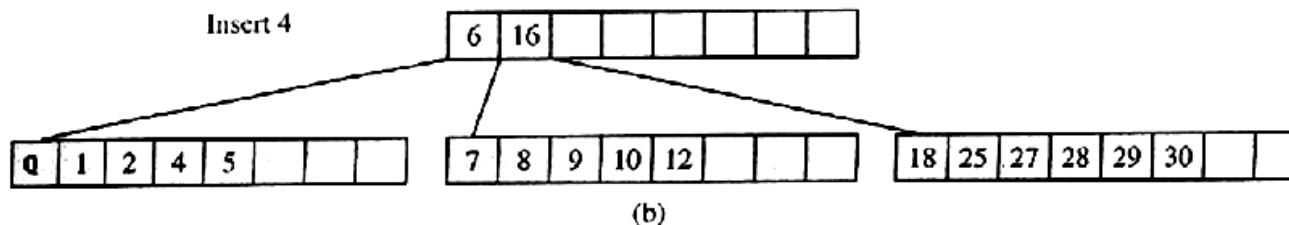
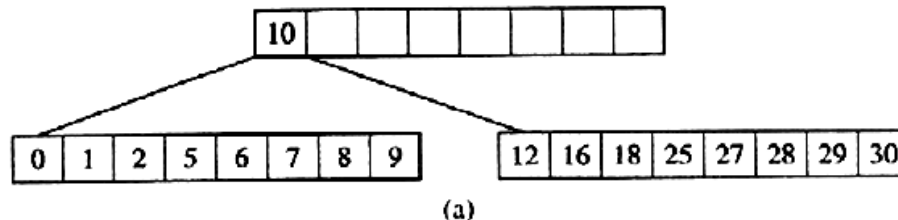


## Çok Yollu Ağaçlar: B\*-Trees

- B-tree'lerde bir node dolunca bölme işlemi yapılmaktadır
- Bölme sonucunda oluşan iki node'da yarı yarıya doludur
- B\*-tree'lerde bölme işlemi geciktirilerek node'ların doluluk oranı artırılır.
- Ortalama Ekleme süresi uzar ve ağacın yüksekliği daha azdır.
- Tüm ağacın dolum faktör değeri B-tree'lere göre daha yüksektir.
- B\*- tree' lerde erişim performansı daha yüksektir. Literatürde B#-tree şeklinde variantları vardır.

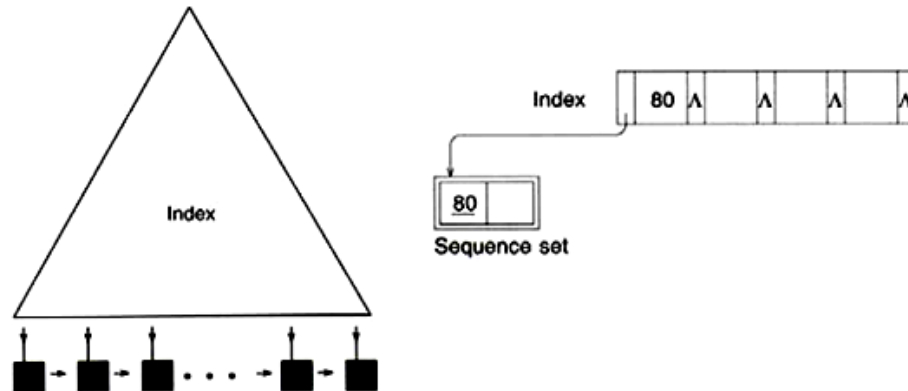
# Çok Yollu Ağaçlar : B\*-Trees

- $m$ . dereceden bir B\*-tree'de, kök olmayan her node'daki anahtar sayısı ( $k$ )  $(2m-1)/3 \leq k \leq m-1$  olarak bulunur. Bunun anlamı B-tree deki herhangi bir düğümün doluluk oranını  $2/3$  oranında tutar.
- Node bölme işlemi B-Tree'dekine göre daha yavaştır. Tüm anahtarlar yeniden dağıtılır. ( $16/24 = 2/3$  oranı var  $16/3 =$  yaklaşık 5 değer gelecek)  
Örneğin: 8 düğüm var yeni düğüm geldi  $8 * 2/3 = 5$  düğüm olacak şekilde yeniden düzenle.
- 0,1,2,4,5,6,7,8,9,10,12,16,18,25,27,28,29,30



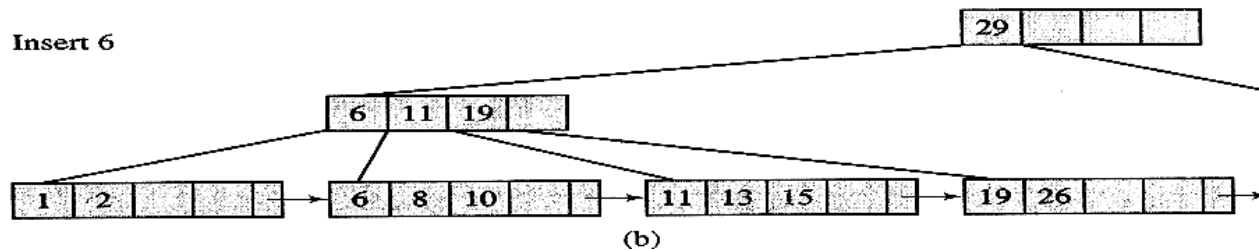
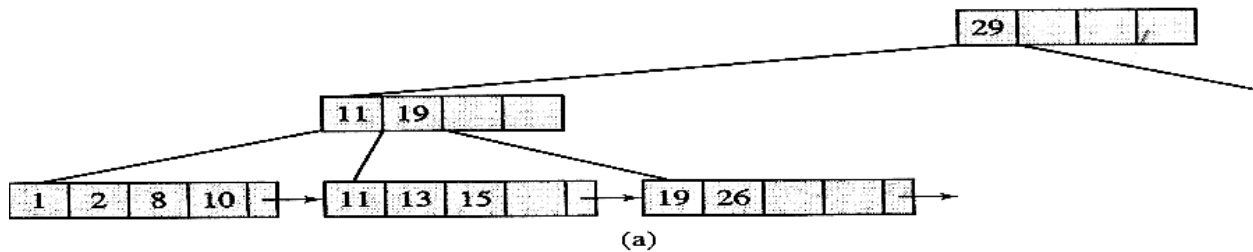
## Çok Yollu Ağaçlar : B+ Trees

- B+ tree'lerde sıralı okuma için parent'a ulaşmaya gerek yoktur. Bilgiler yapraklarda bulunur.
- Yaprak olmayan düğümler (nonleaf node) sadece indeks için kullanılır. Tüm yapraklar tek bağlı listeye bağlanır B+ tree'lerde indeks kısmı B tree ile aynıdır.
- İndeks kısmındaki tüm kısıtlamalar ve işlemler B tree ile aynıdır
- Internal node'lar index-set olarak, yapraklar ise sequence set olarak adlandırılmaktadır.



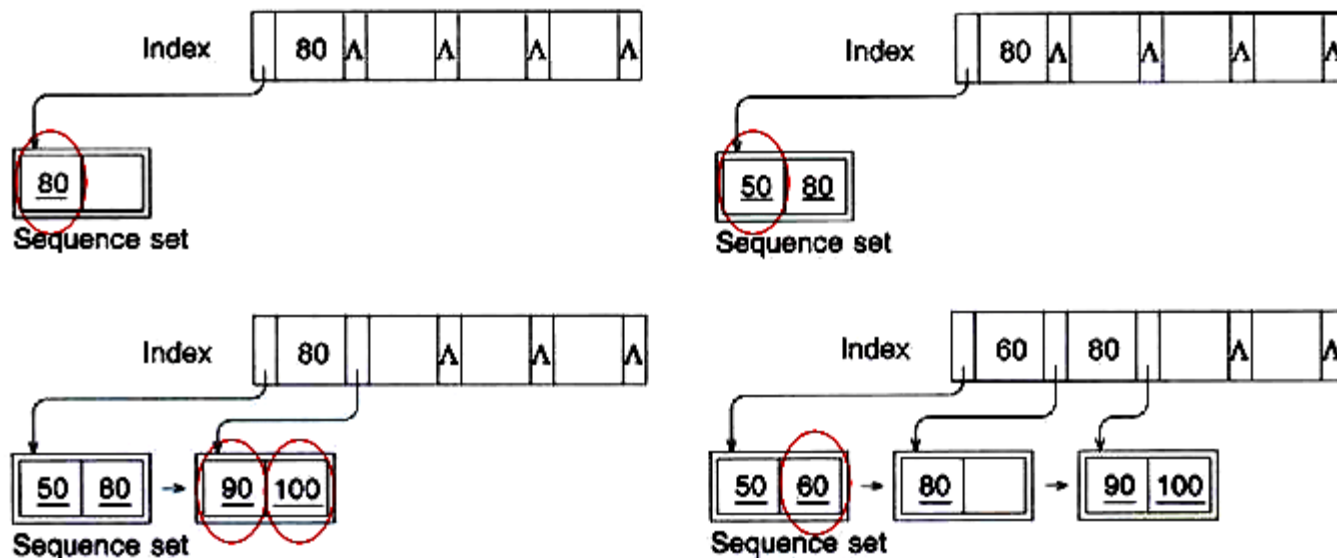
# Çok Yollu Ağaçlar :B+ Trees

- Diğer bir ifadeyle;
- B Tree'de referans herhangi bir node ile yapılabilir.
- B+ Tree'de ise referans sadece yaprak node'ları ile yapılabilir.



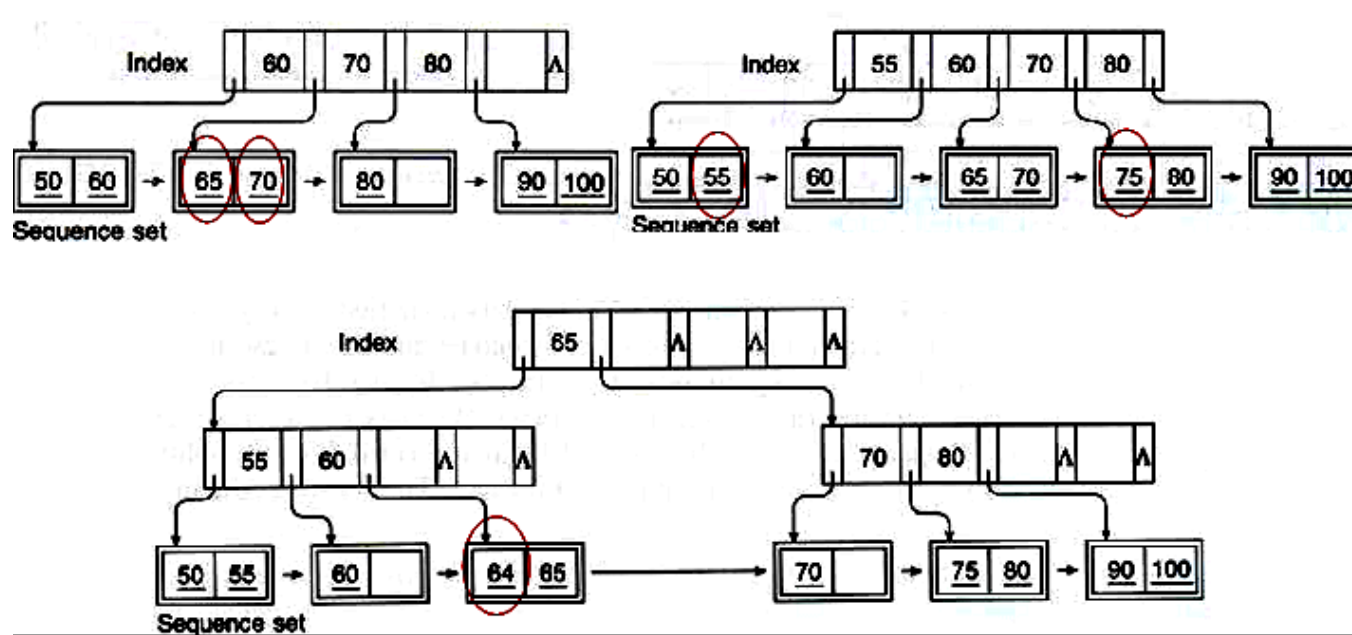
# Çok Yollu Ağaçlar: B+ Trees

- **B+ Trees**
- Örnek
- $d = 2$  (capacity order)
- $s = 1$  (sequence set order)
- 80,50,100,90,60,65,70,75,55,64,51,76,77,78,200,300,150



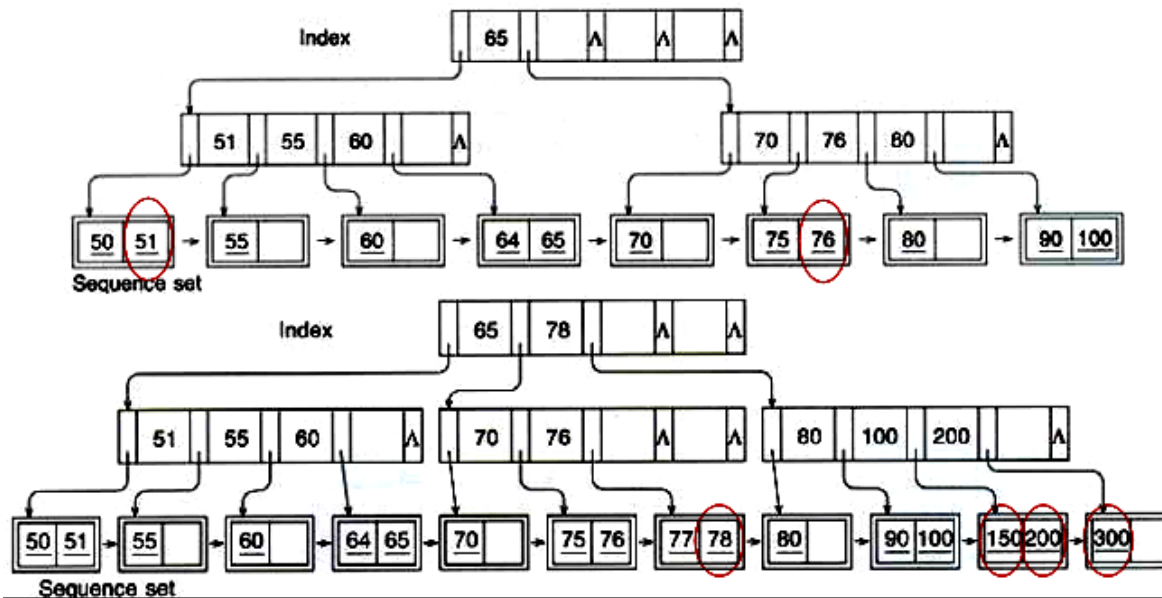
# Çok Yollu Ağaçlar –B+ Trees

- **B+ Trees**
- Örnek
- $d = 2$  (capacity order)
- $s = 1$  (sequence set order)
- 80,50,100,90,60,65,70,75,55,64,51,76,77,78,200,300,150



# Çok Yollu Ağaçlar –B+ Trees

- **B+ Trees**
- Örnek
- $d = 2$  (capacity order)
- $s = 1$  (sequence set order)
- $80, 50, 100, 90, 60, 65, 70, 75, 55, 64, 51, 76, 77, 78, 200, 300, 150$



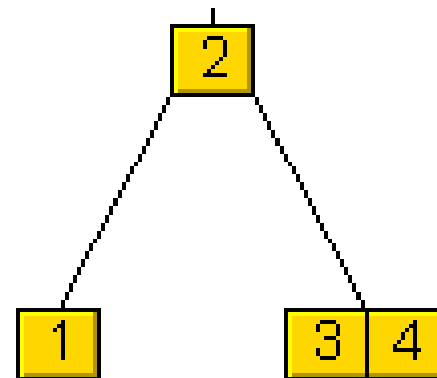
# Çok Yollu Ağaçlar –B+ Trees

- **Haftalık Ödev**
- B,B\*,B+ ağaçlarının kullanıldığı yerler hakkında araştırma yapınız. Literatür taraması yaparak elde ettiğiniz makaleleri inceleyiniz. Kullanıldığı yerlerde ne amaçla kullanıldığına yönelik bilgileri içeren bir rapor hazırlayınız .
- B#- tree hakkında bilgi toplayarak ekleme işlemi nasıl gerçekleştirilir araştırınız ve bir rapor hazırlayınız .
- B+ ve B\* ve B# - tree'lerde silme işleminin nasıl yapıldığını araştırınız ve bir rapor hazırlayınız .
- Bunlara ait program örneğini ve algoritmasını gerçekleştiriniz



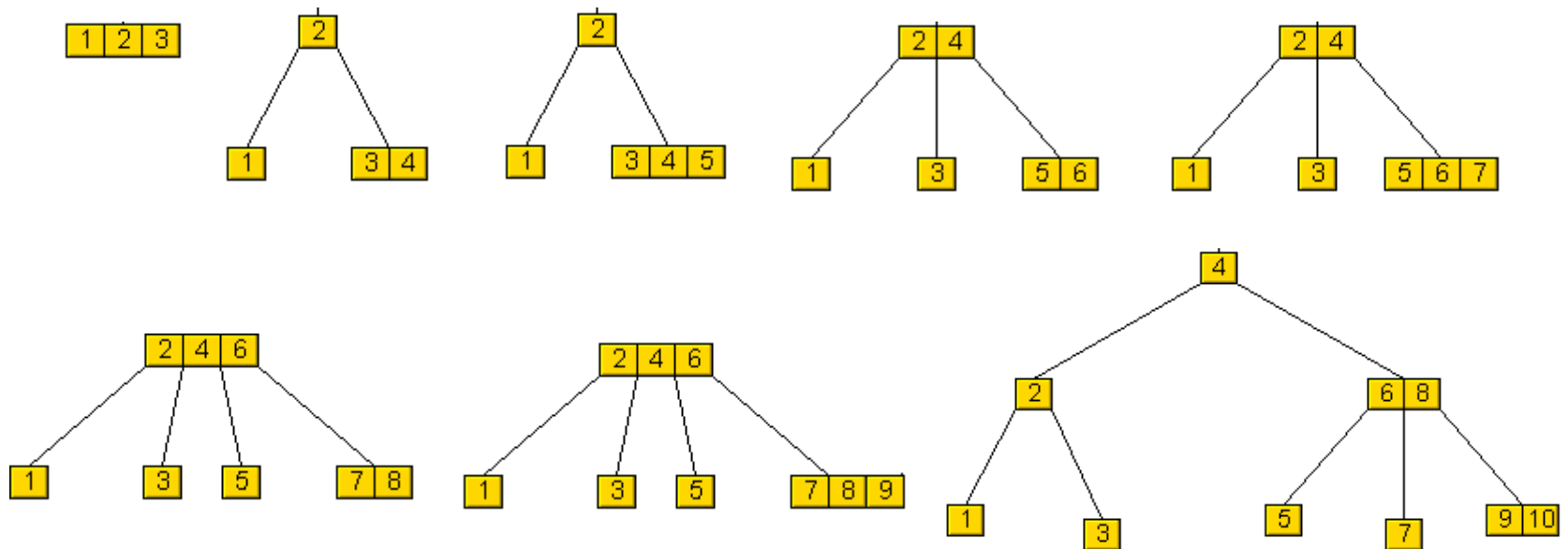
## 2-3-4 tree

- B-tree yapısının basit bir halidir. Daha önce verilen kurallar bu ağaç içinde aynıdır.
- 2-3-4 tree anlamı 2 düğüm var ise 3 çocuğu olur. 3 düğüm var ise 4 çocuğu bulunur.
- Ekleme işleminde eğer maksimum kapasite aşılsa eklenmeden önceki ortada bulunan düğüm kök alınır ekleme işlemi sonra yapılır.
- 1,2,3,4



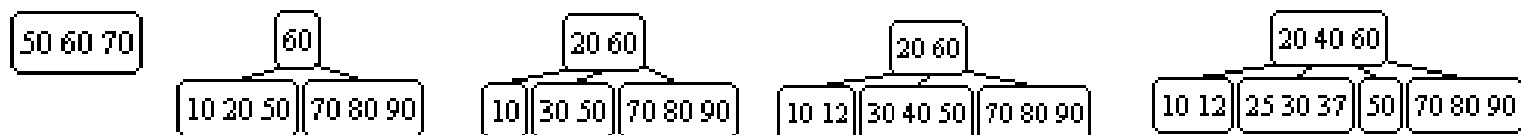
# 2-3-4 tree

Örnek: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

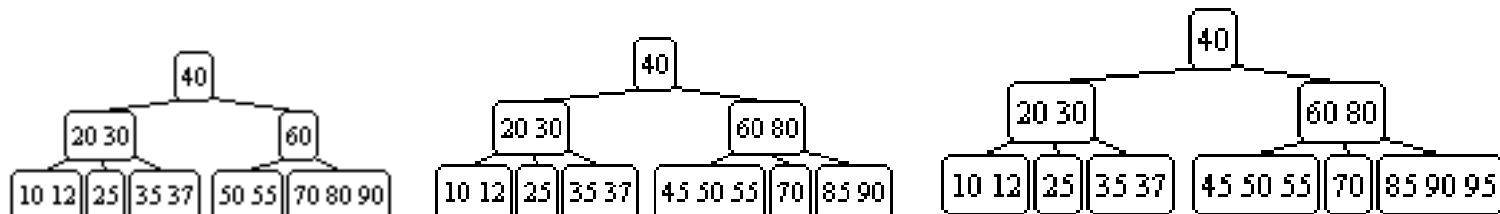


## 2-3-4 tree

- Örnek:,3 düğüm: Ekle: 50,70,60,90,20,10,80,30,40,12,25,37



- Ekle:55,35,45,85,95

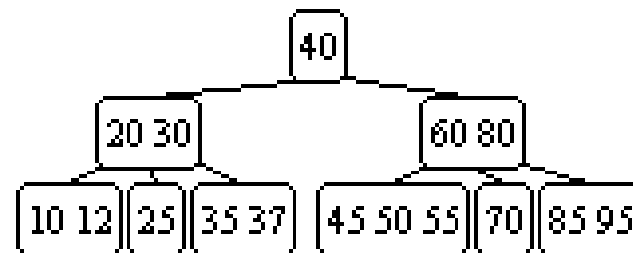
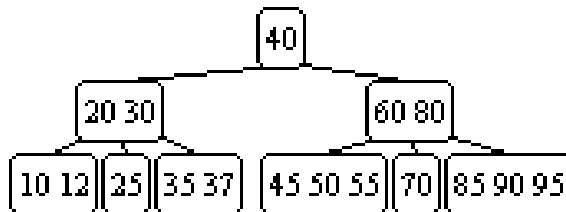


## 2-3-4 tree Silme

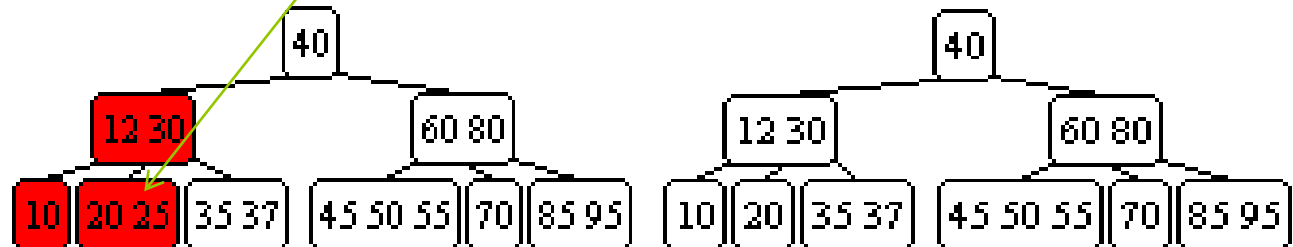
- Silme işlemlerinde düğümlere ait çocuk sayılarında denge bozulmuyorsa sadece kurala göre döner.(Soldaki en büyük veya sağdaki en küçük düğüm alınarak.)
- Sol veya sağa göre işlem yapıldığında öncelik çocuk sayısı fazla olmalıdır.
- Silme işleminde denge bozuluyorsa silinen düğümün kardeşi ve ebeveyni düzenlemeye girer.
- Çocukları olan silinecek düğüm tek ise ya birleştirme yada döndürme yapılarak yanına başka bir değer getirilip daha sonra silinir.
- Silinecek çocuğun atası tek ise atasının ebeveyni ve kardeşleri düzenlemeye girer.

# 2-3-4 tree

- **Kural 1-** Silinen düğüm yaprak ve minimum kapasitenin altına düşmüyorsa silinebilir.
- Örnek: 90 silindi

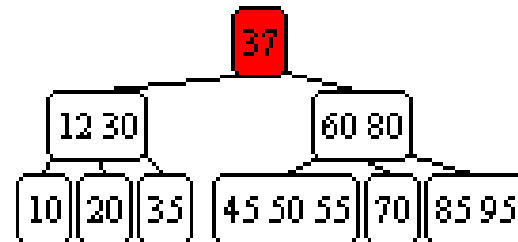
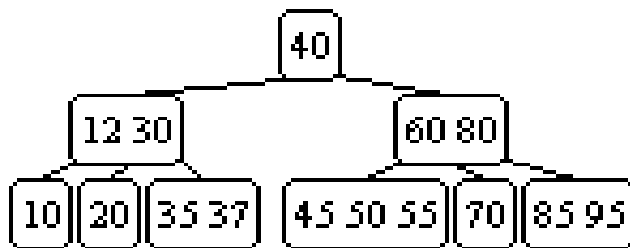


- **Kural2-** Silinen düğüm yaprak ve minimum kapasitenin altına düşüyor ise (öncelik sol) ebeveyn ile çocuklar yeniden düzenlenir (Soldaki en büyük veya sağdaki en küçük düğüm alınır).
- 25 silindi

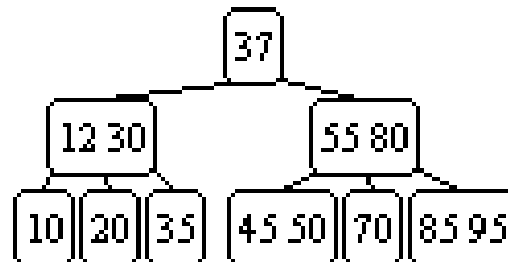


## 2-3-4 tree

- **Kural 3-** Silinen değer kök ve çocukları tek değil ise, soldaki en büyük veya sağdaki en küçük düğüm kök olur. (soldaki değer alındı, çocukları tek ise sonraki kurallar işletilir.)
- Örnek: 40 silindi

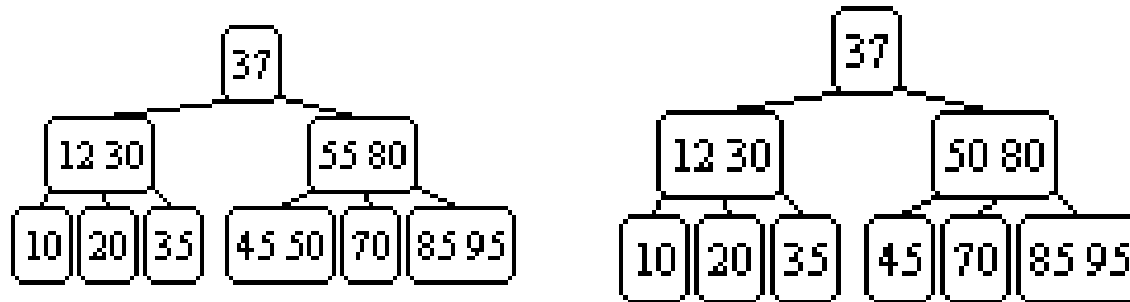


- **Kural 4-** Silinen düğüm çocukları olan bir düğüm ise çocukları fazla olandan değer alınır.
- 60 silindi

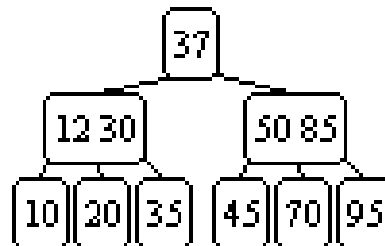


## 2-3-4 tree

- Örnek: 55 silindi (Kural-4)



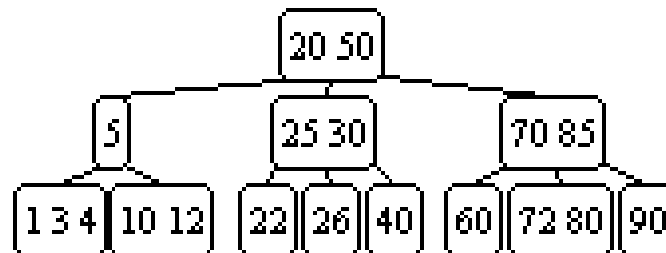
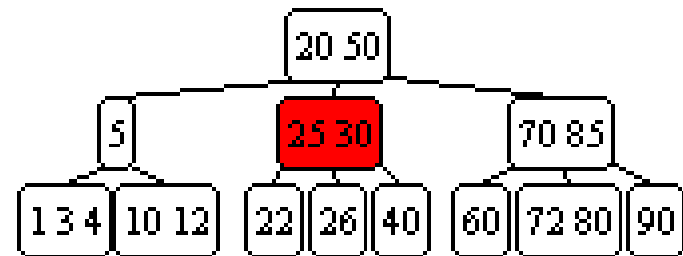
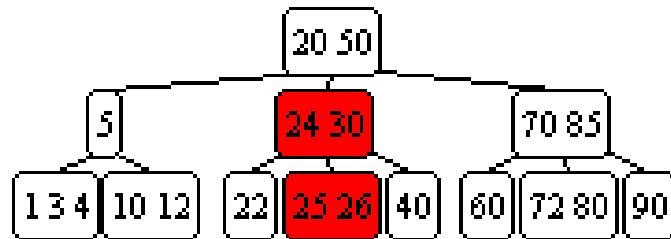
- 80 silindi (Kural-4)



## 2-3-4 tree

Soldaki en büyük düğüme göre

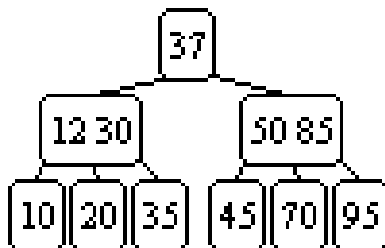
- Örnek: 24 nolu düğüm sil (Öncelik fazla olan çocuktur)



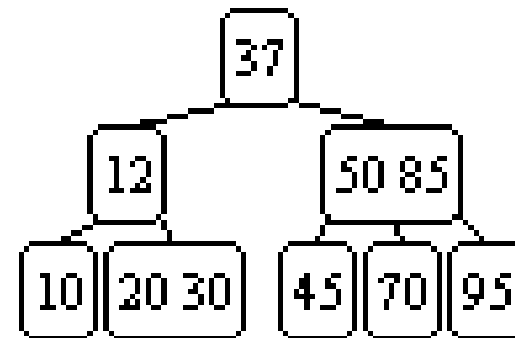
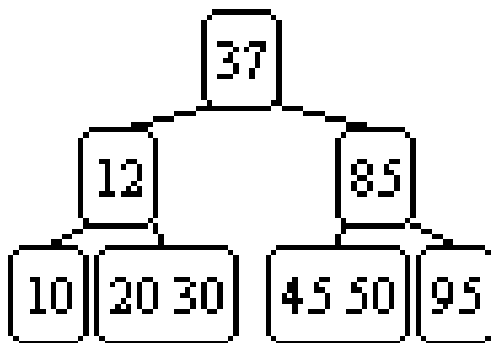


## 2-3-4 tree

- **Kural-5:** Minimum kapasitenin altına düşülürse ebeveyn ve çocuklar birleştirilir.
- Örnek: 35 silindi

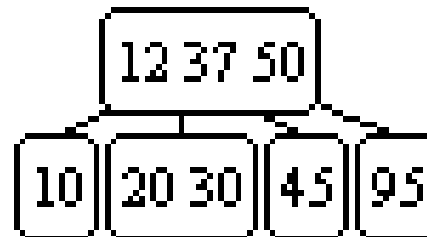
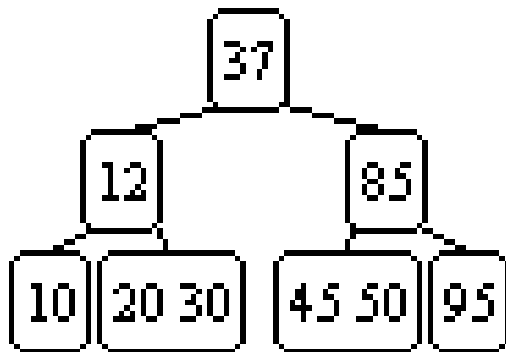


- Örnek: 70 silindi



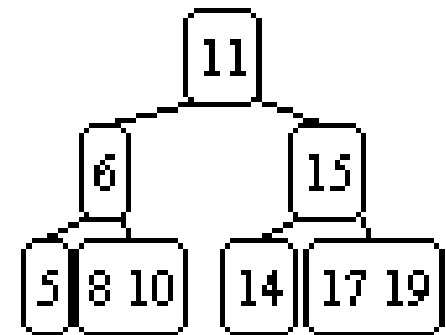
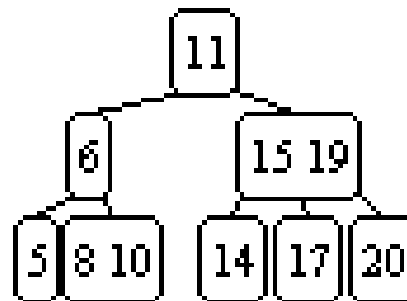
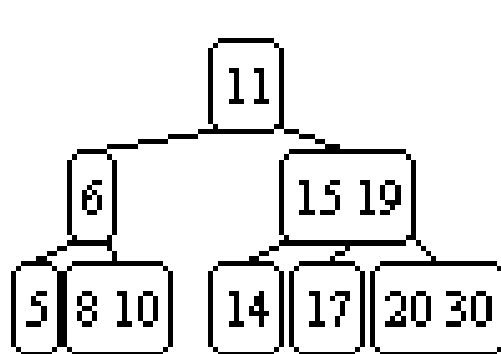
## 2-3-4 tree

- **Kural-6:** (B-tree den farkı) Çocukları olan bir düğüm silinmek istenirse; eğer silinen düğüm ve kardeşi tek değere sahip ise ebeveyn, kardeş ve çocuk düğümler birleştirilir (Döndürme).
- Örnek: 85 silindi, (Kural 4,5,6) Ebeveyn ve çocuklar birleştirildi.

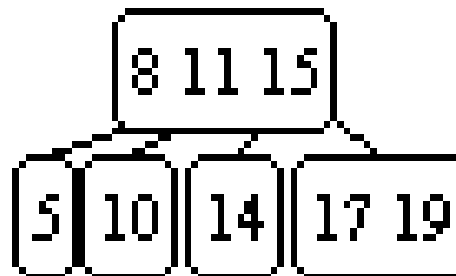


## 2-3-4 tree

- Örnek: 30,20 silindi

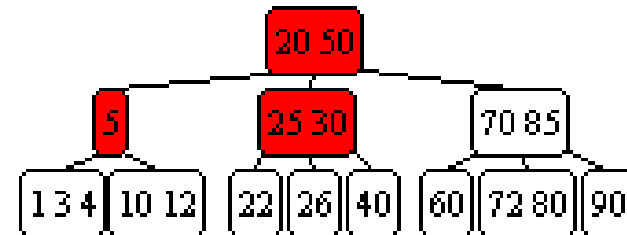
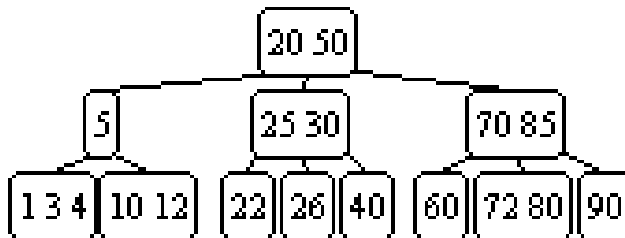


- 6 silindi



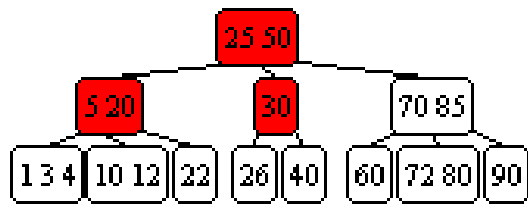
## 2-3- veya 2-3-4 tree

- **Kural-7:** (B-tree den farkı), Çocukları olan bir düğüm silinmek istenirse; Silinen düğüm tek sahip ama kardeşi tek değer değil ise ebeveyn, kardeş düğümler döndürülür ve çocuk düğüm ile birleştirilir.(Döndür ve birleştir.)
- Örnek: 5 nolu düğüm sil.
- 1-Ebeveyn ve kardeş düğümler seçildi

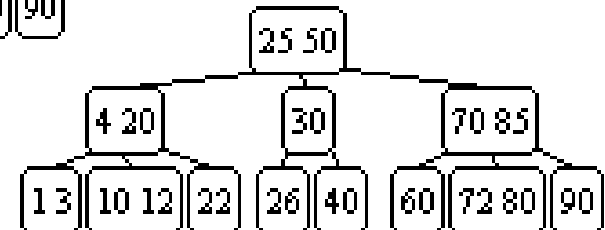
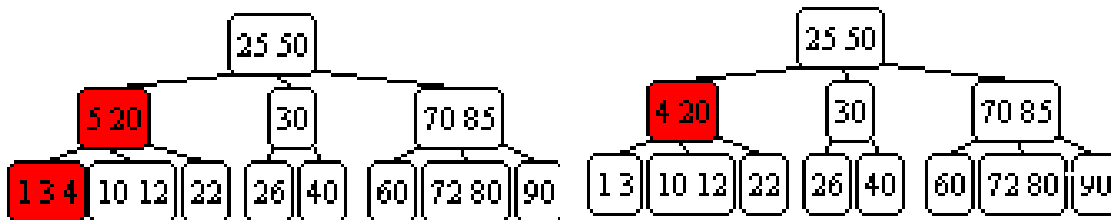


## 2-3- veya 2-3-4 tree

- 2-Döndürme işlemi gerçekleştirilerek silinecek düğüme değeri getirildi.



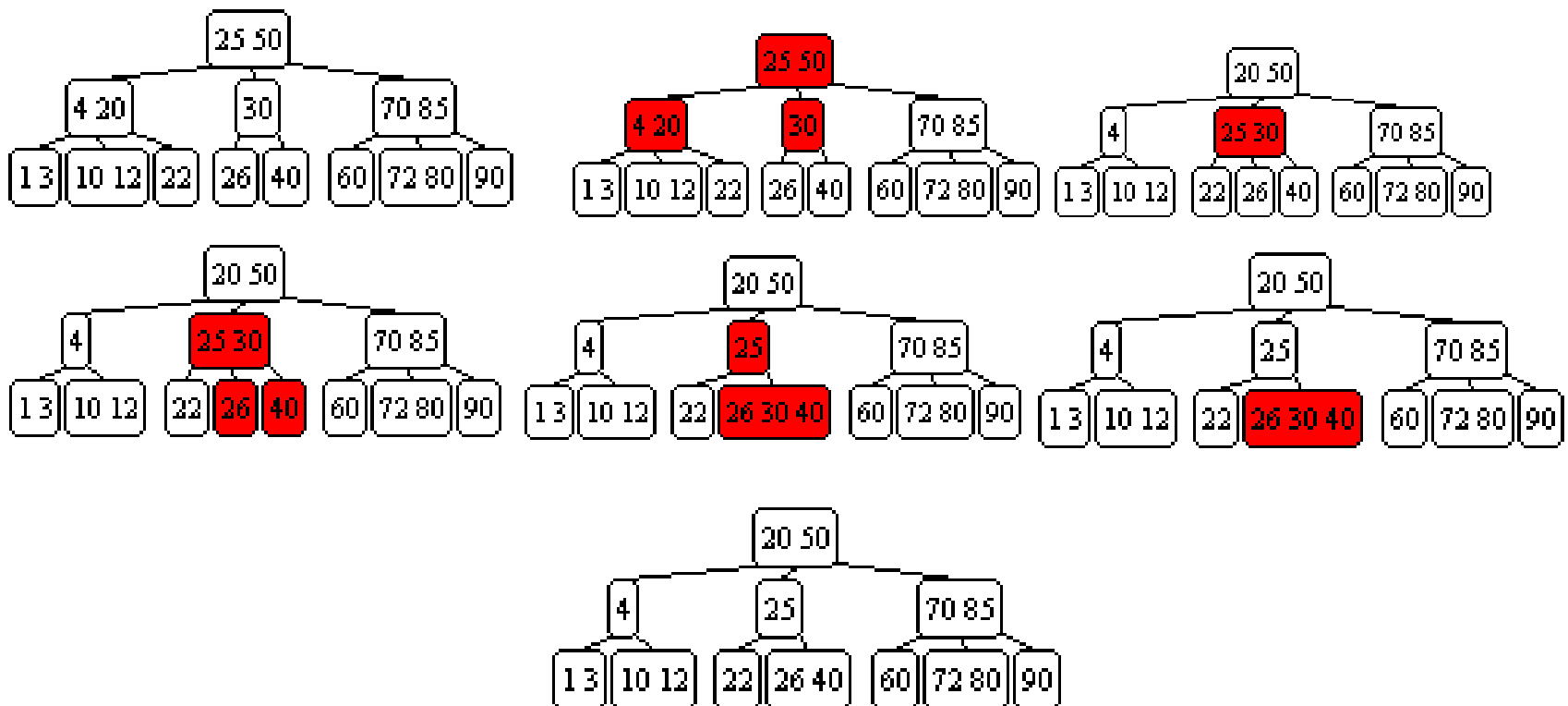
- 3- Çocuklardan düğüm getirildi ve 5 silindi.



## 2-3- veya 2-3-4 tree

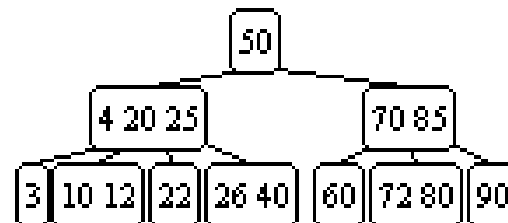
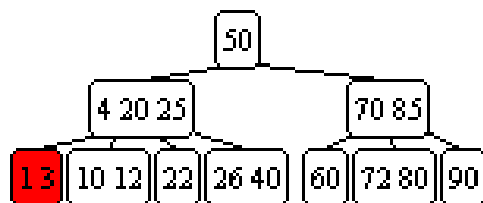
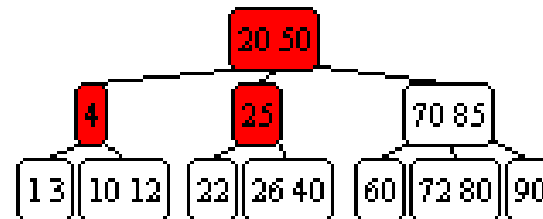
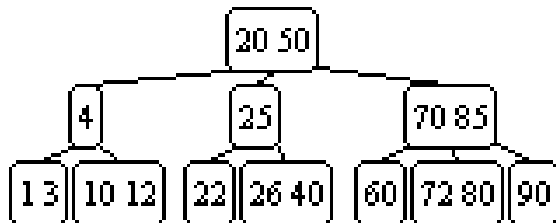
Soldaki en büyük düğüme göre

- Örnek: 30 nolu düğüm sil



## 2-3- veya 2-3-4 tree

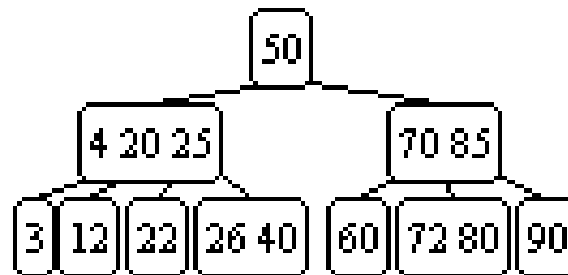
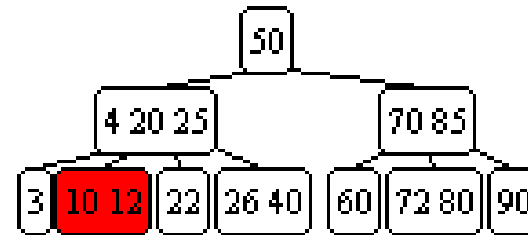
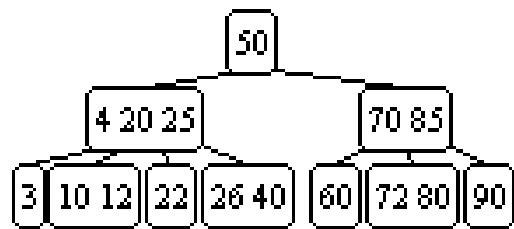
- **Kural-8:** B-tree den farkı, Silinen yaprak düğümün atası ve onun kardeşi tek değere sahip ise ebeveyn, kardeş ve çocuk düğümler birleştirilir.(Döndür ve birleştir.)(Kural-6 ya benzer)
- Örnek: 1 nolu düğüm sil (Kural 5,6,7)



## 2-3- veya 2-3-4 tree

Soldaki en büyük düğüme göre

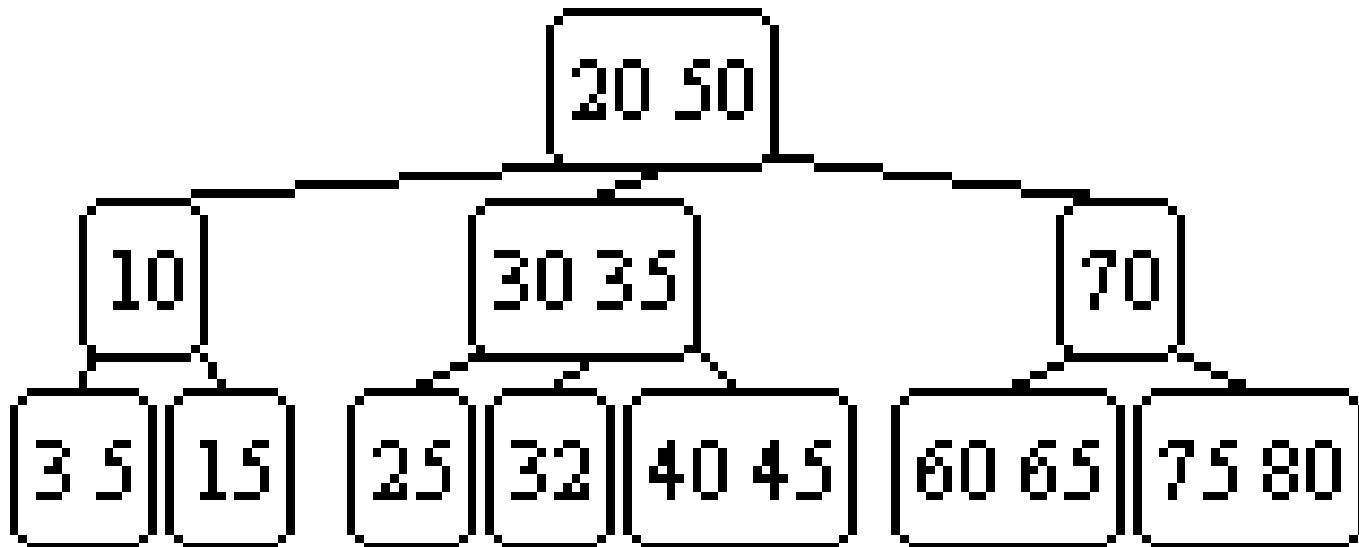
- Örnek: 10 nolu düğüm sil





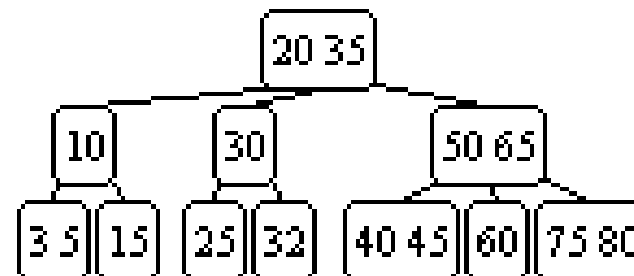
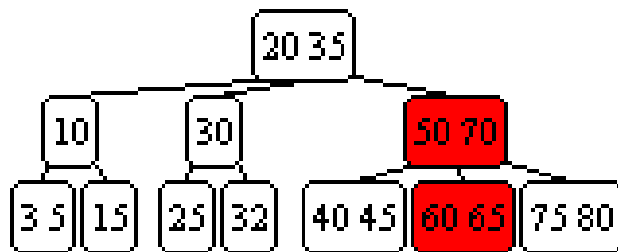
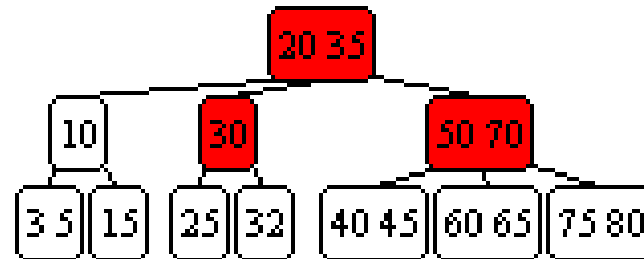
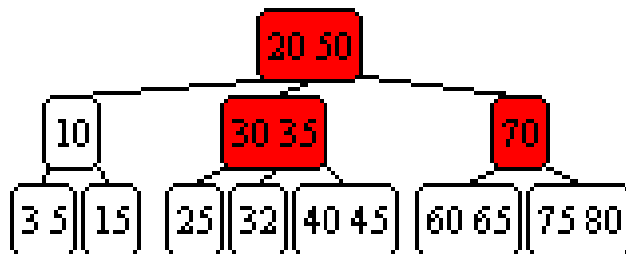
## 2-3- veya 2-3-4 tree

- Örnek: 70,30, 20,40,65,75,35 nolu düğümleri siliniz



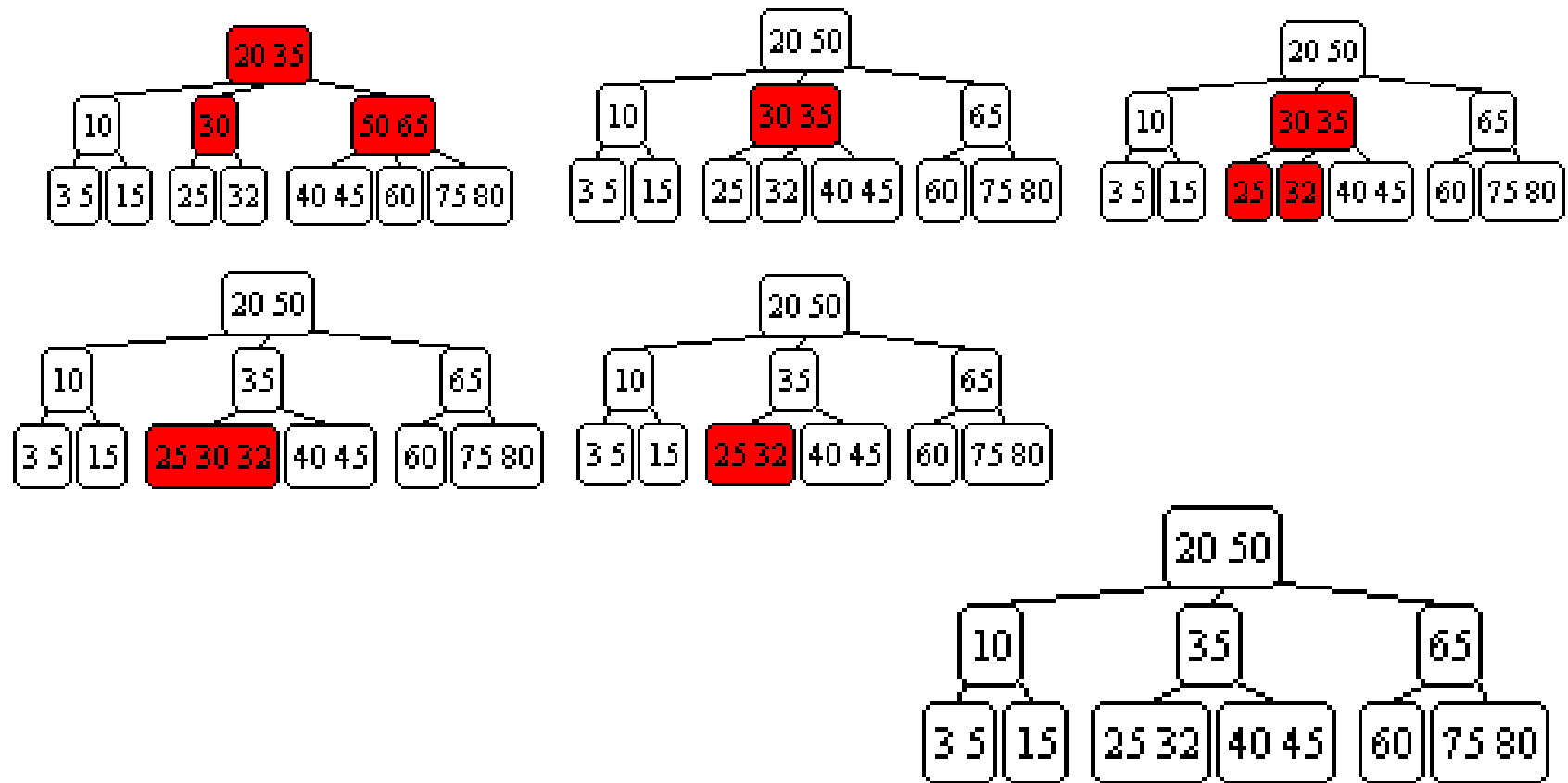
## 2-3- veya 2-3-4 tree

- Örnek: 70 (Kural 7)



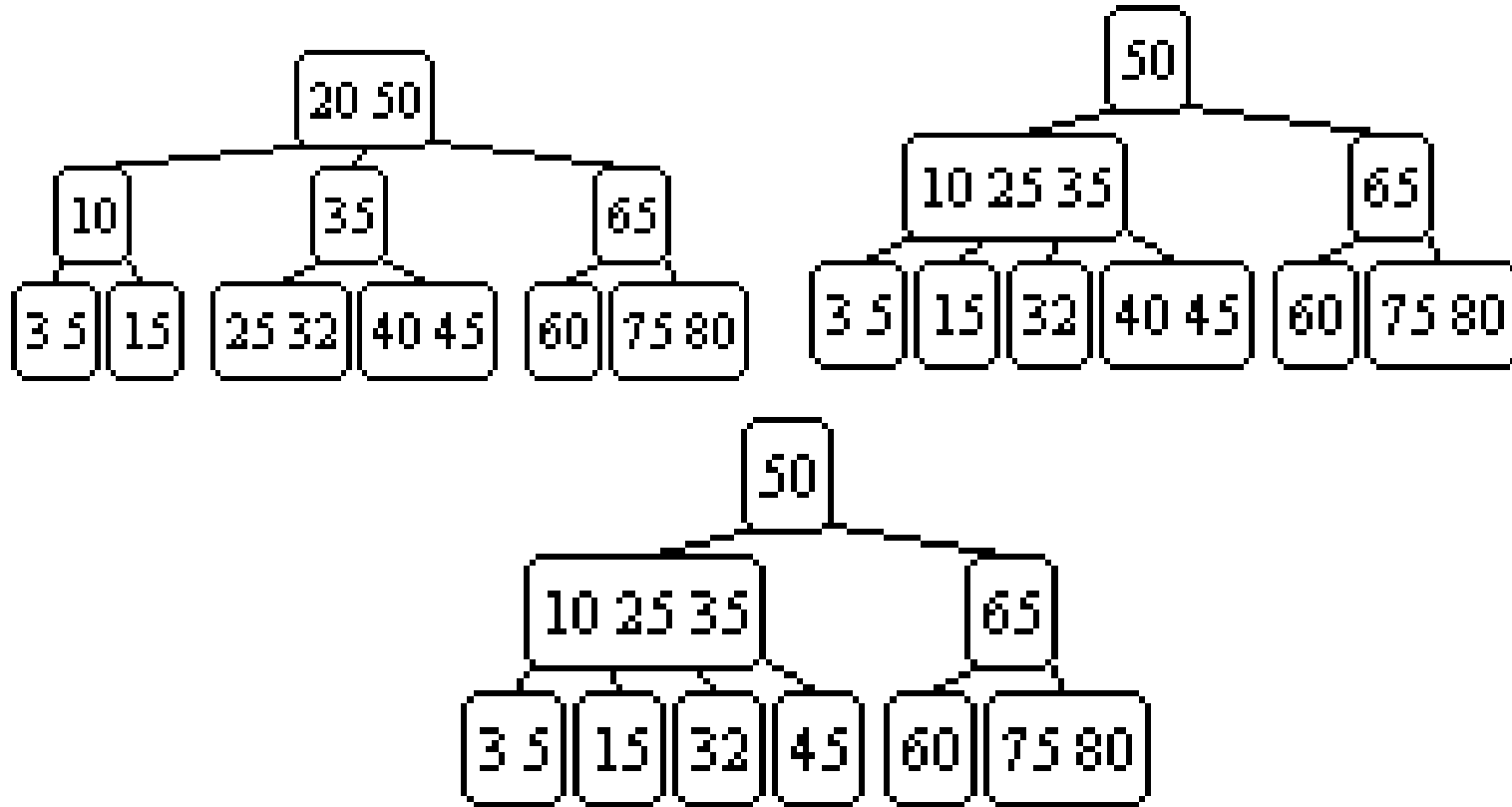
## 2-3- veya 2-3-4 tree

- Örnek: 30 (Kural 7)



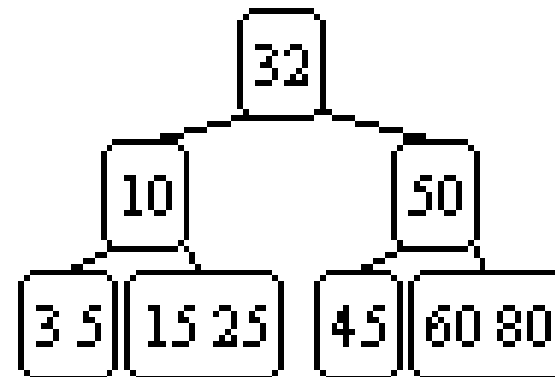
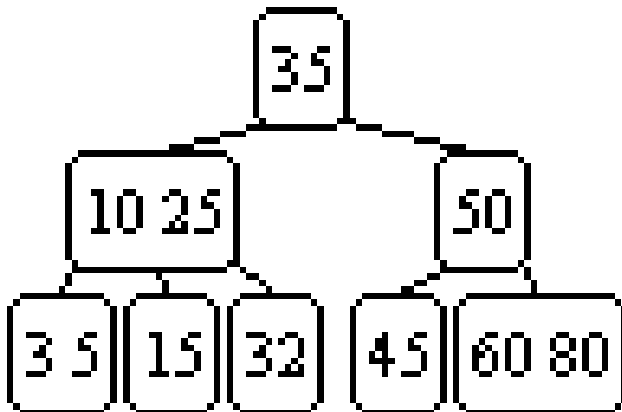
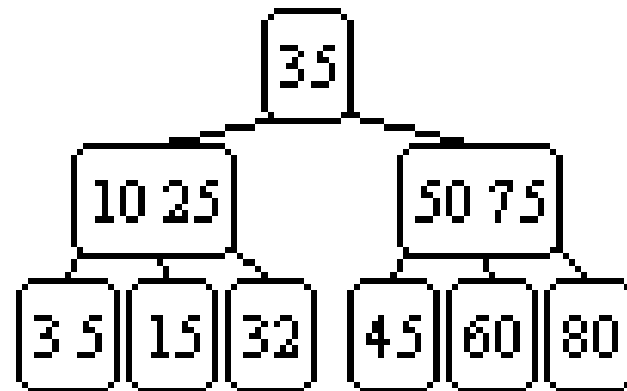
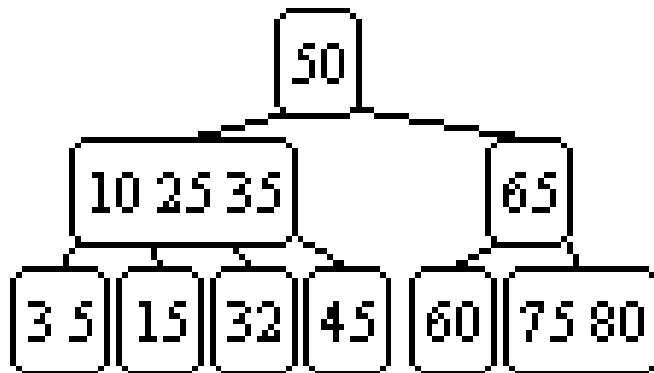
## 2-3- veya 2-3-4 tree

- Örnek: 20 (Kural 4-7), 40 (Kural 1)



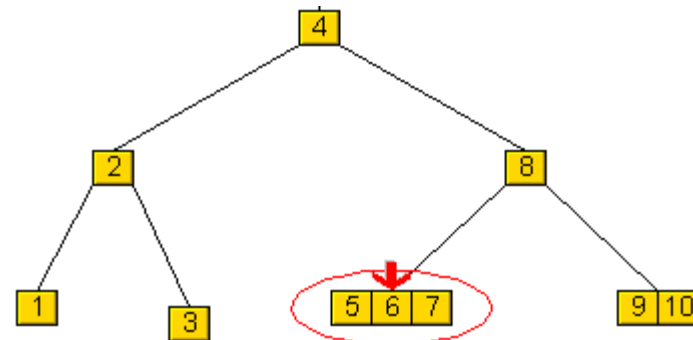
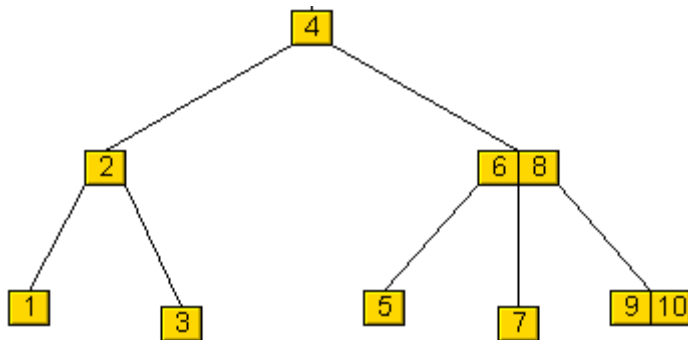
## 2-3- veya 2-3-4 tree

- Örnek: 65 (Kural 7), 75 (Kural 5), 35 (Kural 4)

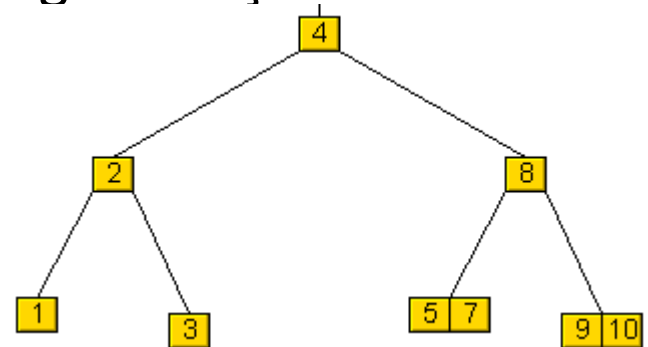


## 2-3- veya 2-3-4 tree

- Örnek: Silme, 6 silindi

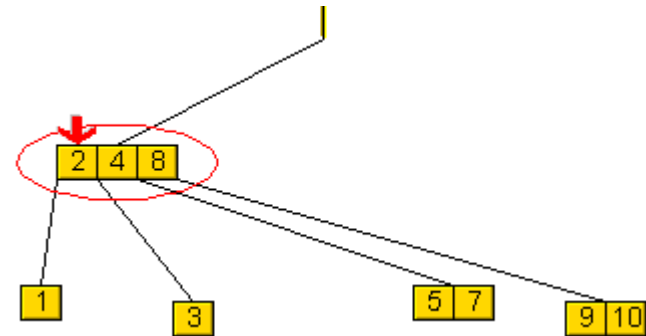
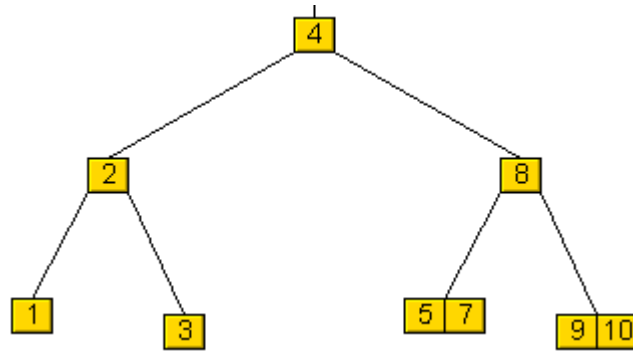


- Silme işlemi gerçekleşmeden 6 değeri sol çocuklar ile birleştiriliyor. Daha sonra silin

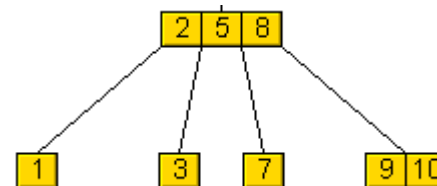
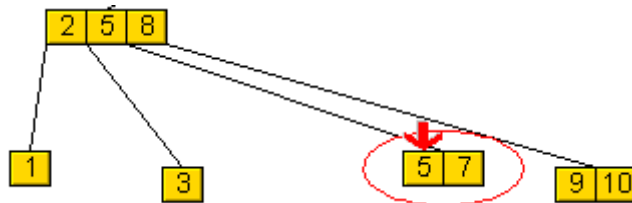


## 2-3- veya 2-3-4 tree

- Örnek: 4 silindi



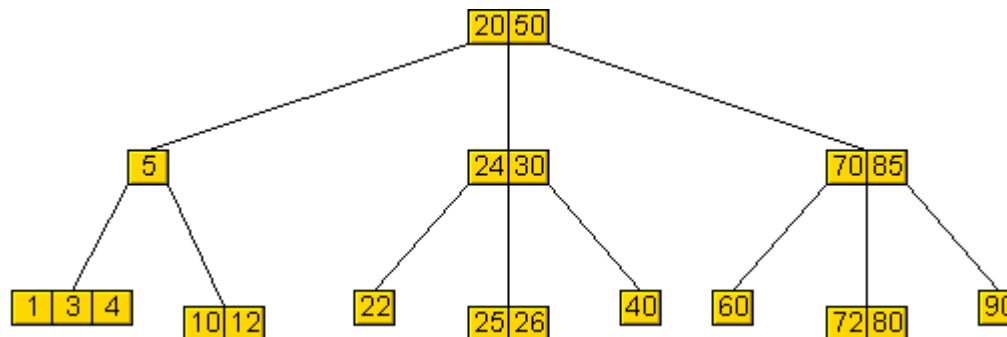
- 4 çocuk olabilmesi için çocuktan da değer alındı.



## 2-3- veya 2-3-4 tree

- Örnek:

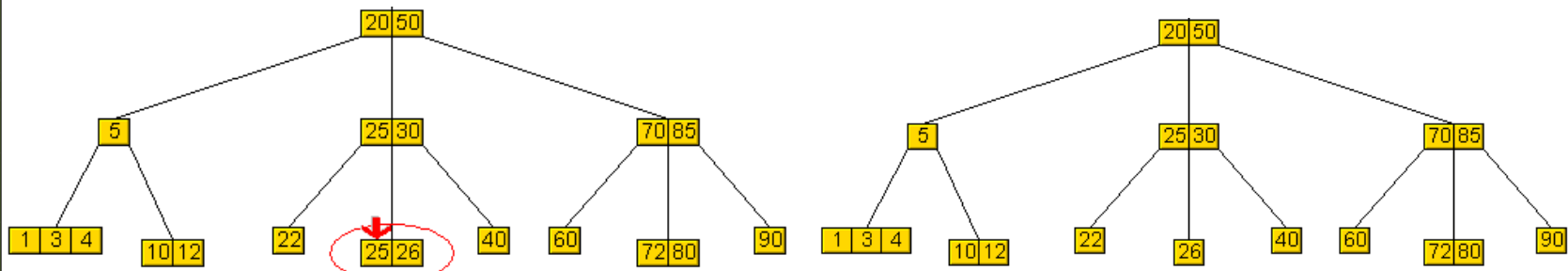
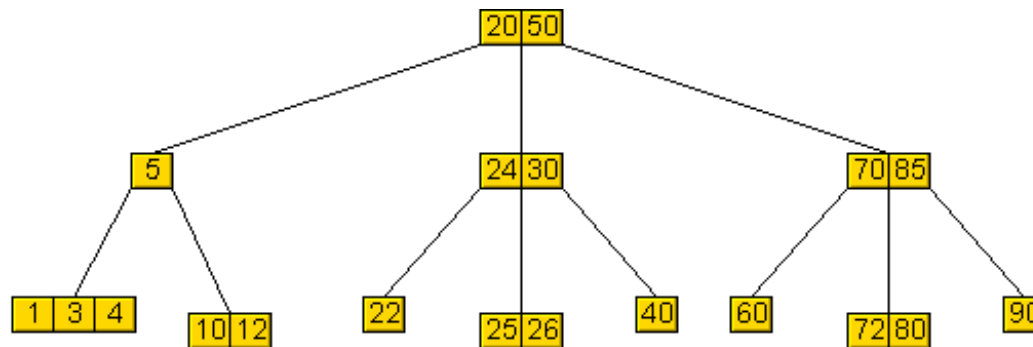
50,60,20,30,10,5,90,25,40,70,80,22,85,72,3,1,4,  
12,24,26 ağacı oluşturunuz.





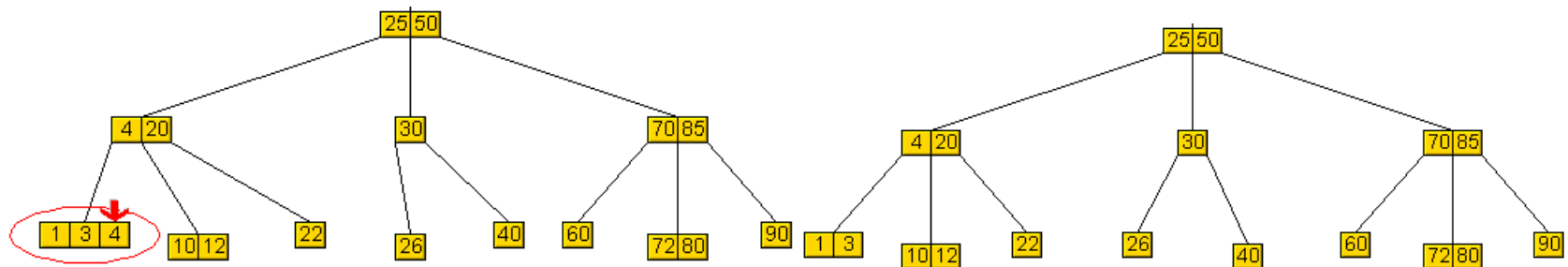
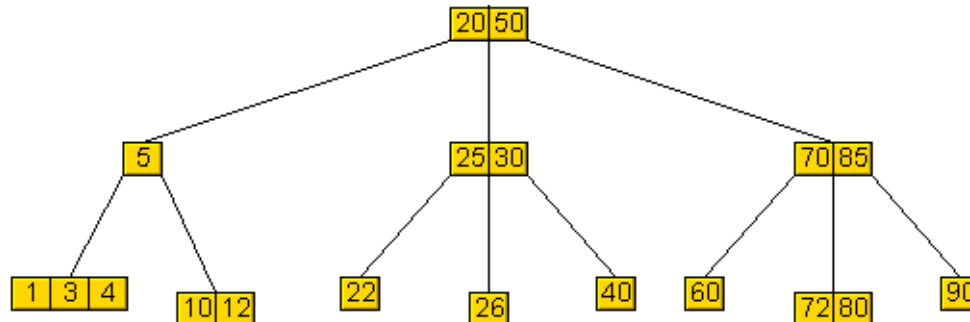
## 2-3- veya 2-3-4 tree

- Örnek: 24 nolu düğüm sil



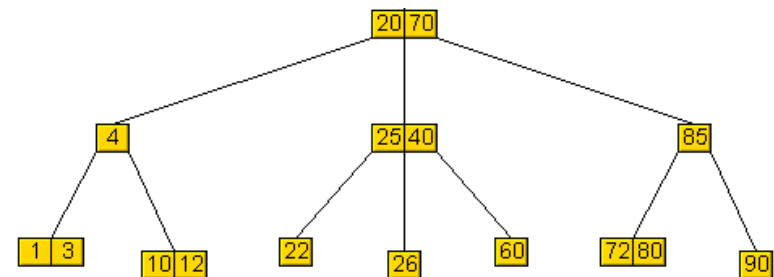
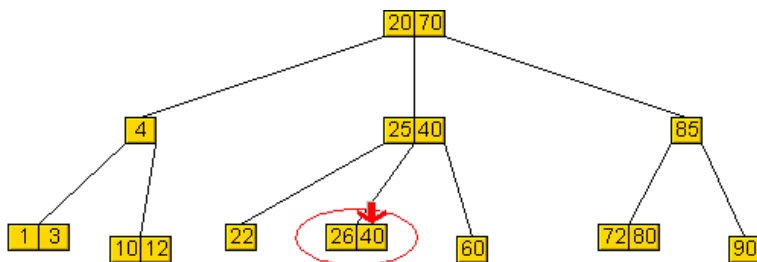
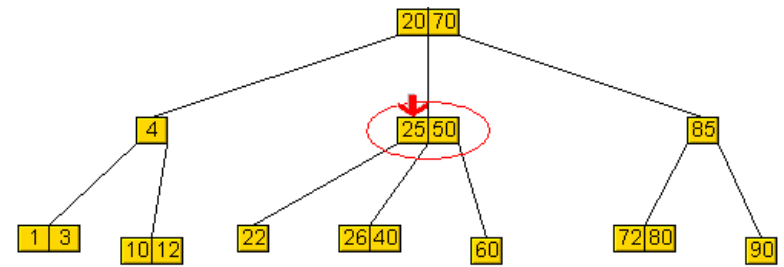
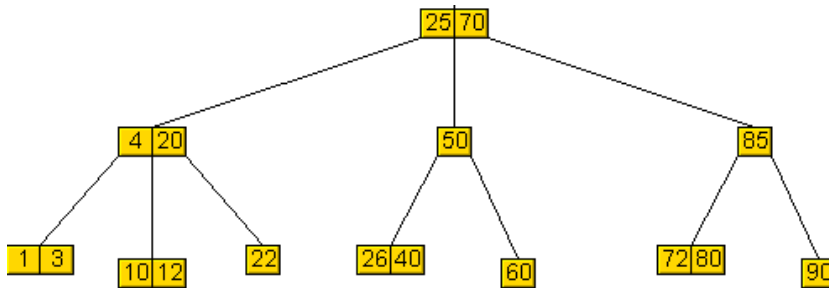
## 2-3- veya 2-3-4 tree

- Örnek: 5 nolu düğüm sil



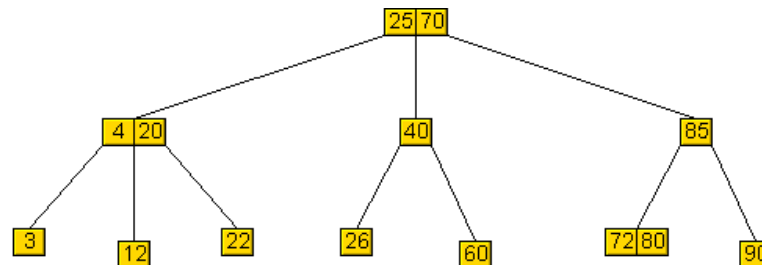
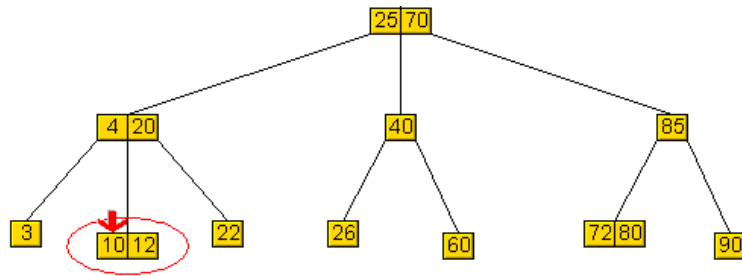
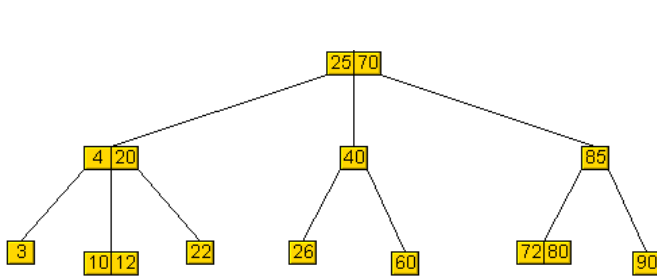
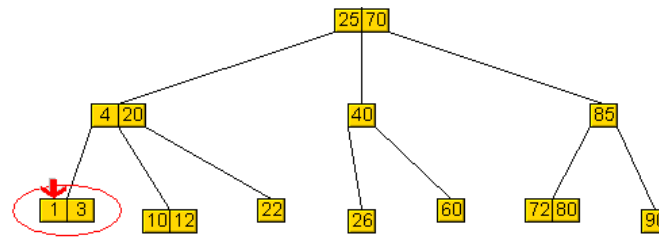
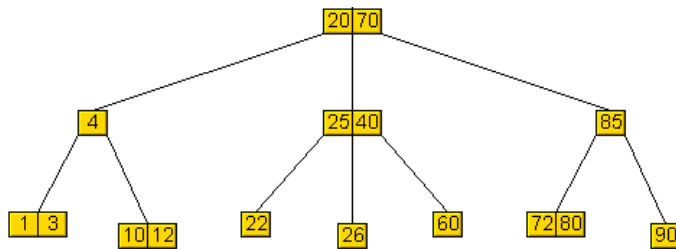
## 2-3- veya 2-3-4 tree

- Örnek: 50 nolu düğüm sil



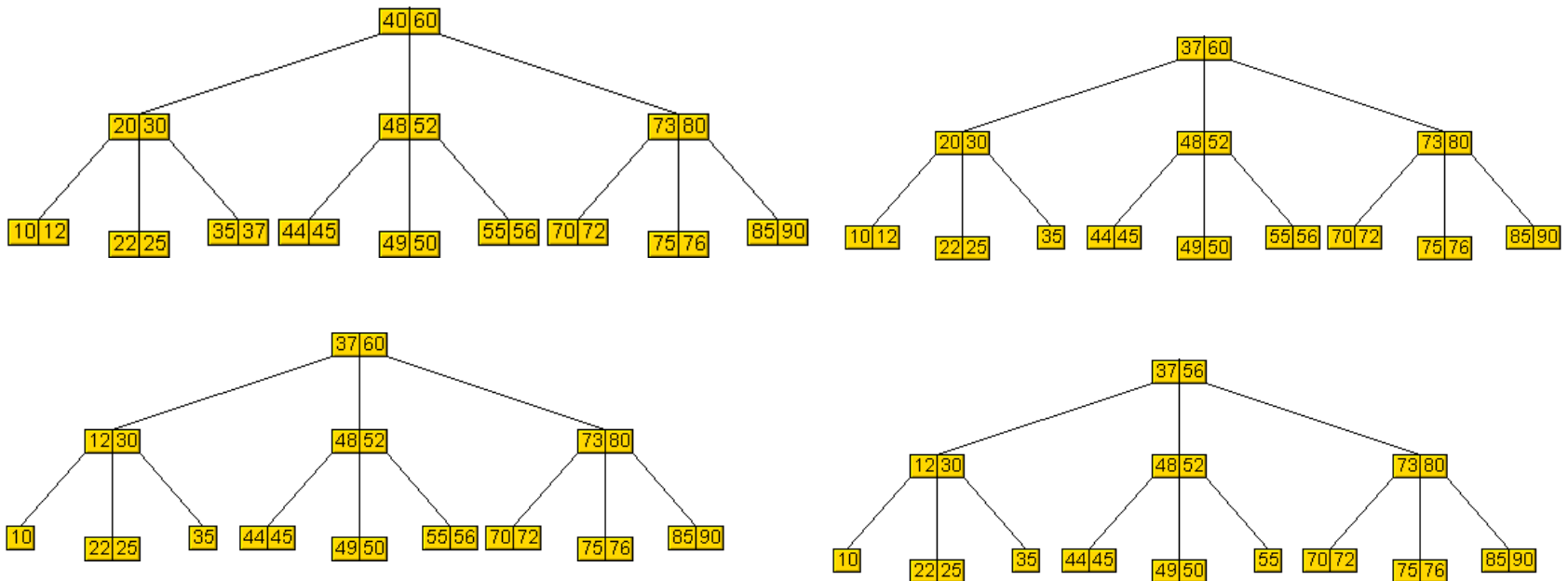
## 2-3- veya 2-3-4 tree

- Örnek: 1,10 nolu düğüm sil



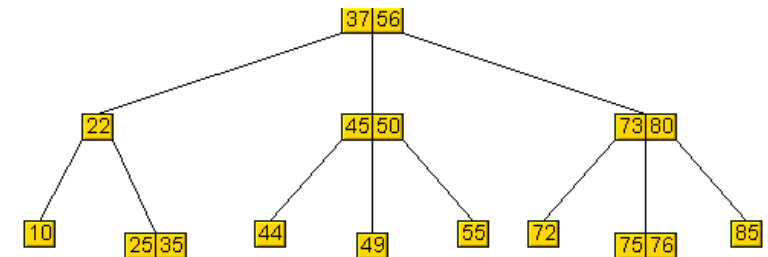
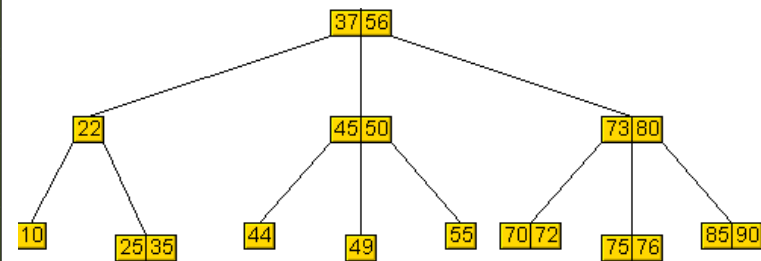
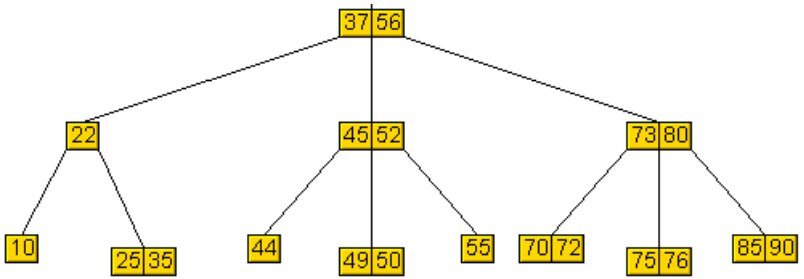
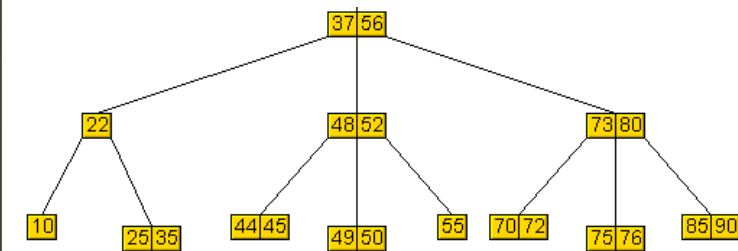
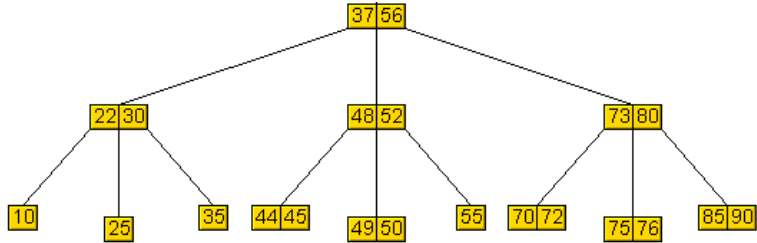
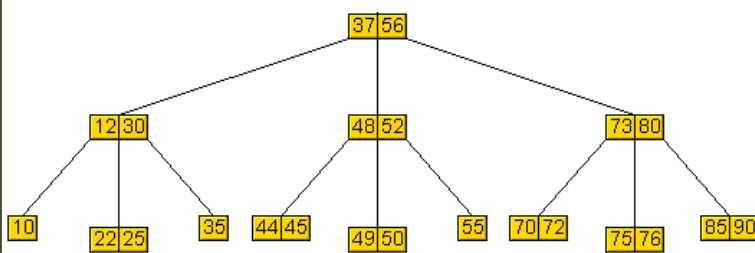
## 2-3- veya 2-3-4 tree

- Örnek: 40, 20,60 nolu düğüm sil



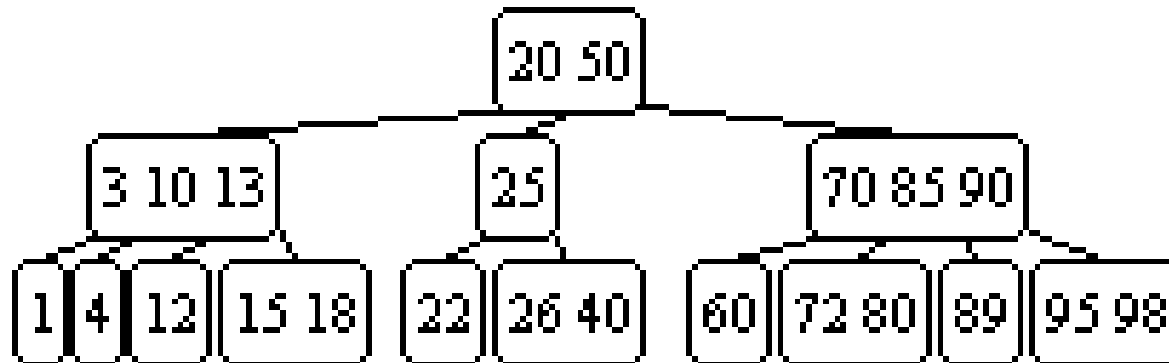
## 2-3- veya 2-3-4 tree

- Örnek: 12,30,48,52,70,90 nolu düğüm sil



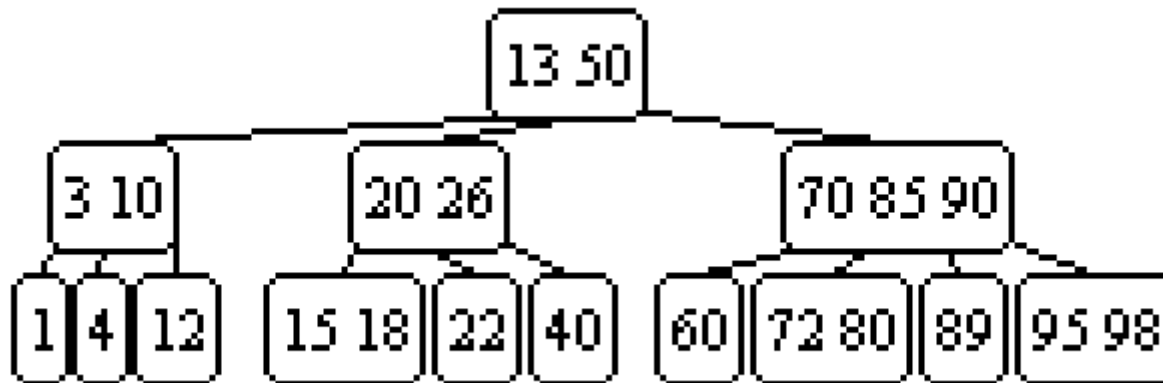
## 2-3- veya 2-3-4 tree

- Örnek: 25 değerini siliniz



## 2-3- veya 2-3-4 tree

- Örnek: 25 değerini siliniz





## 2-3-4 tree java code

- `/* Tree234.java */-`  
<http://www.lensovet.net/~sysadmin/w/CS/61b/Homework/hw7/dict/Tree234.java>
- `package dict;`
- `/**`
- `* A Tree234 implements an ordered integer dictionary ADT using a 2-3-4 tree.`
- `* Only int keys are stored; no object is associated with each key. Duplicate`
- `* keys are not stored in the tree.`
- `* * @author Jonathan Shewchuk, lensovet`
- `**/`