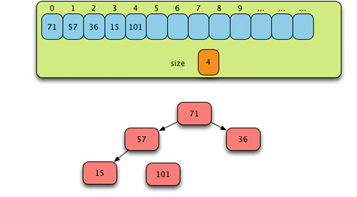
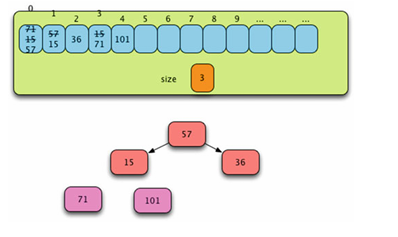
**9.6 Aşama II'nin Analizi**   227



**Şekil 9.14** Faz II'nin İlk Geçişinden Sonra



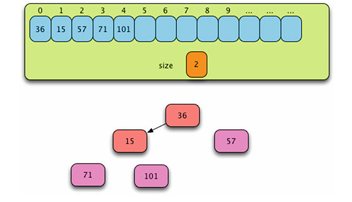
**Şekil 9.15** Faz II'nin İkinci Geçişinden Sonra

Aşama II'nin üçüncü geçişi sırasında 57 son konumuna yerleştirilir ve kendisine yer açmak için 36 ile değiştirilir. Her ne kadar \_\_siftDownFromTo çağrılsa da, 36 en üstte olduğu ve yığındaki en büyük değer olduğu için yığın içinde herhangi bir değer hareketi gerçekleşmez (Şekil.9.16).

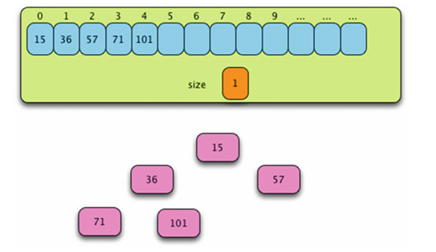
Dördüncü ve son geçiş sırasında 36, 15 ile değiştirilir. Bu sefer \_siftDownFromTo çağrısına gerek yoktur çünkü yığın takastan sonra sadece 1 boyutundadır. Büyüklüğü 1 olan bir yığın zaten sıralanmış ve doğru yerde olduğundan Liste artık Şekil 9.17'de gösterildiği gibi ek bir dizi kullanmadan yerinde sıralanır.

[www.it-ebooks.info](https://www.it-ebooks.info)

228 9 Yığınlar



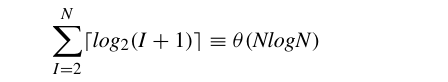
**Şekil 9.16** Faz II’nin Üçüncü Geçişinden Sonra



**Şekil 9.17** Faz II'nin Dördüncü ve Son Geçişinden Sonra

**9.7 İkinci Aşama Analizi**

Faz II'nin çalışması, N−1 kez çağrılan \_\_siftDownFromTo yöntemine yapılan çağrılardadır. Her çağrı, ağaçtaki her seferinde bir öğe küçülen bir öğeyi elemelidir. Bu bölümün başlarında, ortalama ve en kötü durumdaki iş miktarının aşağıdakilerle orantılı olduğunu belirlemek için analizi yaptık



Aşama II'nin en iyi durumu, yığındaki tüm değerlerin aynı olmasını gerektirir. Bu durumda, hesaplama karmaşıklığı O (N) olacaktır, çünkü değerler asla düşmeyecektir.

[www.it-ebooks.info](https://www.it-ebooks.info)

**9.7 Aşama II'nin Analizi** 229

Bu en iyi durum senaryosu iyi bir noktayı ortaya koyuyor. Değerin ne kadar aşağı elendiğini sınırlayabilseydik, I. Aşamayı hızlandırabilirdik. Konumuz veya bir sonraki bölümümüz bu.

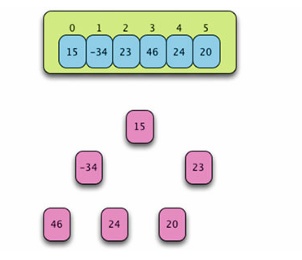
**9.8 Yığın Sıralama Algoritması Sürüm 2**

Birinci versiyonda, yığın algoritması Faz I ve Faz II sırasında O (N log N) karmaşıklığına ulaştı. İkinci versiyonda, yığın algoritmasının Faz I'ini O (N) karmaşıklığına kadar hızlandırabileceğiz. Bunu, yeni eklenen her değerin ne kadar elenmesi gerektiğini sınırlayarak yaparız. Fikir oldukça basit ama yine de güçlü bir teknik. Her öğeyi yığının en üstüne eklemek yerine, yığını veya yığınları aşağıdan yukarıya doğru oluşturacağız. Bu, yığınımızı oluşturmaya listenin başından değil sonundan başlayarak yaklaşacağımız anlamına gelir. Bir örnek bunu daha açık hale getirmeye yardımcı olacaktır. Şekil 9.18'deki yığın sıralamayı kullanarak sıralamak istediğimiz değerlerin listesini düşünün.

Listenin ilk elemanından başlamak yerine, listenin diğer ucundan başlayacağız. Göreceğimiz gibi son unsurla başlamaya gerek yoktur. Ağaçtaki bir düğümün üst öğesi olan bir düğüm seçmemiz gerekiyor. Nihai yığın ikili bir yığın olduğundan, sahip olduğumuz özellik, ağacın düğümlerinin yarısının yaprak düğümleri olması ve yığın içindeki herhangi bir düğümün üst öğesi olamamasıdır. İlk ana endeksi aşağıdaki gibi hesaplayabiliriz.

parentIndex = (size − 2)//2

Yukarıdaki boyut, sıralanacak listenin boyutudur. Listenin 0 ila -1 boyutunda indeksleri olduğundan, her durumda uygun üst İndeksi hesaplamak için iki çıkarmamız gerektiğini unutmayın. Bu durumda, bu üst Dizin 2'dir. Yığınlarımızı aşağıdan yukarıya doğru oluşturmaya başlamak için listede dizin 2 ile başlamamız gerekir. İndeks 2 ilk üst olacak ve gerektiği kadar aşağıya doğru eleyeceğiz.



**Şekil 9.18** Yığınlanacak Bir Liste

[www.it-ebooks.info](https://www.it-ebooks.info)

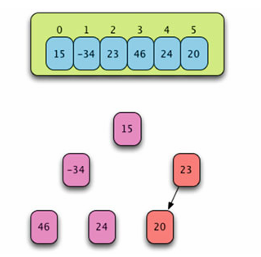
230 9 Yığınlar

childIndex1 = 2 ∗ parentIndex + 1 = 2 ∗ 2 +1 = 5

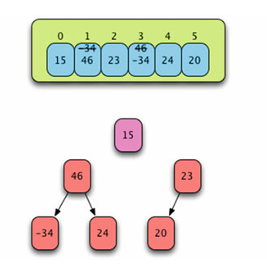
childIndex2 = 2 ∗ parentIndex + 2 = 2 ∗ 2 +2 = 6

Bu indekslerden ikincisi listenin son indeksinin ötesinde olduğundan, \_\_siftDownFromTo yöntemi alt İndeks 2'yi dikkate almayacaktır. 20 ve 23'ü göz önünde bulundurduktan sonra, bu iki düğümün aslında Şekil 9.19'da gösterildiği gibi bir yığın oluşturduğunu görüyoruz. Bunu aşağıdaki şekillerde bir okla birleştirerek göstereceğiz. Artık 5 yığınımız var, başladığımızdan bir tane daha az. Daha da önemlisi, üst ögeyi en fazla bir pozisyon aşağı kaydırmak zorunda kaldık.

Ardından, listede bir geri giderek dizin 1'e gidiyoruz. Bu düğümden başlayacağımızı belirterek \_\_siftDownFromTo'yu çağırıyoruz. Bunu yapmak, sift down yönteminin takas etmek için iki çocuktan büyük olanı seçmesine neden olur ve sonuç olarak -34, 46 ve 24 değerlerinden bir yığın oluşturur. Bu durum Şekil 9.20'de gösterilmektedir.



**Şekil 9.19** Bir Alt Yığın Oluşturduktan Sonra



**Şekil 9.20** İkinci Bir Alt Yığın Oluşturduktan Sonra

[www.it-ebooks.info](https://www.it-ebooks.info)