BLM267

Bölüm 11: Çok Yönlü Arama Ağaçları

C Kullanarak Veri Yapıları, İkinci Baskı

C Kullanarak Veri Yapıları, İkinci Baskı
Reema Thareja

- giriiş
- B Ağaçları
- B+ Ağaçları
- 2-3 Ağaç

giriiş

- İkili arama ağacındaki her düğümün bir değer ve sırasıyla düğümün sol ve sağ alt ağaçlarına işaret eden sol ve sağ olmak üzere iki işaretçi içerdiğini tartıştık.
- İkili arama ağacı düğümünün yapısı Şekil 11.1'de gösterilmiştir.
- Aynı kavram, düğüm başına M – 1 değer ve M alt ağacı olan M yönlü bir arama ağacında da kullanılır.
- Böyle bir ağaçta M, ağacın derecesi olarak adlandırılır. İkili arama ağacında M = 2 olduğunu ve bu nedenle bir değere ve iki alt ağaca sahip olduğunu unutmayın.
- Başka bir deyişle, M yönlü arama ağacının her bir dahili düğümü, M alt ağaca ait işaretçilerden oluşur ve M > 2 olmak üzere M – 1 anahtar içerir.
- M-yollu arama ağacı düğümünün yapısı Şekil 11.2'de gösterilmiştir.

giriiş

Pointer to left sub-tree	Value or Key of the node	Pointer to right sub-tree

Figure 11.1 Structure of a binary search tree node

P_0	K_0	P_1	K_1	P_2	K_2	......	P_{n-1}	K_{n-1}	P_n

Figure 11.2 Structure of an M-way search tree node

- **Gösterilen yapıda, P_0, P_1, P_2, ..., Pn işaretçilerdir Düğümün alt ağaçları ve K_0, K_1, K_2, ..., Kn–1 düğümün anahtar değerleridir.**
- **Tüm anahtar değerleri artan sırada saklanır. Yani, $0 \leq i \leq n-2$ için $Ki < Ki+1$.**

giriiş

- M yönlü bir arama ağacında, her düğümün tam olarak M–1 değere ve M alt ağaca sahip olması zorunlu değildir.
- Bunun yerine, düğüm 1 ile M–1 arasında herhangi bir değere sahip olabilir ve alt ağaçların sayısı 0'dan (bir yaprak düğüm için) i + 1'e kadar değişebilir; burada i, düğümdeki anahtar değerlerinin sayısıdır.
- Dolayısıyla M, düğümde kaç adet anahtar değerinin saklanabileceğini tanımlayan sabit bir üst sınırdır.
- Şekil 11.3'te gösterilen M-yollu arama ağacını ele alalım. Burada M = 3.
- Yani bir düğüm en fazla iki anahtar değerini depolayabilir ve üç alt ağaca işaretçiler içerebilir.

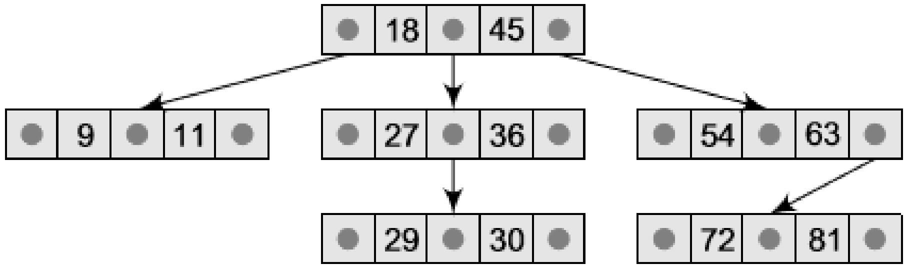


Figure 11.3 M-way search tree of order 3

giriiş

- Örneğimizde, kavramın okuyucu için daha kolay olması amacıyla M için çok küçük bir değer aldık, ancak pratikte M genellikle çok büyüktür.
- 3 yönlü bir arama ağacı kullanarak, M yönlü bir arama ağacının bazı temel özelliklerini ortaya koyalım.
  - P_0 tarafından işaret edilen alt ağaçtaki anahtar değerlerinin K_0 anahtar değerinden daha az olduğunu unutmayın. Benzer şekilde, P_1 tarafından işaret edilen alt ağaçtaki tüm anahtar değerleri K_1'den daha azdır, vb. ve benzeri. Dolayısıyla, genelleştirilmiş kural, Pi tarafından işaret edilen alt ağaçtaki tüm anahtar değerlerinin Ki'den daha az olmasıdır, burada $0 \leq i \leq n-1$.
  - P_1 tarafından işaret edilen alt ağaçtaki anahtar değerlerinin K_0 anahtar değerinden büyük olduğuna dikkat edin. Benzer şekilde, P_2 tarafından işaret edilen alt ağaçtaki tüm anahtar değerleri K_1'den büyüktür, vb. Böylece, genelleştirilmiş kural, Pi tarafından işaret edilen alt ağaçtaki tüm anahtar değerlerinin Ki–1'den büyük olmasıdır, burada $0 \leq i \leq n-1$. M yönlü bir arama ağacında, her alt ağaç aynı zamanda bir M yönlü arama

B Ağaçları

- B ağacı, Rudolf Bayer ve Ed McCreight tarafından 1970 yılında geliştirilen, disk erişimi için yaygın olarak kullanılan özel bir M yönlü ağaçtır.
- m sırasındaki bir B ağacının en fazla m–1 anahtarı ve alt ağaçlarına ait m işaretçisi olabilir.
- Bir B ağacı çok sayıda anahtar değer ve alt ağaçlara işaret eden işaretçiler içerebilir.
- Çok sayıda anahtarın tek bir düğümde depolanması ağacın yüksekliğini nispeten küçük tutar.
- Sıralanmış verileri depolamak için tasarlanan B ağacı, arama, ekleme ve silme işlemlerinin logaritmik amortize sürede gerçekleştirilmesine olanak tanır.
- m derecesindeki (her düğümün sahip olabileceği maksimum çocuk sayısı) bir B ağacı, M yönlü arama ağacının tüm özelliklerine sahip bir ağaçtır.
- Ayrıca aşağıdaki özelliklere sahiptir:
  - 1. B ağacındaki her düğümün en fazla (maksimum) m çocuğu vardır.
  - 2. B ağacındaki kök düğüm ve yaprak düğümleri hariç her düğümün en az (minimum) m/2 çocuğu vardır. Bu durum ağacın gür kalmasına yardımcı olur, böylece kök düğümden yaprağa giden yol çok fazla veri depolayan bir ağaçta bile çok kısa olur.
  - 3. Kök düğüm, terminal (yaprak) düğüm değilse en az iki çocuğa sahiptir.
  - 4. Tüm yaprak düğümleri aynı seviyededir.

B Ağaçları

- B ağacındaki bir iç düğümün n sayıda çocuğu olabilir, burada 0 ≤ n ≤ m.
- Her düğümün aynı sayıda çocuğa sahip olması gerekmez, ancak tek kısıtlama düğümün en az m/2 çocuğa sahip olmasıdır.
- Şekil 11.4'te 4. dereceden bir B ağacı verilmiştir.
- B ağacında ekleme ve silme işlemleri yapılırken alt düğüm sayısı değişebilir.
- Bu nedenle, minimum sayıda çocuğu korumak için iç düğümler birleştirilebilir veya bölünebilir.
- Bu bölümde arama, ekleme ve silme işlemlerini ele alacağız.

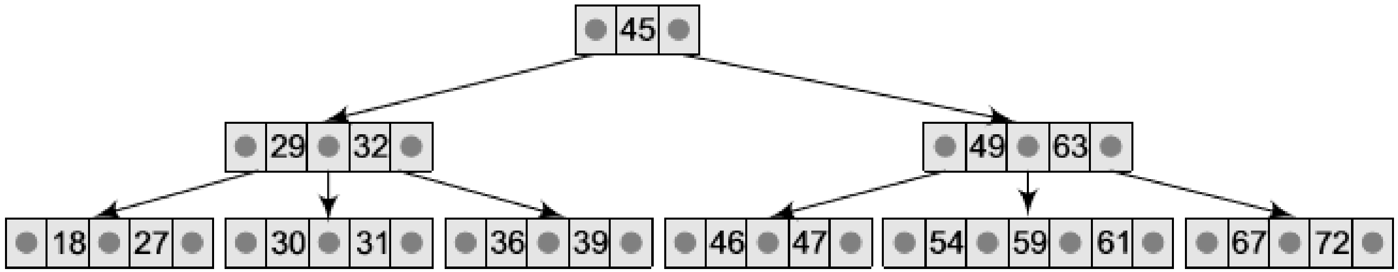


Figure 11.4 B tree of order 4

B Ağaçları

- B Ağacında bir öğe arama
- B ağacında bir öğe aramak ikili arama ağaçlarındakine benzerdir. Şekil 11.4'te verilen B ağacını düşünün.
- 59'u aramak için kök düğümden başlıyoruz.
- Kök düğümün değeri 59'dan küçük olan 45'tir. Dolayısıyla sağ alt ağaçta dolaşıyoruz.
- Kök düğümün sağ alt ağacının iki anahtar değeri vardır: 49 ve 63. 49 ≤ 59 ≤ 63 olduğundan, 49'un sağ alt ağacını, yani 63'ün sol alt ağacını geçeriz.
- Bu alt ağacın 54, 59 ve 61 olmak üzere üç değeri vardır.
- 59 değerini bulduğumuzda arama başarılı olur. Başka bir örnek alalım.
- Eğer 9'u aramak istiyorsanız, kök düğümün sol alt ağacını dolaşmalıyız.
- Sol alt ağacın iki anahtar değeri vardır, 29 ve 32. Tekrar 29'un sol alt ağacını dolaşıyoruz.
- Bunun 18 ve 27 olmak üzere iki anahtar değeri olduğunu görüyoruz. 18'in sol alt ağacı yok, dolayısıyla ağaçta 9 değeri saklanmıyor.
- Arama işleminin çalışma süresi ağacın yüksekliğine bağlı olduğundan, B ağacında bir elemanı arayan algoritmanın yürütülmesi O(logt n) zaman alır.

B Ağaçları

- **B Ağacına Yeni Bir Eleman Ekleme**
- **B ağacında tüm eklemeler yaprak düğüm düzeyinde yapılır.**
- **Aşağıda verilen algoritma kullanılarak B ağacına yeni bir değer eklenir.**
  - **1. Yeni anahtar değerinin ekleneceği yaprak düğümünü bulmak için B ağacını arayın.**
  - **2. Yaprak düğümü dolu değilse, yani m–1'den az anahtar değeri içeriyorsa, düğümün elemanlarını sıralı tutarak yeni elemanı düğüme ekle.**
  - **3. Yaprak düğümü doluysa, yani yaprak düğümü zaten m–1 anahtar değeri içeriyorsa, o zaman**
    - **(a) yeni değeri mevcut anahtar kümesine sırayla ekleyin,**
    - **(b) düğümü medyanından iki düğüme bölün (bölünen düğümlerin yarı dolu olduğuna dikkat edin) ve**
    - **(c) medyan elemanını ebeveyninin düğümüne kadar itin. Ebeveynin düğümü zaten doluysa, aynı adımları izleyerek ebeveyn düğümünü bölün.**

B Ağaçları

Example 11.1 Look at the B tree of order 5 given below and insert 8, 9, 39, and 4 into it.

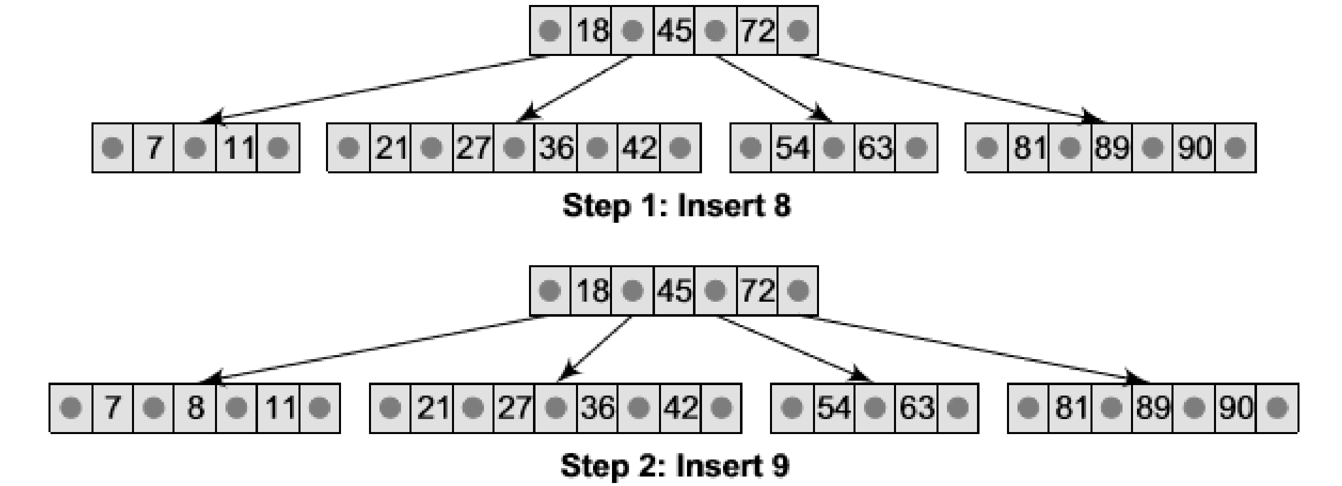


Figure 11.5(a)

B Ağaçları

- Şu ana kadar yaprak düğümleri dolu olmadığı için 8 ve 9'u ağaca kolayca yerleştirdik.
- Ancak şimdi 39'un eklenmesi gereken düğüm dört değer içerdiğinden zaten dolu.
- Burada düğümleri bölerek iki ayrı düğüm oluşturuyoruz.
- Ancak bölmeden önce anahtar değerlerini (yeni değer dahil) sıraya koyun.
- Sıralı değer kümesi 21, 27, 36, 39 ve 42 olarak verilmiştir.
- Ortanca değer 36 olduğundan, 36'yı ana düğümüne itin ve yaprak düğümleri bölün.

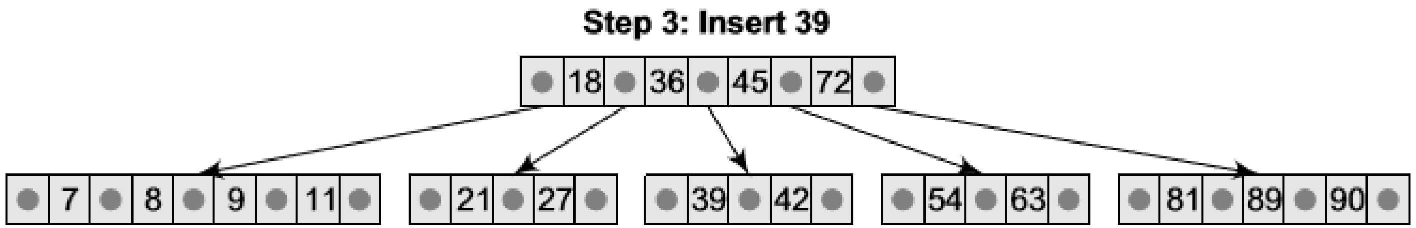


Figure 11.5(b)

B Ağaçları

- Şimdi 4'ün ekleneceği düğüm zaten dolu çünkü dört anahtar değeri içeriyor.
- Burada düğümleri bölerek iki ayrı düğüm oluşturuyoruz.
- Ancak bölmeden önce anahtar değerlerini (yeni değer de dahil olmak üzere) sıraya koyuyoruz.
- Sıralı değer kümesi 4, 7, 8, 9 ve 11 olarak verilmiştir.
- Ortanca değer 8 olduğundan, 8'i ana düğümüne itiyoruz ve yaprak düğümleri bölüyoruz.
- Ancak yine ebeveyn düğümünün dolu olduğunu görüyoruz, bu nedenle aynı prosedürü kullanarak ebeveyn düğümünü bölüyoruz.

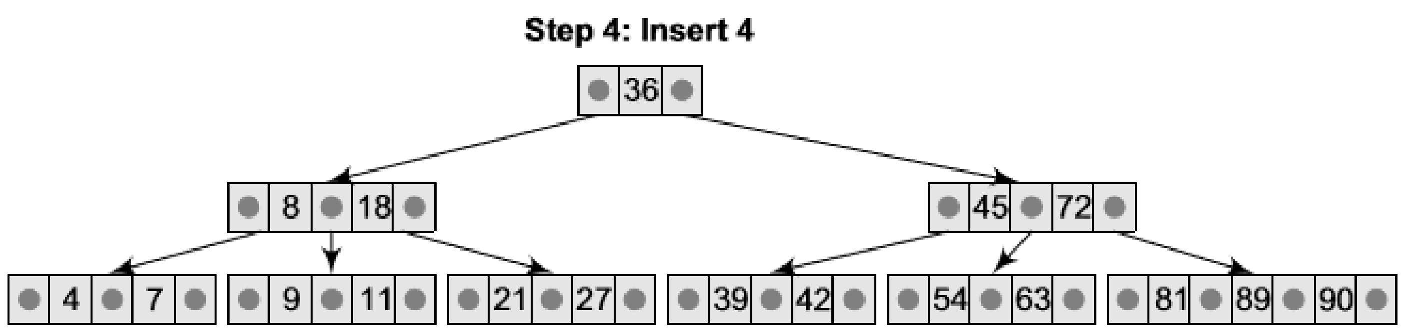


Figure 11.5(c) B tree

B Ağaçları

- Ekleme işlemi gibi silme işlemi de yaprak düğümlerinden yapılır.
- Silme işleminin iki durumu vardır. İlk durumda, bir yaprak düğümünün silinmesi gerekir.
- İkinci durumda, dahili bir düğüm silinmelidir. Öncelikle bir yaprak düğümü silmenin içerdiği adımlara bakalım.
- 1. Silinmesi gereken yaprak düğümünü bulun.
- 2. Yaprak düğümü minimum anahtar değer sayısından (m/2'den fazla eleman) daha fazlasını içeriyorsa, değeri silin.
- 3. Eğer yaprak düğümü m/2 eleman içermiyorsa, o zaman düğümü soldan veya sağ kardeşten bir eleman alarak doldur.
  - (a) Sol kardeş, minimum anahtar değerlerinden daha fazlasına sahipse, en büyük anahtarını ana düğümüne itin ve araya giren öğeyi ana düğümden anahtarın silindiği yaprak düğüme çekin.
  - (b) Aksi takdirde, sağ kardeş minimum anahtar değerlerinden daha fazlasına sahipse, en küçük anahtarını ana düğümüne itin ve ara öğeyi ana düğümden anahtarın silindiği yaprak düğüme çekin.

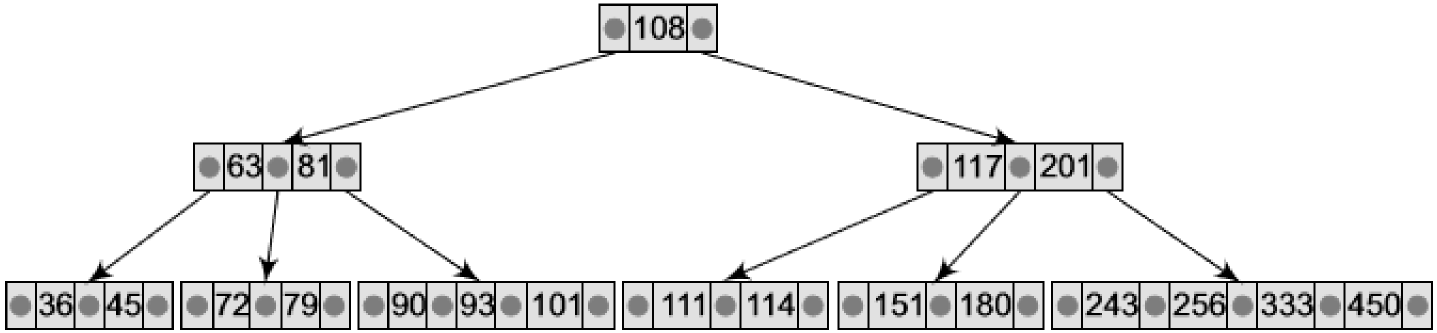
B Ağaçları

- 4. Aksi takdirde, hem sol hem de sağ kardeşler yalnızca minimum sayıda öğe içeriyorsa, iki yaprak düğümü ve ana düğümün araya giren öğesini birleştirerek yeni bir yaprak düğümü oluşturun (öğe sayısının bir düğümün sahip olabileceği maksimum öğe sayısını, yani m'yi aşmadığından emin olun). Araya giren öğeyi ana düğümden çekmek düğümde minimum anahtar sayısından daha az sayıda anahtar bırakıyorsa, işlemi yukarı doğru yayarak B ağacının yüksekliğini azaltın.

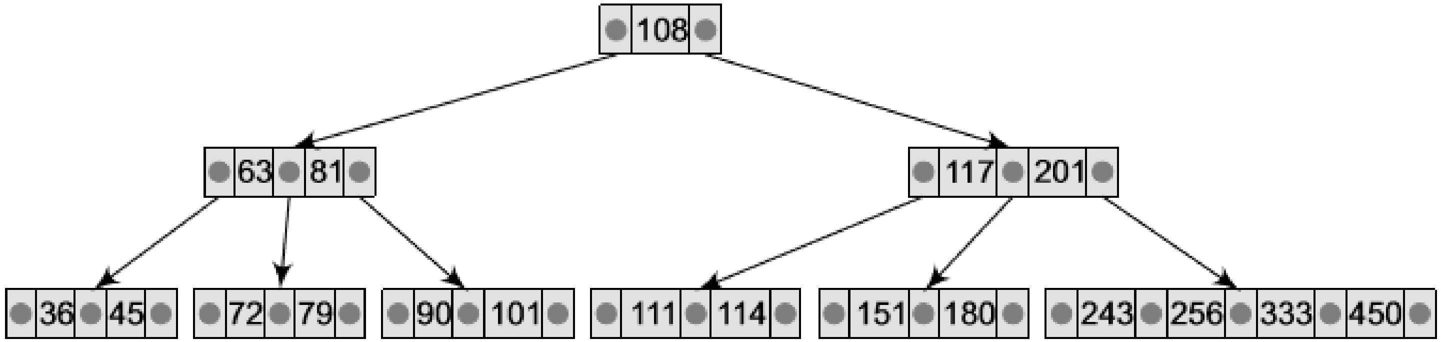
- Dahili bir düğümü silmek için, silinecek anahtarın halefini veya öncülünü silinen anahtarın konumunu işgal edecek şekilde yükseltin.
- Bu öncül veya halef her zaman yaprak düğümünde olacaktır.
- Yani sanki yaprak düğümünden bir değer silinmiş gibi işlem yapılacaktır.

B Ağaçları

Example 11.2 Consider the following B tree of order 5 and delete values 93, 201, 180, and 72 from it (Fig. 11.6(a)).



Step 1: Delete 93



Step 2: Delete 201

B Ağaçları

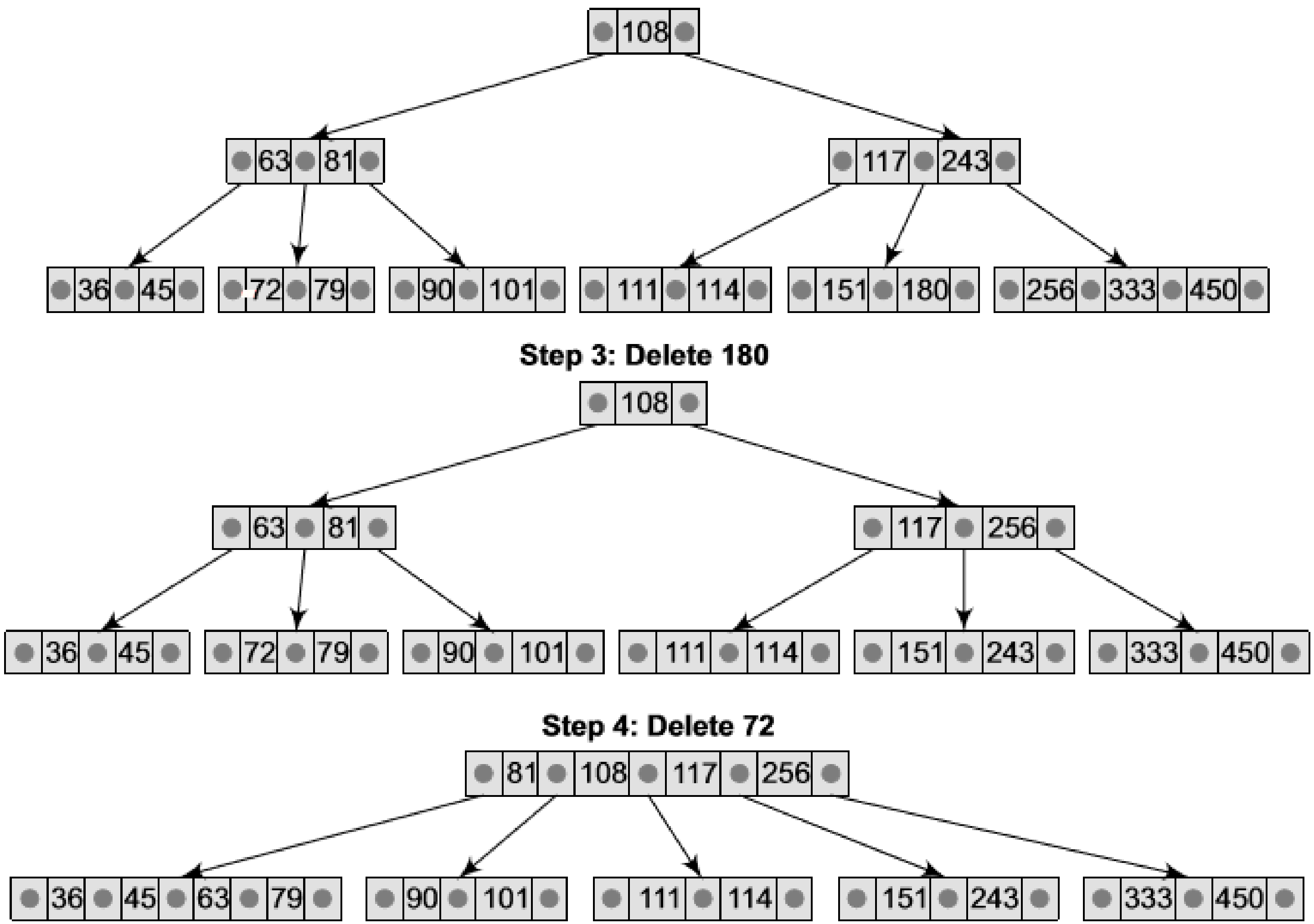


Figure 11.6 B tree

B Ağaçları

Example 11.3 Consider the B tree of order 3 given below and perform the following operations: (a) insert 121, 87 and then (b) delete 36, 109.

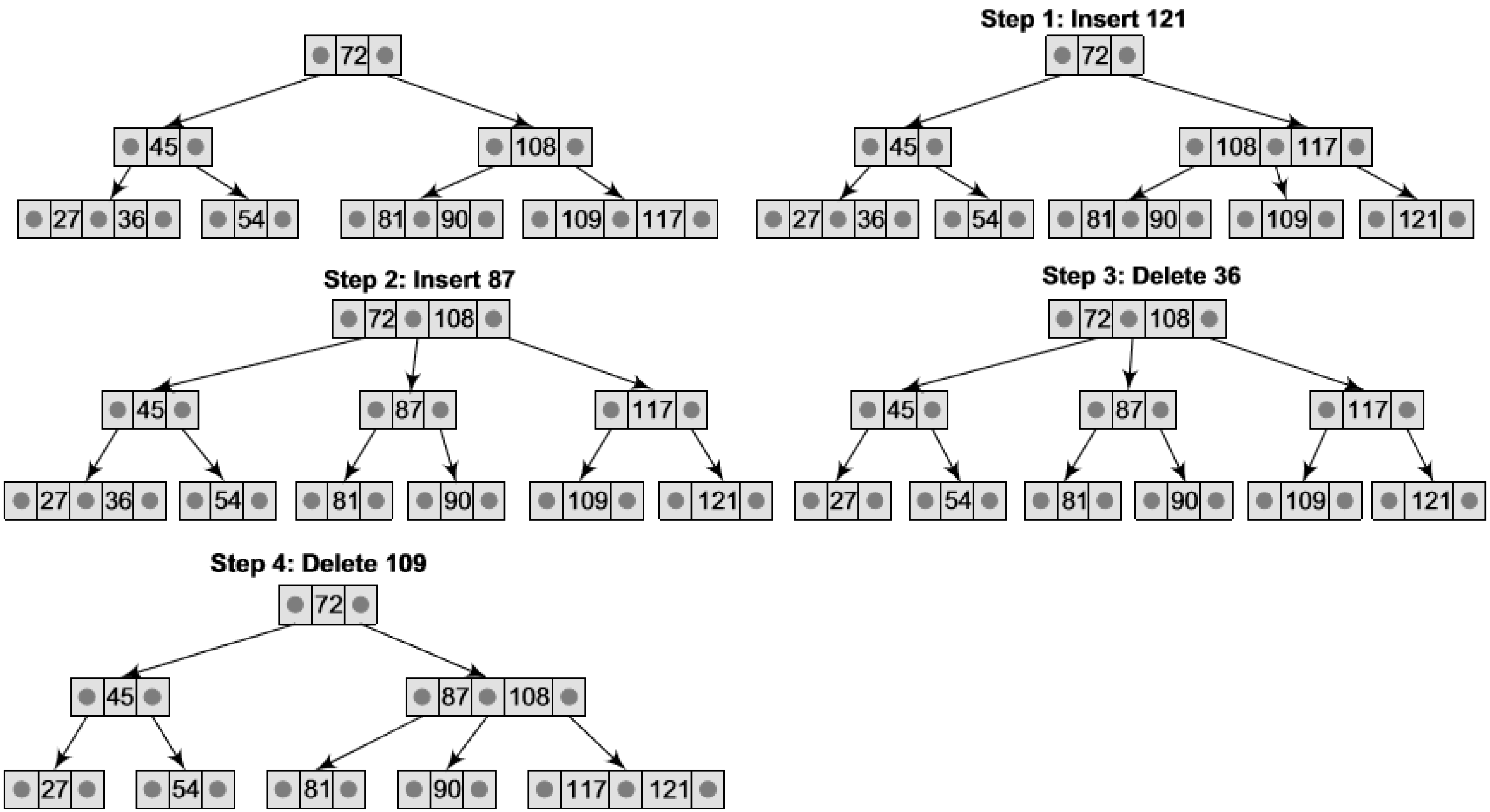
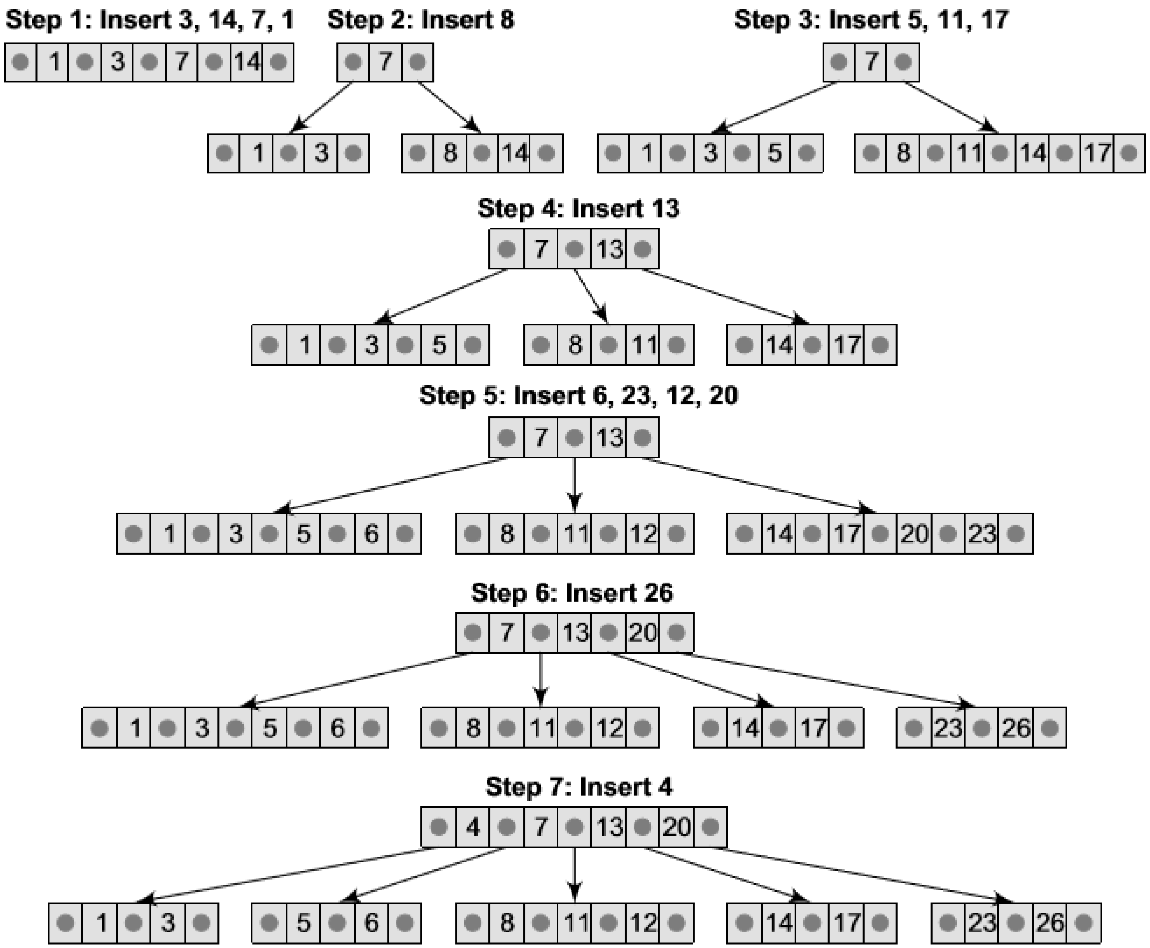


Figure 11.7 B tree

B Ağaçları

Example 11.4 Create a B tree of order 5 by inserting the following elements:
3, 14, 7, 1, 8, 5, 11, 17, 13, 6, 23, 12, 20, 26, 4, 16, 18, 24, 25, and 19.



(Contd)

B Ağaçları

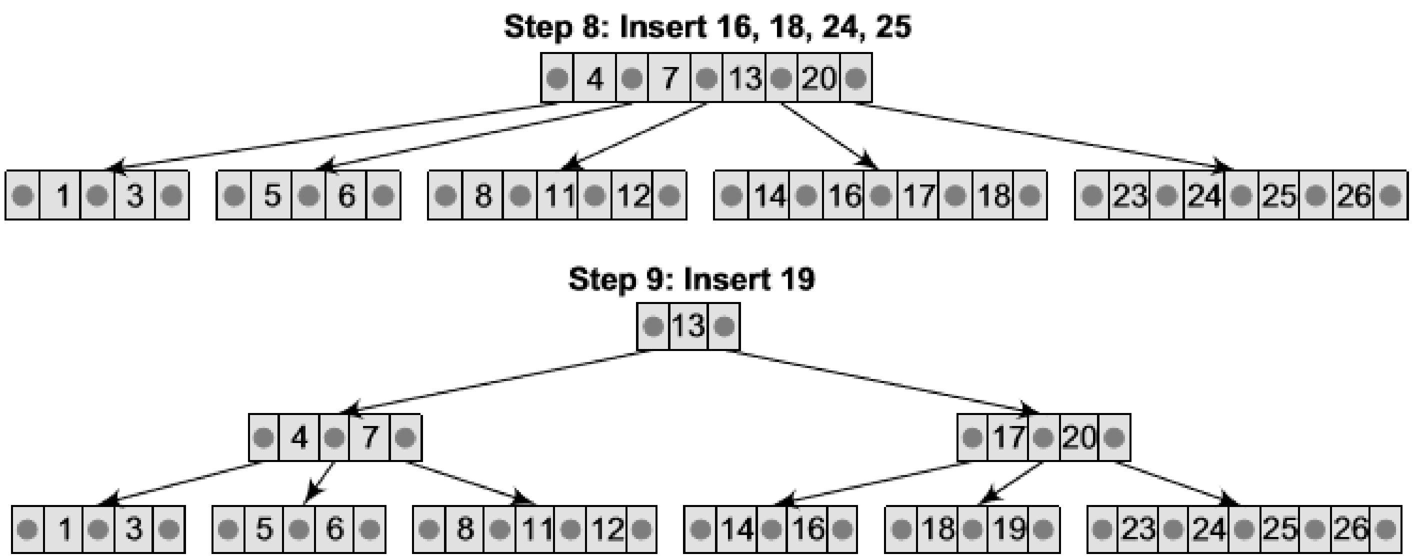


Figure 11.8 B tree

B+ Ağaçları

- B+ ağacı, kayıtların etkin bir şekilde eklenmesine, alınmasına ve kaldırılmasına olanak tanıyan bir şekilde sıralanmış verileri depolayan bir B ağacı çeşididir; her bir kayıt bir anahtarla tanımlanır.
- Bir B ağacı hem anahtarları hem de kayıtları iç düğümlerinde depolayabilirken, bir B+ ağacı ise tüm kayıtları ağacın yaprak düzeyinde depolar; iç düğümlerde yalnızca anahtarlar depolanır.
- B+ ağacının yaprak düğümleri genellikle birbirine bağlı bir liste halinde bağlanır.
- Bu, sorguları daha basit ve daha verimli hale getirme gibi ek bir avantaja sahiptir.
- B+ ağaçları genellikle ana bellekte depolanamayan büyük miktardaki verileri depolamak için kullanılır.
- B+ ağaçlarında, ikincil depolama (manyetik disk) ağaçların yaprak düğümlerini depolamak için kullanılır ve ağaçların iç düğümleri ana bellekte saklanır.
- B+ ağaçları verileri yalnızca yaprak düğümlerinde depolar.
- Diğer tüm düğümler (dahili düğümler) indeks düğümleri veya i-düğümleri olarak adlandırılır ve indeks değerlerini depolar.
- Bu, ağacı kökten başlayarak istenilen veri öğesini depolayan yaprak düğümüne kadar taramamızı sağlar.

B+ Ağaçları

- **Şekil 11.9, 3. dereceden bir B+ ağacını göstermektedir.**
- **Basitliği nedeniyle pek çok veritabanı sistemi B+ ağaç yapısı kullanılarak uygulanmaktadır.**
- **Tüm veriler yaprak düğümlerinde göründüğü ve sıralandığı için ağaç her zaman dengelidir ve veri aramayı verimli hale getirir.**
- **B+ ağacı, yaprakların yoğun bir endeksi, yaprak olmayan düğümlerin ise seyrek bir endeksi oluşturduğu çok seviyeli bir endeks olarak düşünülebilir.**
- **B+ ağaçlarının avantajları şu şekilde sıralanabilir:**
  - **1. Kayıtlar eşit sayıda disk erişiminde alınabilir**
  - **2. Yapraklar üst seviyedeki düğümlere bağlı olduğundan geniş bir yelpazede sorgular kolayca gerçekleştirmek için kullanılabilir**
  - **3. Ağacın yüksekliği daha az ve dengelidir**
  - **4. Kayıtlara hem rastgele hem de sıralı erişimi destekler**
  - **5. Anahtarlar indeksleme için kullanılır**

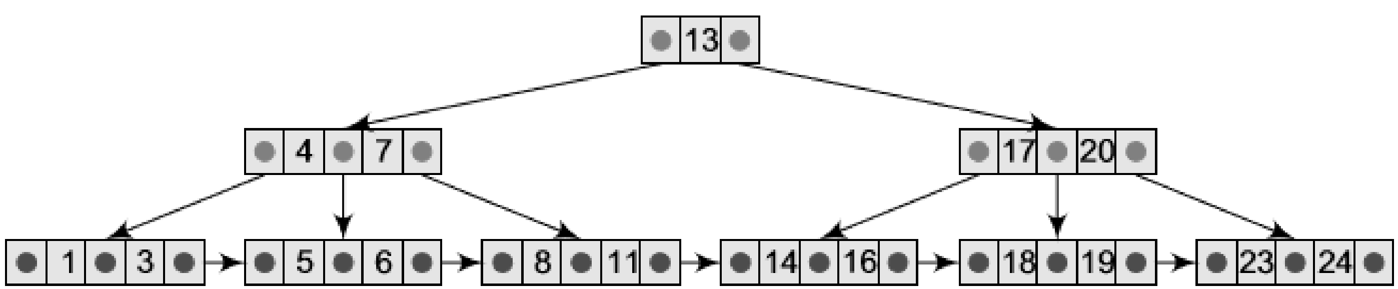


Figure 11.9 B+ tree of order 3

B+ Ağaçları

Comparison Between B Trees and B+ Trees

Table 11.1 shows the comparison between B trees and B+ trees.

Table 11.1 Comparison between B trees and to B+ trees

B Tree	B+ Tree
1. Search keys are not repeated	1. Stores redundant search key
2. Data is stored in internal or leaf nodes	2. Data is stored only in leaf nodes
3. Searching takes more time as data may be found in a leaf or non-leaf node	3. Searching data is very easy as the data can be found in leaf nodes only
4. Deletion of non-leaf nodes is very complicated	4. Deletion is very simple because data will be in the leaf node
5. Leaf nodes cannot be stored using linked lists	5. Leaf node data are ordered using sequential linked lists
6. The structure and operations are complicated	6. The structure and operations are simple

B+ Ağaçları

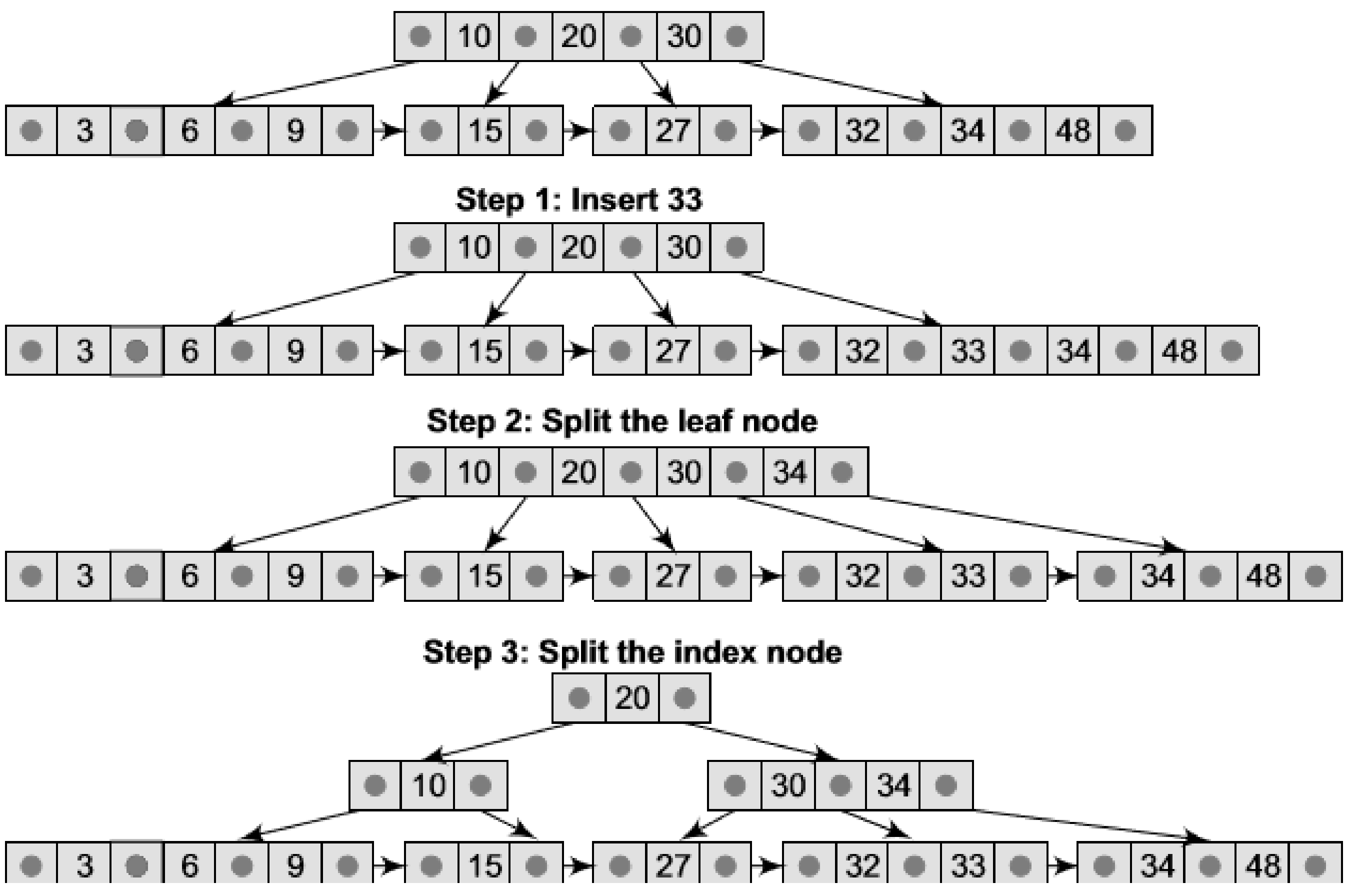
- B+ Ağacına Yeni Bir Eleman Ekleme
- Yeni bir eleman, eğer yer varsa, basitçe yaprak düğümüne eklenir.
- Ancak eklemenin yapılması gereken ağaçtaki veri düğümü doluysa, o düğüm iki düğüme bölünür.
- Bu, gelecekteki sorguların iki yeni düğüm arasında hakemlik yapabilmesi için ana dizin düğümüne yeni bir dizin değeri eklenmesini gerektirir.
- Ancak, yeni endeks değerinin ana düğüme eklenmesi, onun da bölünmesine neden olabilir.
- Aslında, bir yaprak düğümüne yeni bir değer eklendiğinde, bir yapraktan köke giden yoldaki tüm düğümler bölünebilir.
- Eğer kök düğüm ayrılırsa yeni bir yaprak düğümü oluşturulur ve ağaç bir seviye büyür.
- B+ Ağacına yeni bir düğüm ekleme adımları Şekil 11.10'da özetlenmiştir.

B+ Ağaçları

Step 3: If the index node overflows, split that node and move the middle element to next index page.

Figure 11.10 Algorithm for inserting a new node in a B+ tree

Example 11.5 Consider the B+ tree of order 4 given and insert 33 in it.



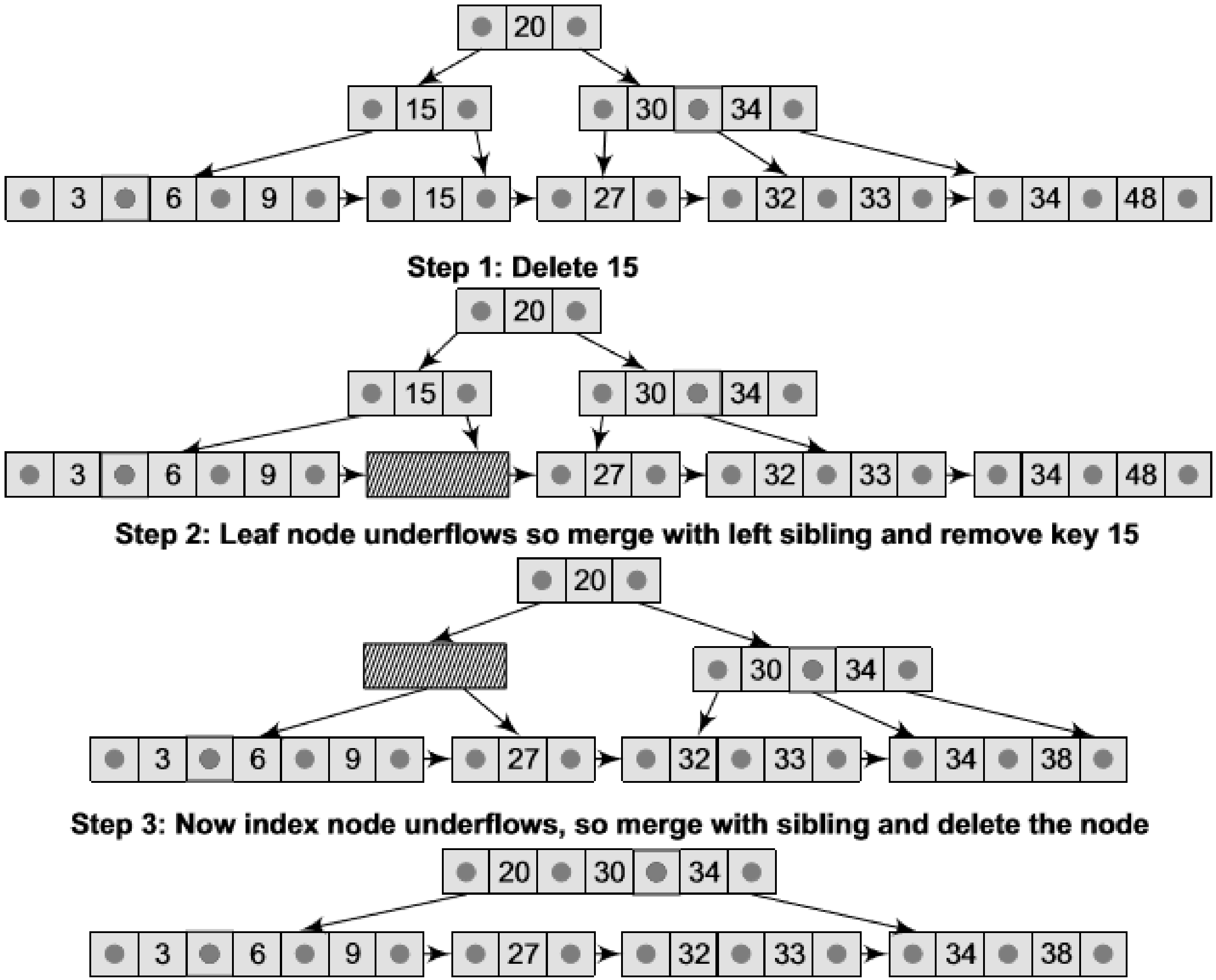
B+ Ağaçları

- B+ Ağacından bir öğeyi silme
- B ağaçlarında olduğu gibi silme işlemi her zaman yaprak düğümünden yapılır.
- Bir veri öğesinin silinmesi o düğümü boş bırakırsa, komşu düğümler incelenir ve tam dolu olmayan düğümle birleştirilir.
- Bu işlem, ana dizin düğümünden bir dizin değerinin silinmesini gerektirir; bu da dizinin boş kalmasına neden olabilir.
- Ekleme işlemine benzer şekilde, silme işlemi birleştirme-silme dalgasının yaprak düğümden kök düğüme kadar uzanmasına neden olabilir.
- Bu, ağacın bir seviye küçülmesine yol açar. B+ ağacından bir düğümü silme adımları Şekil 11.12'de özetlenmiştir.

Step 1: Delete the key and data from the leaves.
Step 2: If the leaf node underflows, merge that node with the sibling and delete the key in between them.
Step 3: If the index node underflows, merge that node with the sibling and move down the key in between them.

Figure 11.12 Algorithm for deleting a node from a B+ Tree

B+ Ağaçları


20
15
30
34
3
6
9
15
27
32
33
34
48
Step 1: Delete 15
20
15
30
34
3
6
9
27
32
33
34
48
Step 2: Leaf node underflows so merge with left sibling and remove key 15
20
30
34
3
6
9
27
32
33
34
38
Step 3: Now index node underflows, so merge with sibling and delete the node
20
30
34
3
6
9
27
32
33
34
38

2-3 Ağaç

- Önceki bölümde, ikili arama ağaçları için arama/ekleme/silme gibi işlemler için ortalama durum süresinin O(log N) ve en kötü durum süresinin O(N) olduğunu gördük; burada N ağaçtaki düğüm sayısıdır.
- Ancak, yüksekliği O(log N) olan dengeli bir ağaç her üç yöntem için de her zaman O(log N) süresini garanti eder.
- Yüksekliği dengelenmiş ağaçların tipik örnekleri arasında AVL ağaçları, kırmızı-siyah ağaçlar, B ağaçları ve 2-3 ağaçları bulunur.
- Bu veri yapılarını daha önceki bölüm ve kısımda tartışmıştık; şimdi 2-3 ağaçlarını tartışacağız. 2-3 ağacında, her iç düğümün iki veya üç çocuğu vardır.
  - İki çocuğu olan düğümlere 2 düğüm denir. 2 düğümün bir veri değeri ve iki çocuğu vardır
  - Üç çocuğu olan düğümlere 3 düğüm denir. 3 düğümlerin iki veri değeri ve üç çocuğu vardır (sol çocuk, orta çocuk ve sağ çocuk). Bu, 2-3 ağacının ikili ağaç olmadığı anlamına gelir. Bu ağaçta, tüm yaprak düğümleri aynı seviyededir (alt seviye).

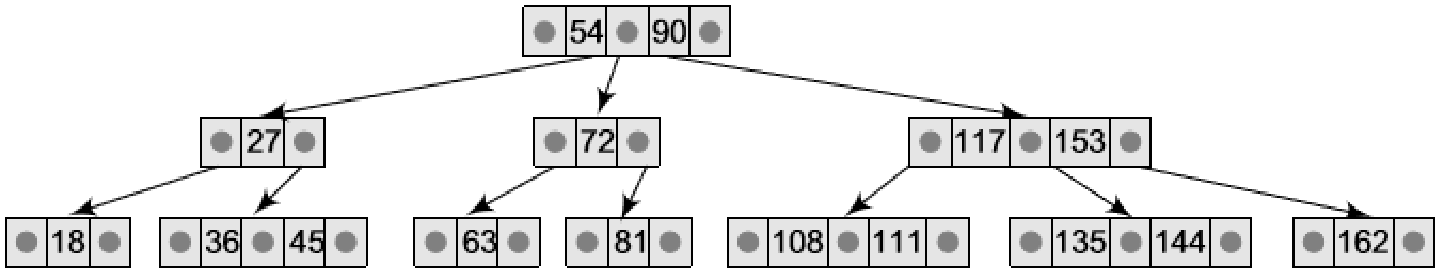


Figure 11.14 2-3 Tree

2-3 Ağaç

- **2-3 Ağacında bir öğeyi arama**
- **Arama işlemi, x veri değerinin 2-3 ağaç T'de bulunup bulunmadığını belirlemek için kullanılır.**
- **2-3'lü bir ağaçta değer arama süreci, ikili arama ağacında değer aramaya çok benzer.**
- **Bir veri değeri x'in aranması kökten başlar. Eğer k_1 ve k_2 kök düğümde saklanan iki değerse, o zaman**
  - **eğer x < k1 ise sol çocuğa geç.**
  - **eğer x ≥ k1 ise ve düğümün sadece iki çocuğu varsa, sağdaki çocuğa geç.**
  - **eğer x ≥ k1 ve düğümün üç çocuğu varsa, o zaman x < k_2 ise ortadaki çocuğa, aksi takdirde x ≥ k_2 ise sağdaki çocuğa geç. İşlemin sonunda, x veri değerine sahip düğüme ancak ve ancak x bu yapraktaysa ulaşılır.**

2-3 Ağaç

Example 11.7 Consider the 2-3 tree in Fig. 11.14 and search 63 in the tree.

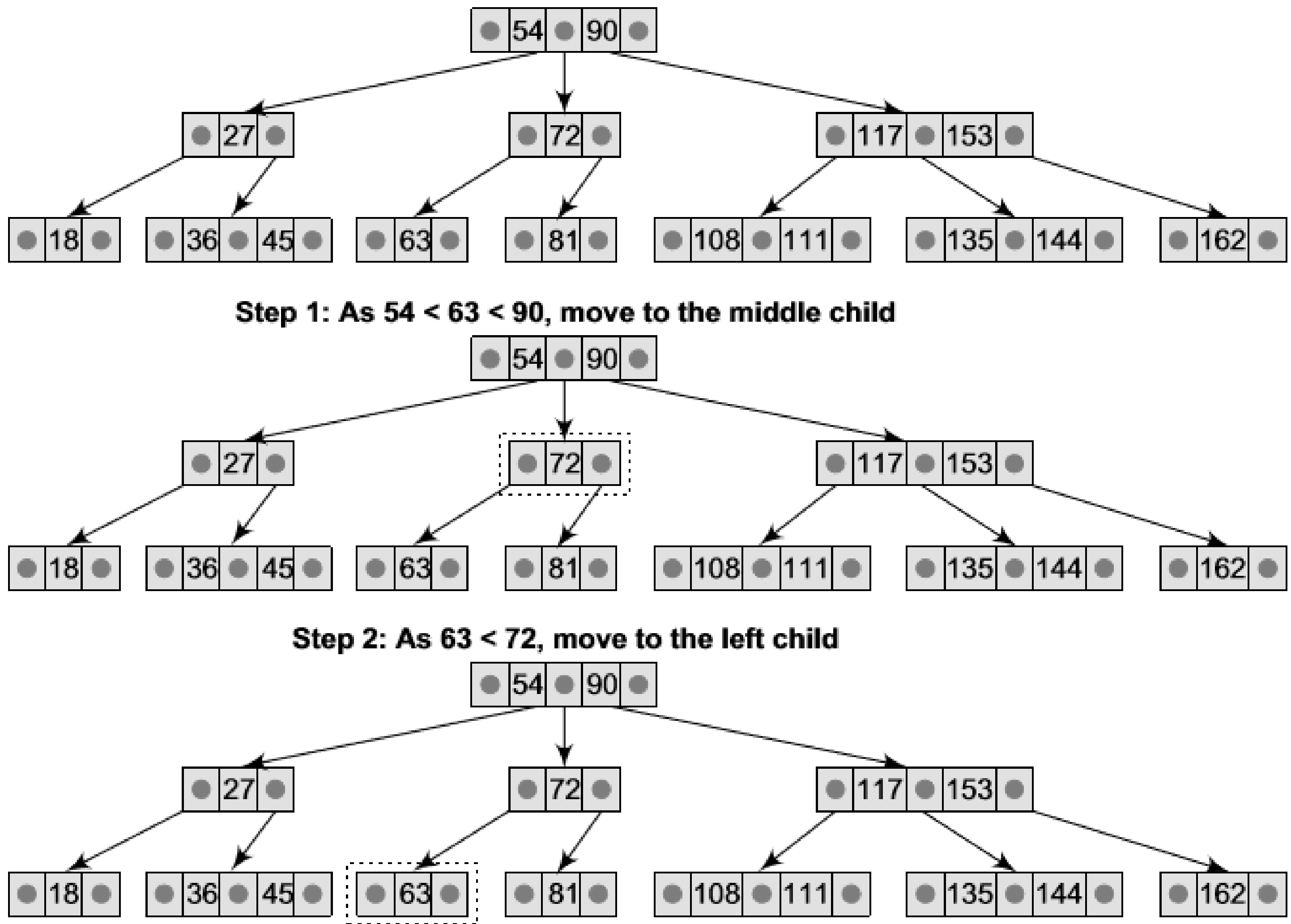


Figure 11.15 Searching for element 63 in the 2-3 tree of Fig. 11.14

2-3 Ağaç

- **2-3 Ağacına Yeni Bir Eleman Ekleme**
- **2-3 ağacına yeni bir değer eklemek için, değerin uygun konumu yaprak düğümlerinden birinde bulunur.**
- **Yeni değerin eklenmesinden sonra 2-3 ağacının özellikleri ihlal edilmezse ekleme işlemi tamamlanmış olur.**
- **Aksi takdirde herhangi bir özellik ihlal edilirse ihlal eden düğümün bölünmesi gerekir (Şekil 11.16).**
- **Bir düğümü bölme Bir düğüm üç veri değeri ve dört çocuğu olduğunda bölünür. Burada, P ebeveyndir ve L, M, R sol, orta ve sağ çocukları belirtir.**

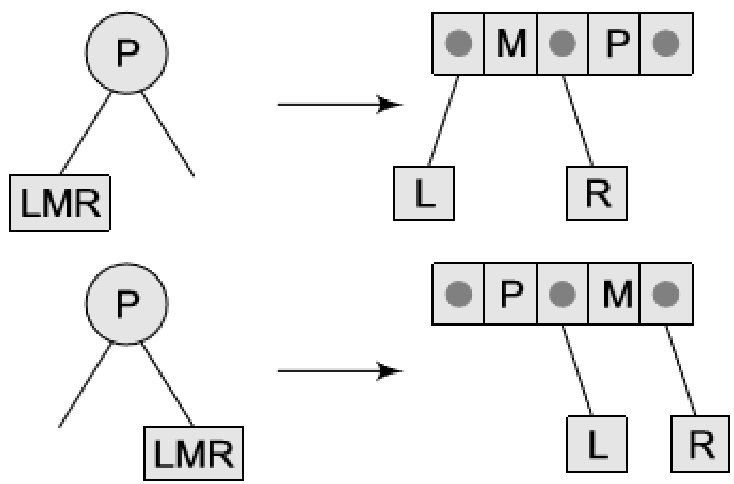


Figure 11.16(a)

2-3 Ağaç

Example 11.8 Consider the 2-3 tree given below and insert the following data values into it: 39, 37, 42, 47.

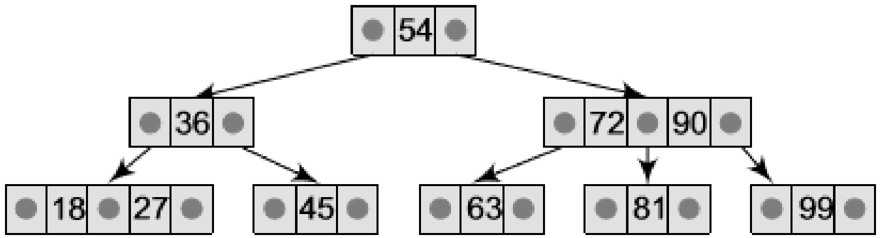


Figure 11.16(b)

Step 1: Insert 39 in the leaf node The tree after insertion can be given as

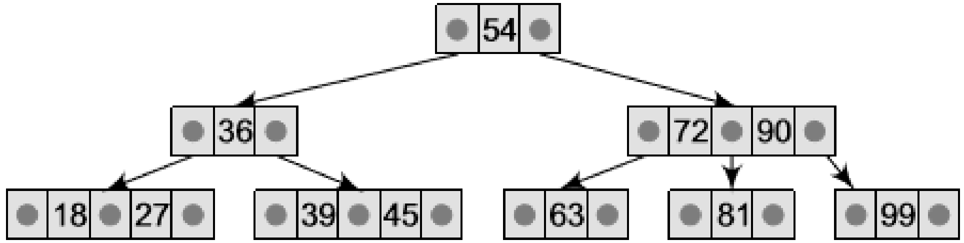


Figure 11.16(c)

2-3 Ağaç

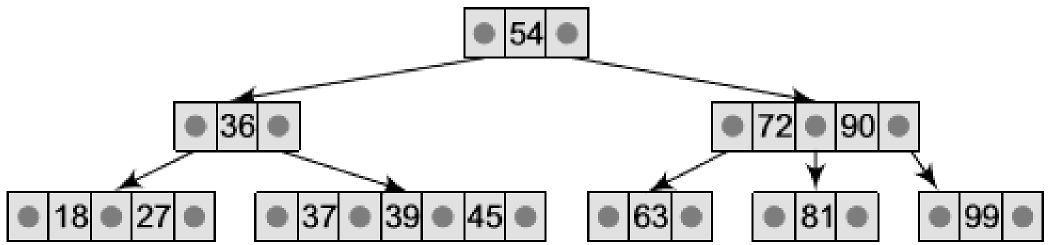


Figure 11.16(d)

After splitting the leaf node, the tree can be given as below.

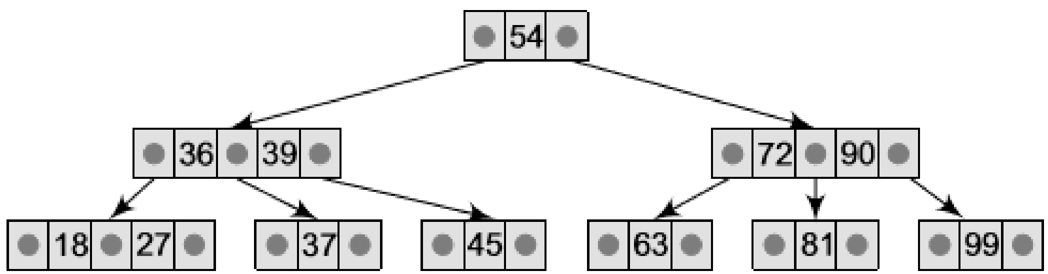
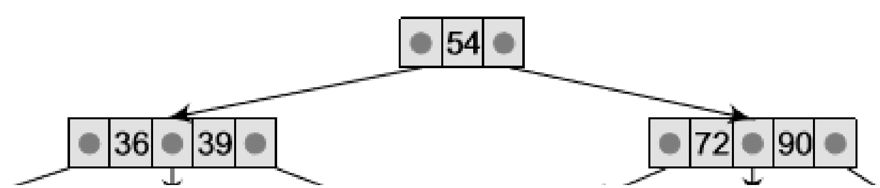


Figure 11.16(e)

Step 3: Insert 42 in the leaf node The tree after insertion can be given as follows.



2-3 Ağaç

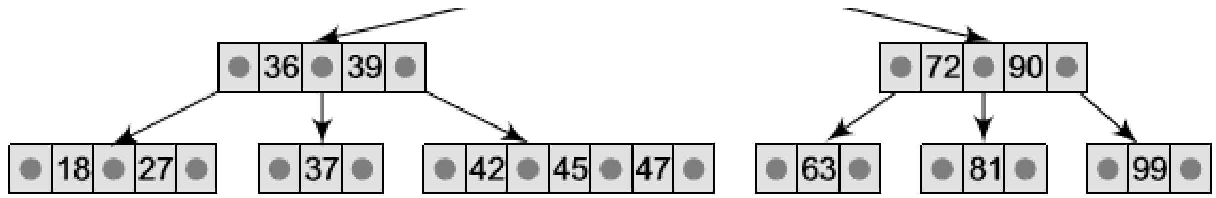


Figure 11.16(g)

The leaf node has three data values. Therefore, the node is violating the properties of the tree and must be split.

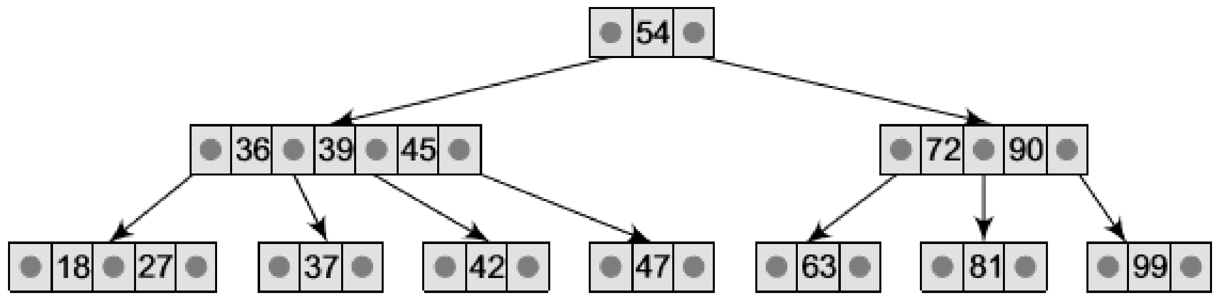
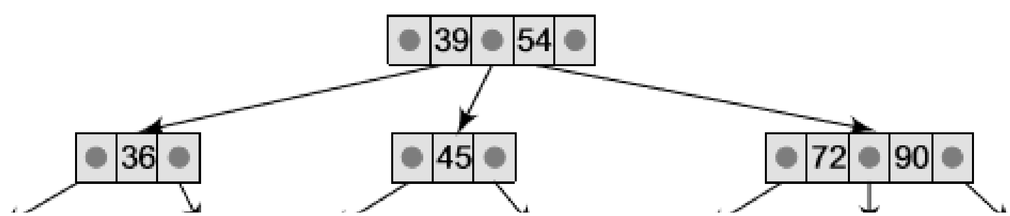


Figure 11.16(h)

The parent node has three data values. Therefore, the node is violating the properties of the tree and must be split.



2-3 Ağaç

- **2-3 Ağacından bir öğeyi silme**
- **Silme işleminde 2-3 ağacından belirtilen bir veri değeri silinir.**
- **Bir düğümden bir değerin silinmesi ağacın özelliğini ihlal ediyorsa, yani bir düğümde birden az veri değeri kalmışsa, 2-3 ağacının genel özelliklerini korumak için iki düğüm birleştirilmelidir.**
- **Ekleme işleminde yeni değerin herhangi bir yaprak düğümüne eklenmesi gerekirken, silme işleminde değerin yaprak düğümünden silinmesi gerekmez.**
- **Değer herhangi bir düğümden silinebilir.**
- **Bir x değerini silmek için, sıralı halefiyle değiştirilir ve sonra kaldırılır. Bir düğüm, bir değer silindikten sonra boşalırsa, ağacın özelliğini geri yüklemek için başka bir düğümle birleştirilir.**

Example 11.9 Consider the 2-3 tree given below and delete the following values from it: 69, 72, 99, 81.

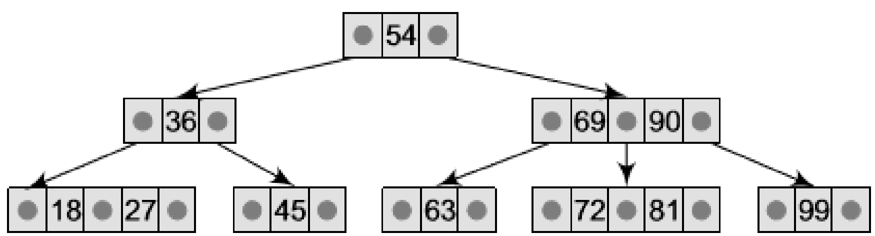


Figure 11.17(a)

To delete 69, swap it with its in-order successor, that is, 72. 69 now comes in the leaf node. Remove the value 69 from the leaf node.

2-3 Ağaç

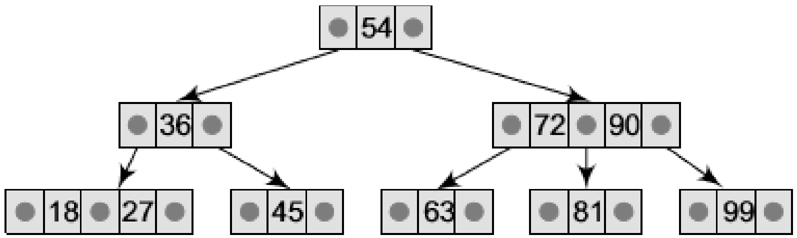


Figure 11.17(b)

72 is an internal node. To delete this value swap 72 with its in-order successor 81 so that 72 now becomes a leaf node. Remove the value 72 from the leaf node.

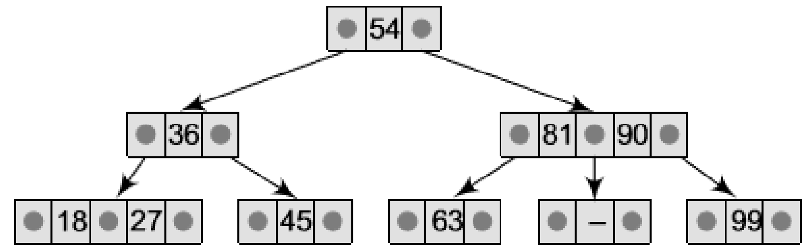


Figure 11.17(c)

Now there is a leaf node that has less than 1 data value thereby violating the property of a 2-3 tree. So the node must be merged. To merge the node, pull down the lowest data value in the parent's node and merge it with its left sibling.

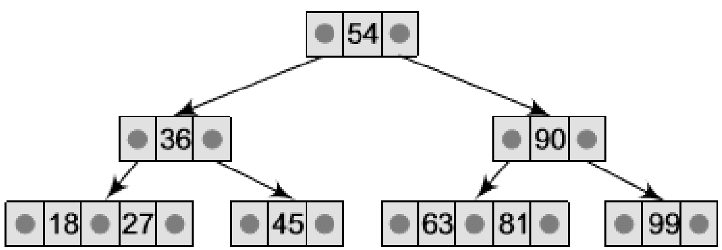


Figure 11.17(d)

99 is present in a leaf node, so the data value can be easily removed.

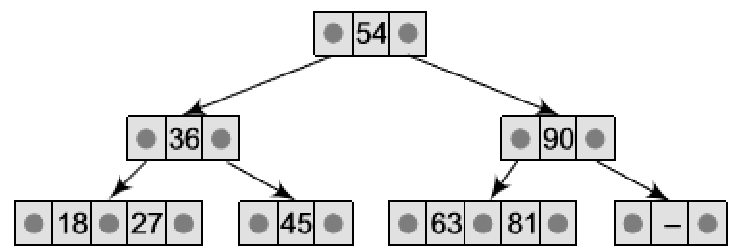


Figure 11.17(e)

Now there is a leaf node that has less than 1 data value, thereby violating the property of a 2-3 tree. So the node must be merged. To merge the node, pull down the lowest data value in the parent's node and merge it with its left sibling.

2-3 Ağaç

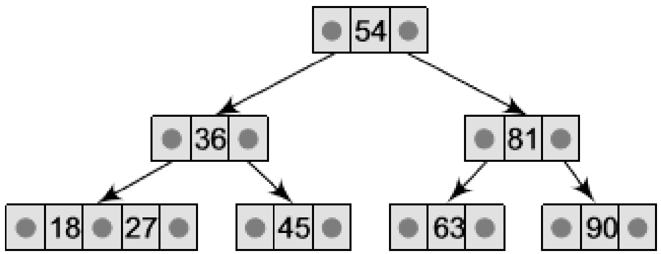


Figure 11.17(f)

81 is an internal node. To delete this value swap 81 with its in-order successor 90 so that 81 now becomes a leaf node. Remove the value 81 from the leaf node.

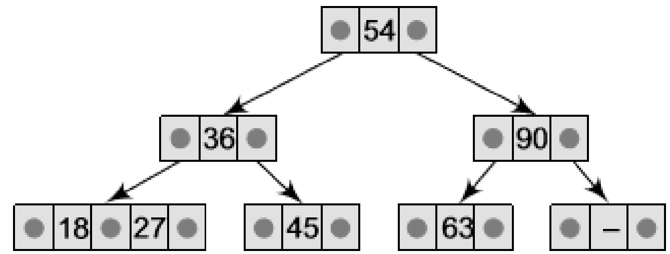


Figure 11.17(g)

Now there is a leaf node that has less than 1 data value, thereby violating the property of a 2-3 tree. So the node must be merged. To merge the node, pull down the lowest data value in the parent's node and merge it with its left sibling.

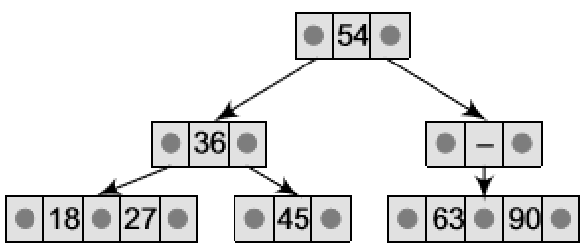


Figure 11.17(h)

An internal node cannot be empty, so now pull down the lowest data value from the parent's node and merge the empty node with its left sibling.

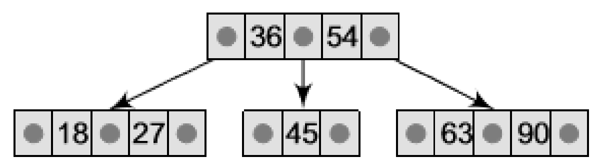


Figure 11.17(i) Deleting values from the given 2-3 tree