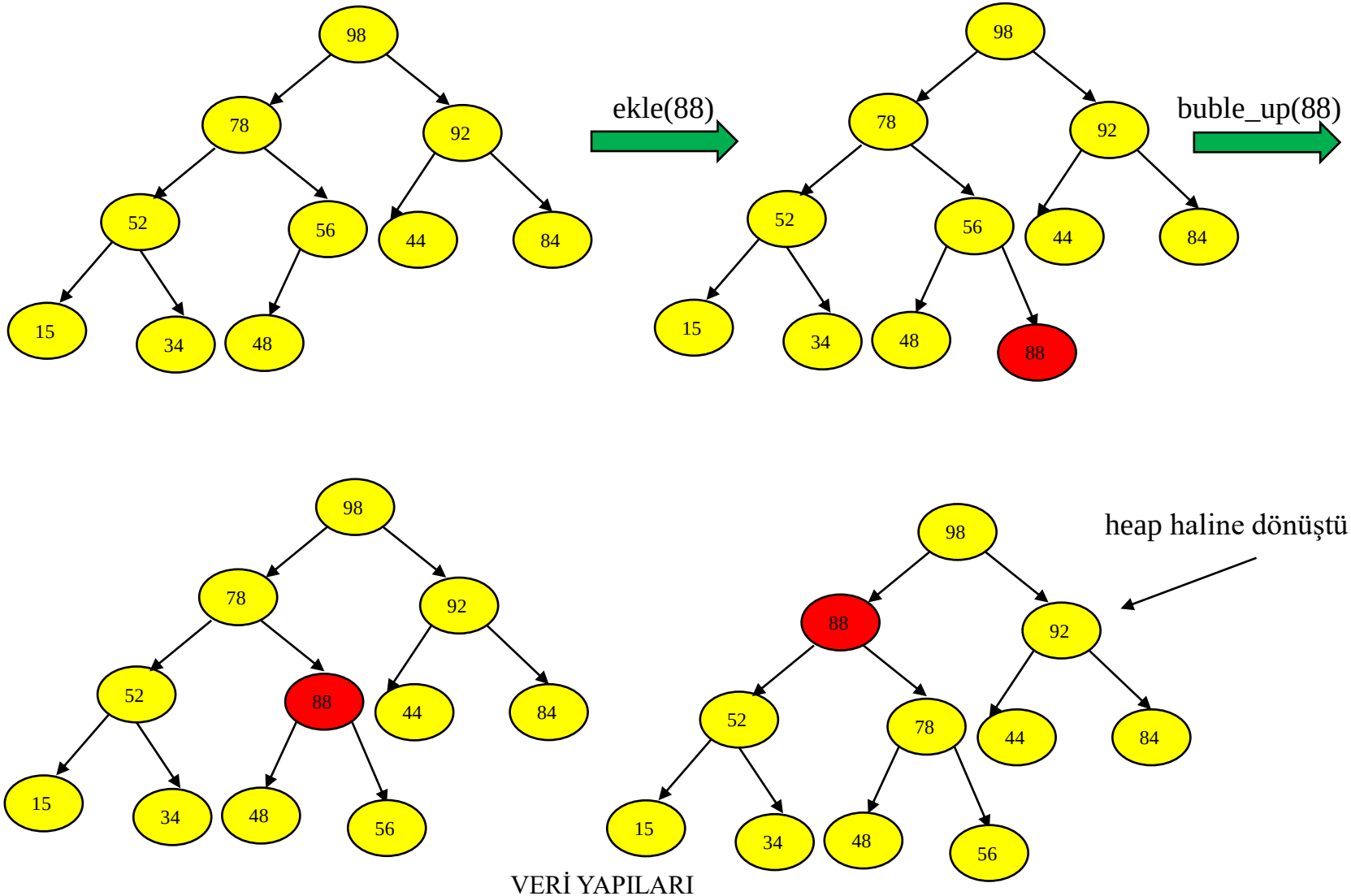
heap haline dönüştü



buble_up algoritması

- Heap'e düğüm eklemelerinde kullanılır
- Düğüm heap'in sonuna eklenir
- Ekleme heap özelliğini bozabileceğinden, yapıyı tekrar heap haline getirmek için eklenen düğümden başlanmak üzere buble_up algoritması uygulanır
  - ✓ Eklenen düğümden başlamak üzere köke doğru olan yolda, düğümün anahtarı parentin anahtarından büyükse anahtarlar değiş tokuş ettirilir

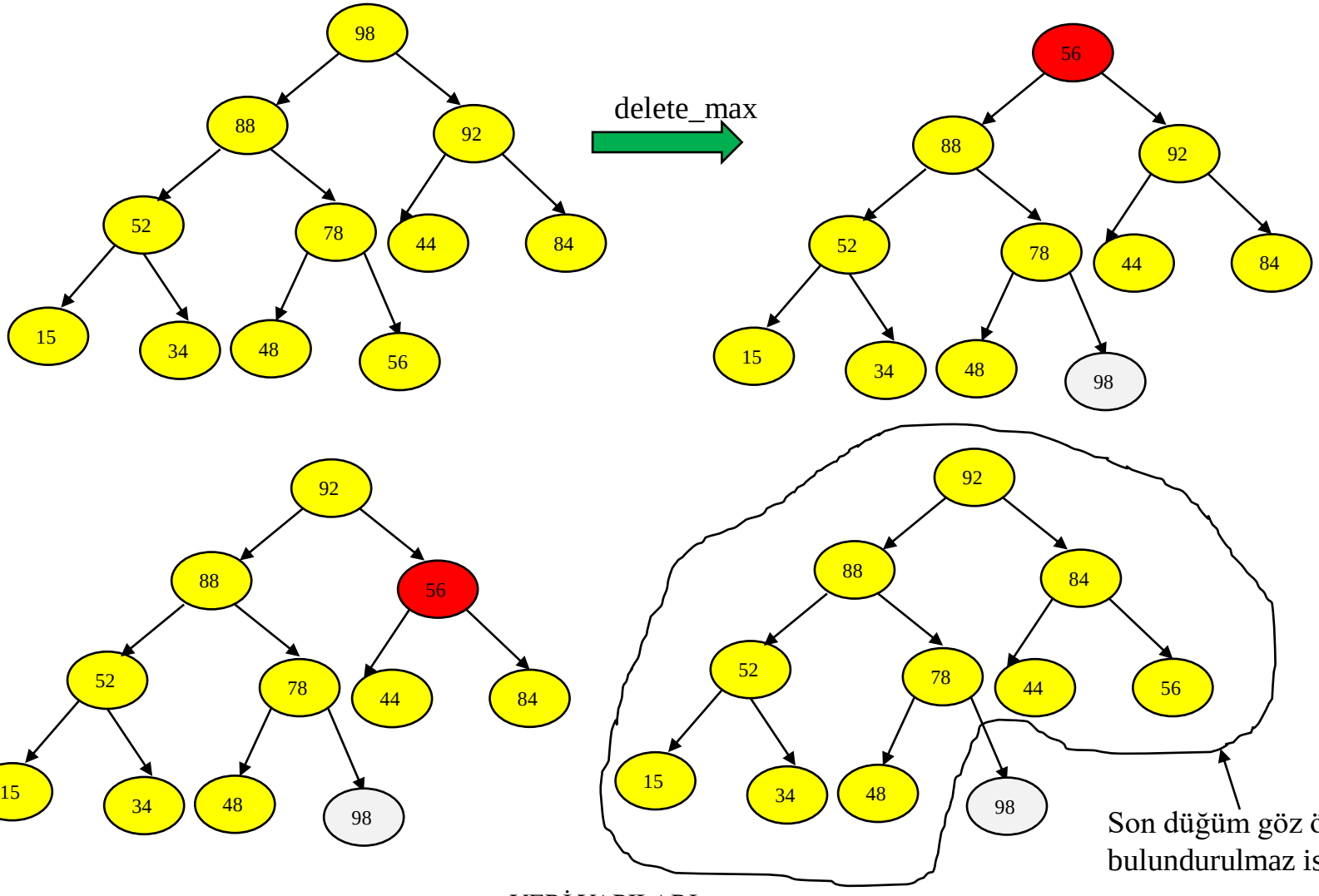
buble_up algoritması



delete_max algoritması

- Heap veri yapısındaki son eleman ile ilk elemanın (düğüm) yerleri değiştirilir
- En son eleman (başlangıçta düğüm konumundaki) hariç tutulmak üzere kökten başlamak üzere köke heap_down algoritması uygulanır
- delete_max, heap_sort algoritmasında (sıralama algoritması) kullanılır

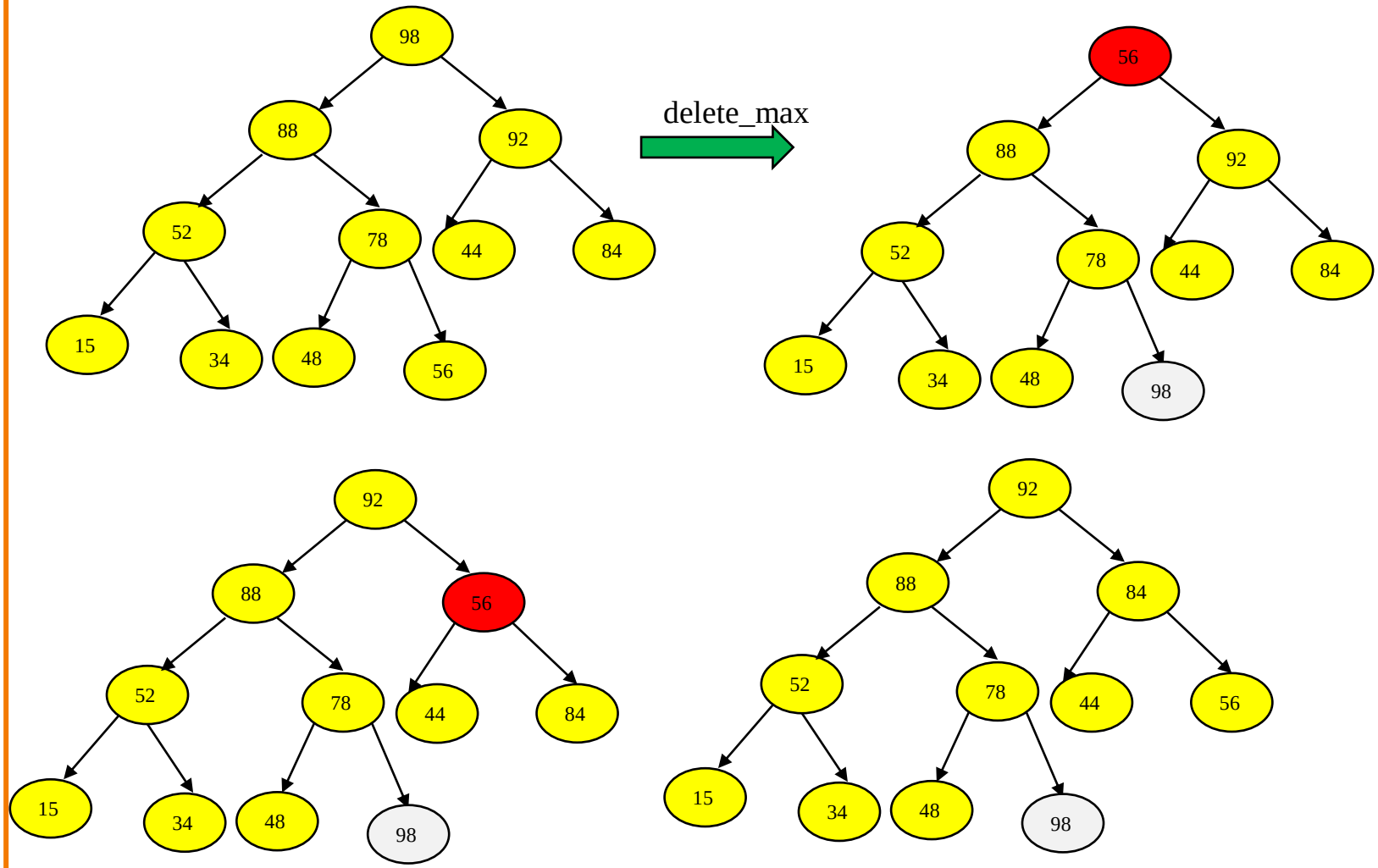
delete_max algoritması



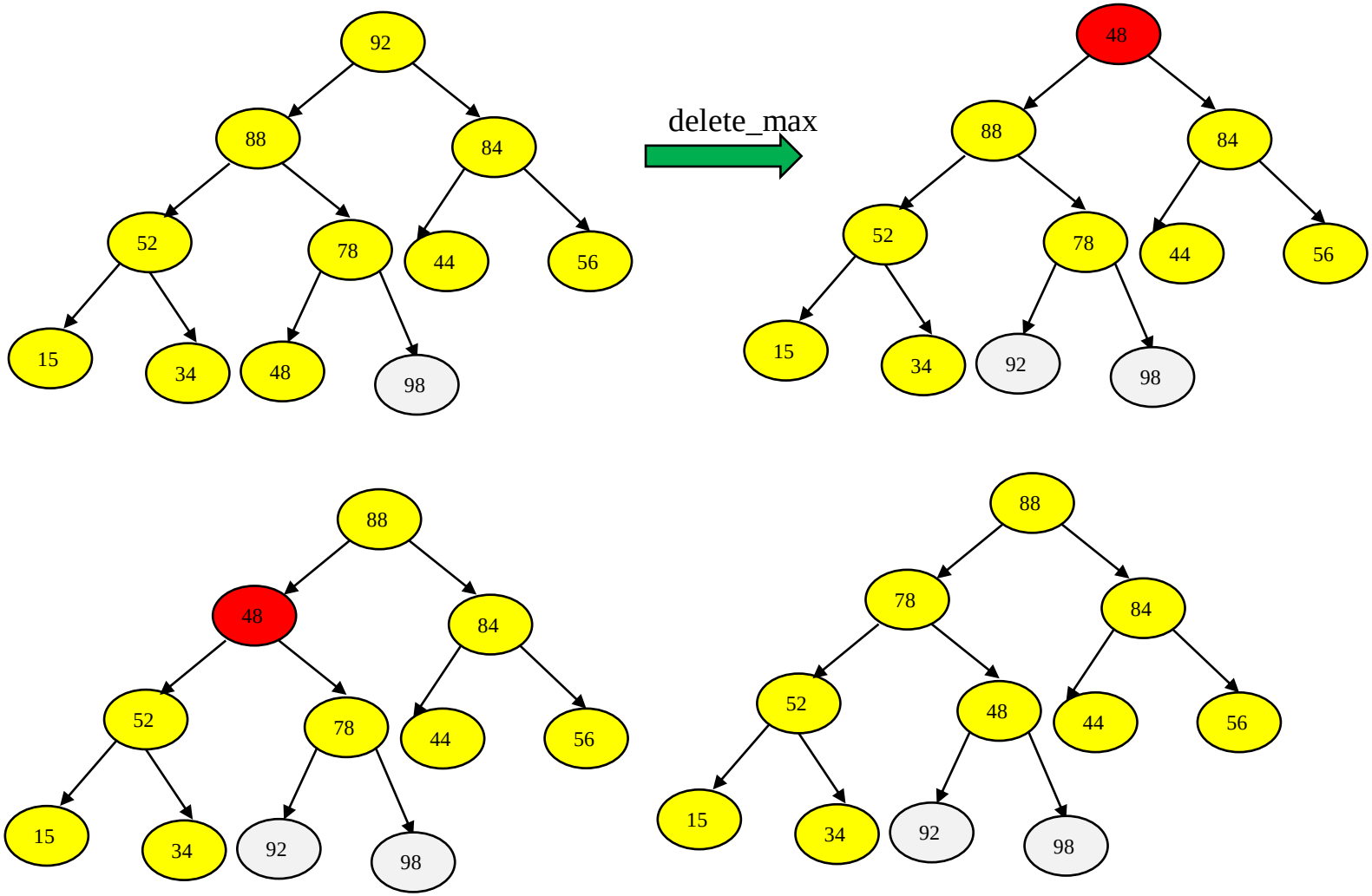
heap_sort algoritması

- n pozitif bir tam sayı olmak üzere n düğüme sahip bir heape n-1 kez delete_max algoritması uygulanırsa veri yapısı küçükten büyüğe doğru sıralanmış olur

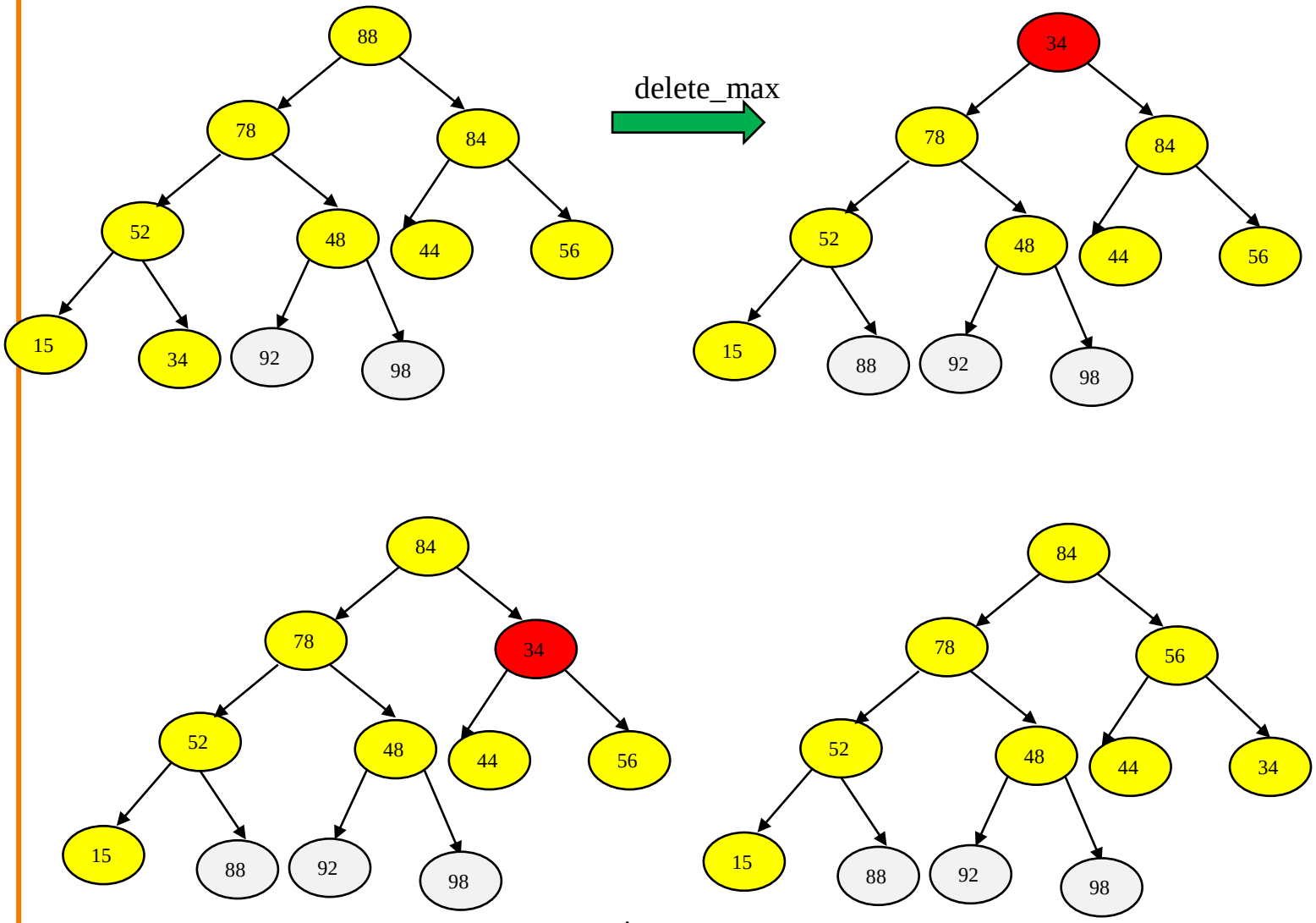
heap_sort algoritması



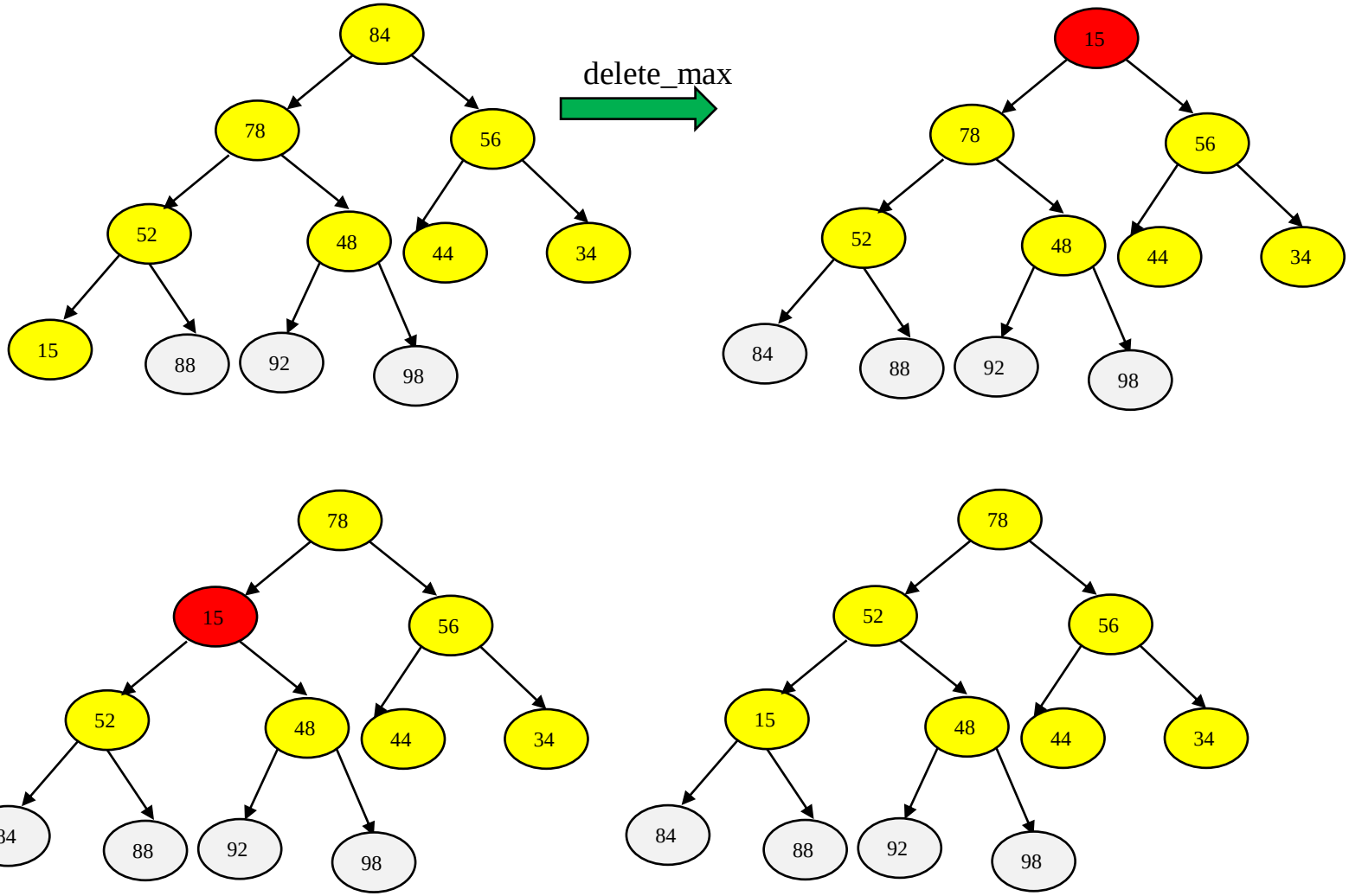
heap_sort algoritması devam



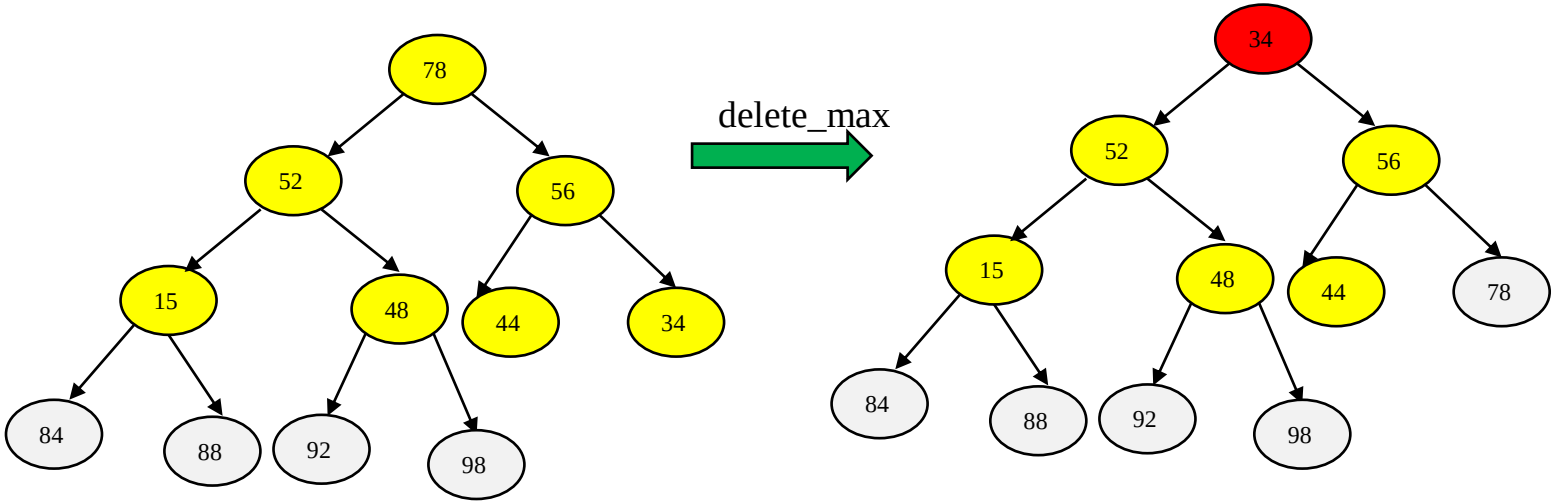
heap_sort algoritması devam

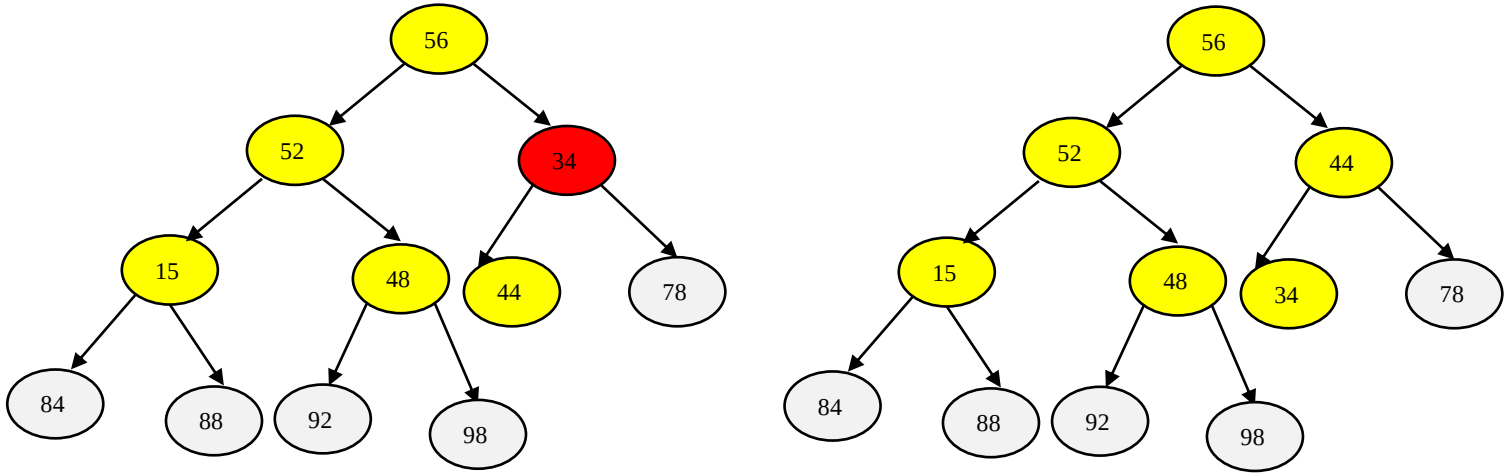


heap_sort algoritması devam

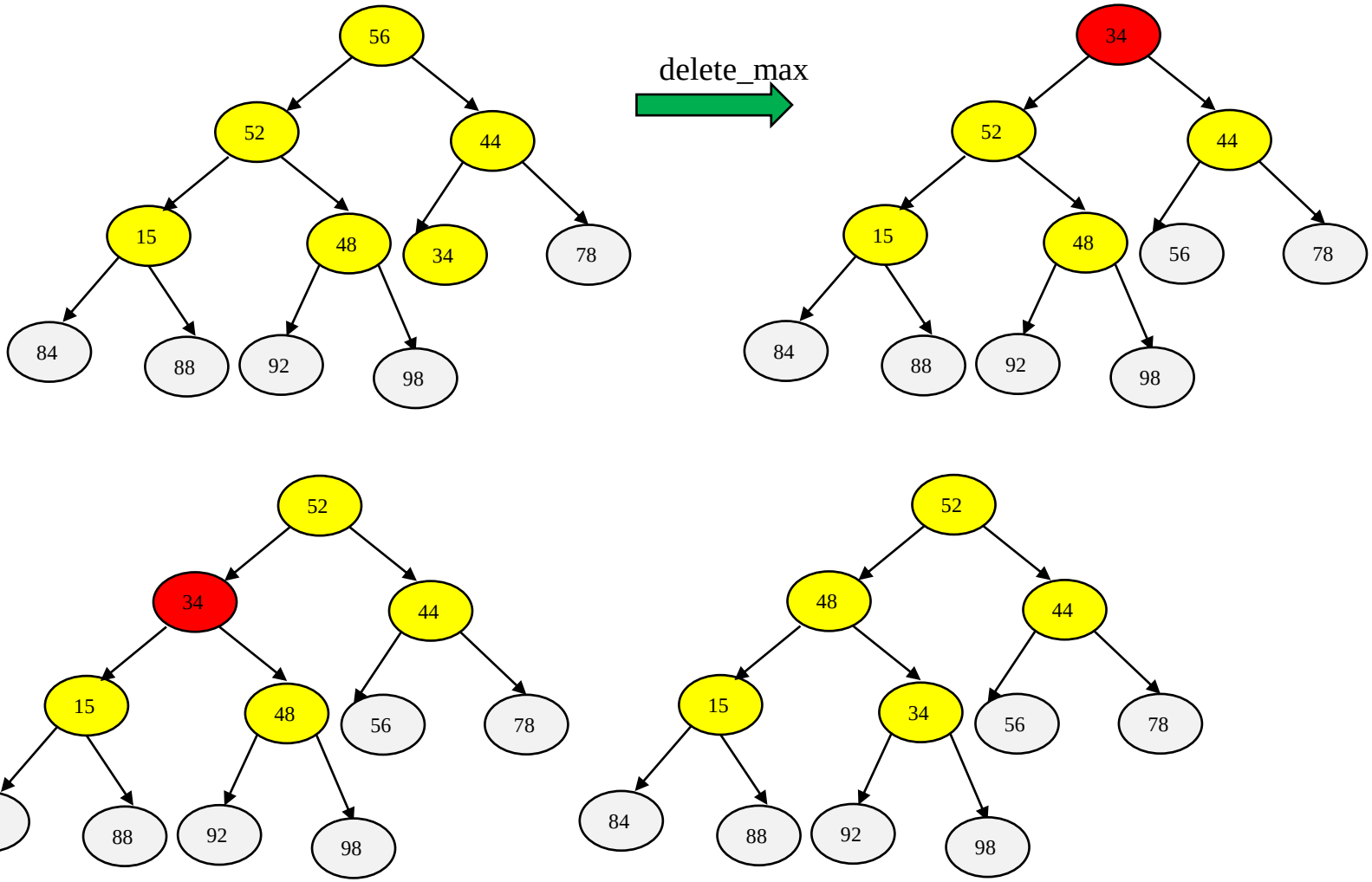


heap_sort algoritması devam

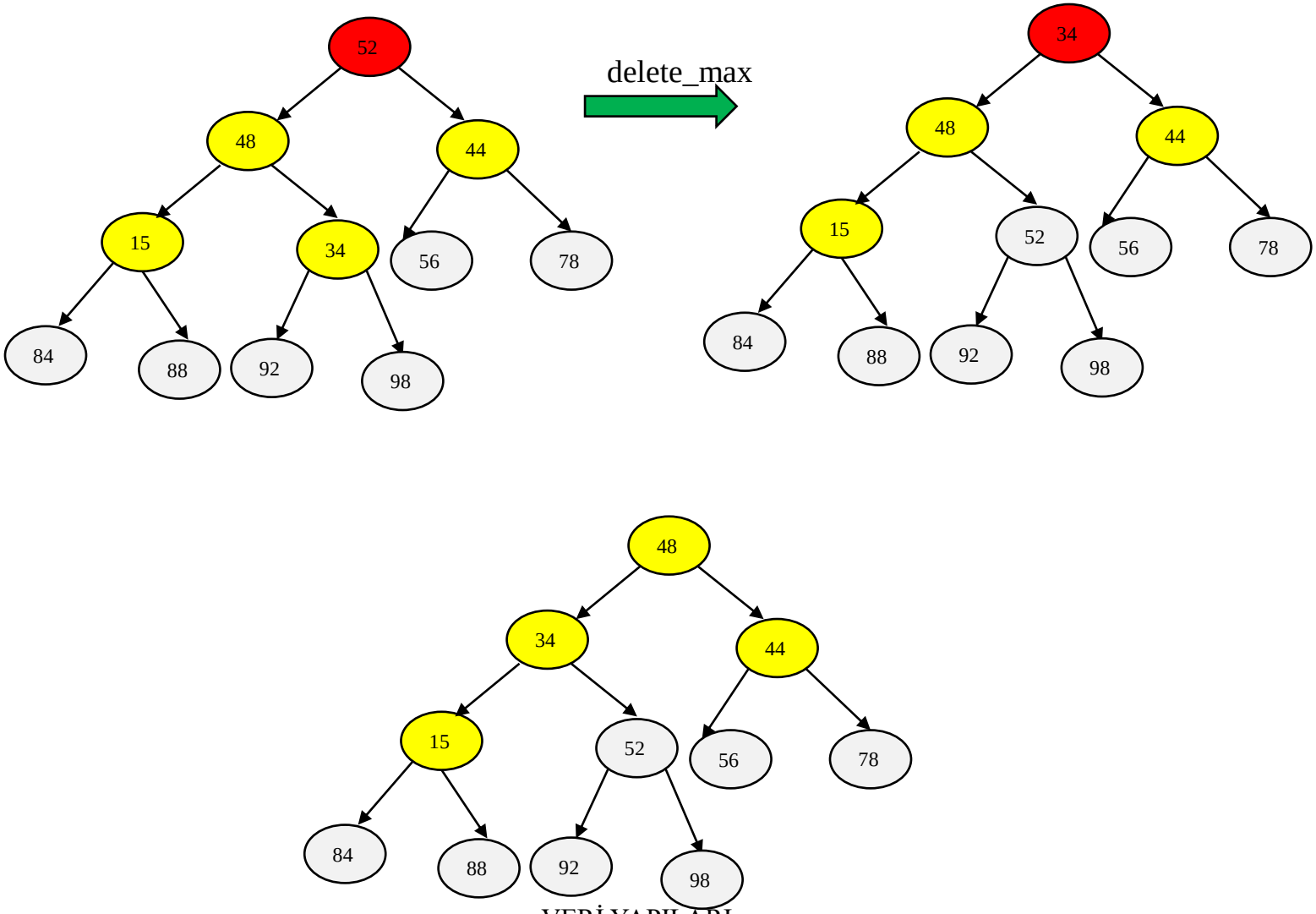




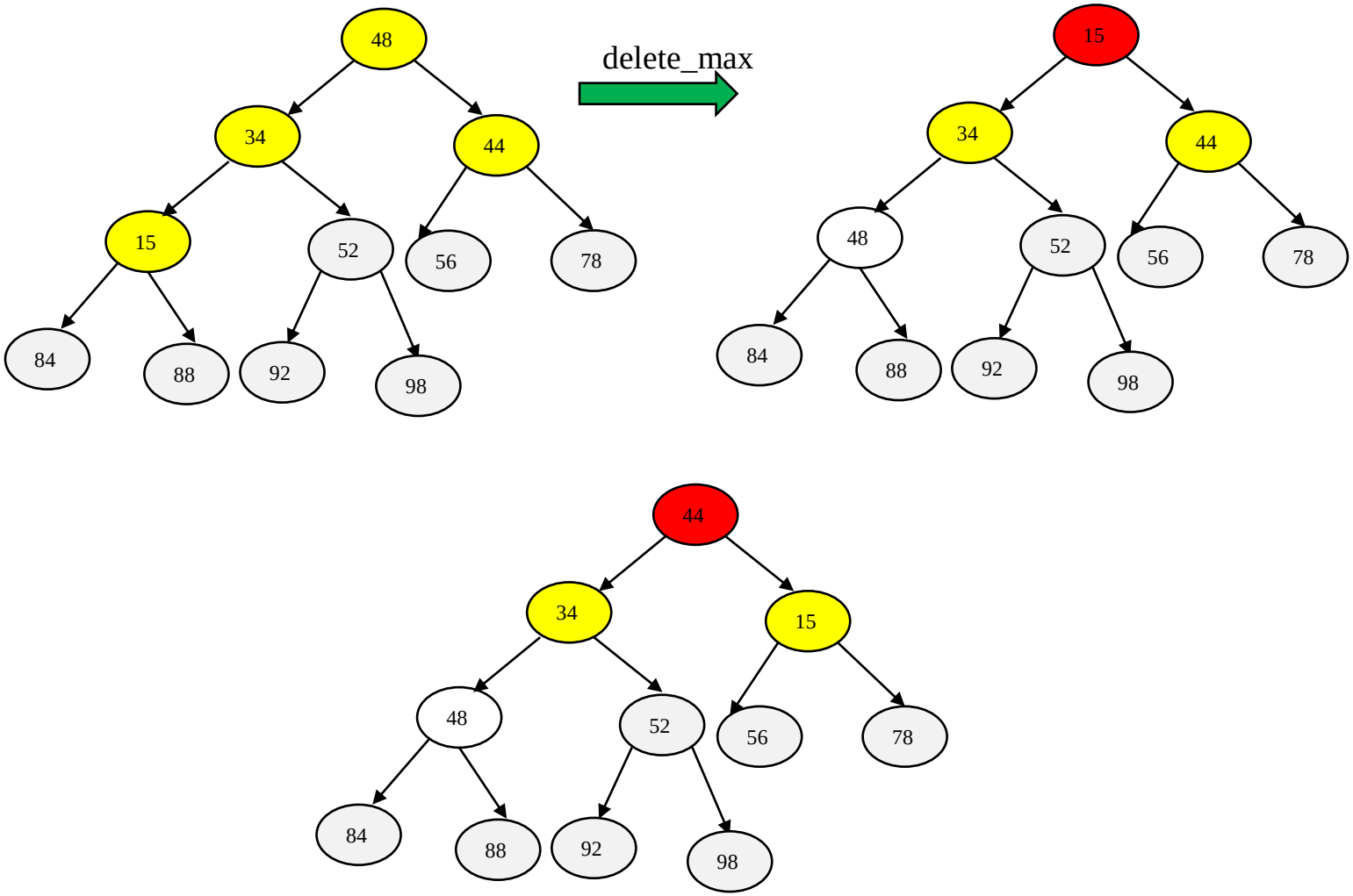
heap_sort algoritması devam



heap_sort algoritması devam



heap_sort algoritması devam



heap_sort algoritması devam