10.3 AVL Trees 241

AVL Trees, ancak belki de tree her bir düğümün yüksekliğini korumaktan biraz daha zordur. Bölümün ilerleyen kısımlarında bu algoritmaların her bir düğümün yüksekliğini koruyan modifikasyonları tartışılacaktır. AVL Ağaçlarında ister yükseklik ister denge saklansın, ağaç işlemlerinin karmaşıklığı etkilenmez.

# **10.3.4 AVL Tree Yinelemeli Ekleme**

Son bölümde açıklandığı gibi, yükseklik dengeli AVL trees için ekleme algoritmasının iki çeşidi vardır. Ekleme yinelemeli veya özyinelemeli olarak gerçekleştirilebilir. Denge ayrıca açık bir şekilde saklanabilir veya her bir alt ağacın yüksekliğinden hesaplanabilir. Bu bölümde, her bir düğümün yüksekliğini muhafaza etmeden dengenin açık bir şekilde nasıl muhafaza edileceği açıklanmaktadır.

Yüksekliği dengelenmiş bir AVL tree yeni bir değerin yinelemeli olarak eklenmesi, yeni eklenen değere giden yolun takip edilmesini gerektirir. Bu yolu korumak için bir yığın kullanılır. Algoritmada bu yığına *yol yığını* diyeceğiz. Yeni bir düğüm eklemek için, kökten yeni düğümün konumuna giden benzersiz arama yolunu izleriz ve ilerledikçe her düğümü, tıpkı bir ikili arama ağacına

ekliyormuşuz gibi, yol yığınına iteriz. Yeni düğümün hedefine giden yol boyunca ilerlerken, karşılaştığımız tüm düğümleri *yol yığınına* iteriz. Yeni öğeyi ikili arama ağacı özelliğine göre olması gereken yere yerleştiririz. Daha sonra algoritma, yol yığınından değerleri çıkarmaya ve ayarlanmadan önce sıfıra eşit olmayan bir dengeye sahip bir düğüm bulunana kadar dengelerini ayarlamaya devam eder. Sıfır olmayan bakiyeye sahip en yakın ata olan bu düğüme *pivot* adı verilir. Pivot ve yeni değerin konumuna bağlı olarak, aşağıda açıklanan birbirini dışlayan üç durum göz önünde bulundurulmalıdır. Aşağıdaki 3. durumda ayarlamalar yapıldıktan sonra, pivotta köklenen alt tree için yeni bir kök düğüm olabilir. Bu durumda, pivotun ebeveyni yol yığınındaki bir sonraki düğümdür ve yeni alt tree bağlanabilir. Pivot patlatıldıktan sonra yol yığını boşsa, ağacın kökü pivottur. Bu durumda, AVL tree kök düğümü, tree yeni kök düğüm gösterecek şekilde yapılabilir. Belirtildiği gibi Yukarıda, tree yeni bir değer eklerken üç durumdan biri ortaya çıkacaktır.

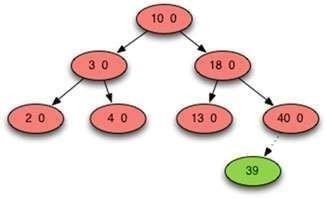
**Durum 1: Pivot Yok** Pivot düğüm yoktur. Bu durumda, arama yolundaki her bir düğümün dengesini, her bir düğümün anahtarına göre yeni anahtarın göreli değerine göre ayarlayın. Yeni düğüme giden yolu incelemek için *yol* yığınını kullanabilirsiniz.

Bu durum AVL tree 39'un ekleneceği Şekil 10.1'de gösterilmektedir. Her düğümde değer solda ve denge sağda verilmiştir. 10, 18 ve 40 içeren düğümlerin her biri *yol yığınına* itilir. 39'u içeren yeni düğümün bakiyesi 0 olarak ayarlanır. 40'ı içeren düğümün yeni bakiyesi

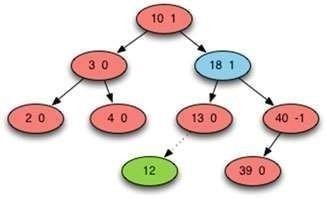
-1. 16 içeren düğümün yeni bakiyesi 1'dir. Eklemeden sonra kök düğümün bakiyesi 1'dir çünkü 39 sağına eklenmiştir ve bu nedenle bakiyesi bir artar. Yeni değer 40'ı içeren düğümün soluna eklenir, bu nedenle bakiyesi bir azalır. Şekil

* 1. yeni değerin eklenmesiyle değişen tree’ yi göstermektedir.

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



**Şekil 10.1** AVL Tree Vaka 1-Pivot Düğüm Yok



**Şekil 10.2** AVL Vaka Tree 2-Döndürme Yok

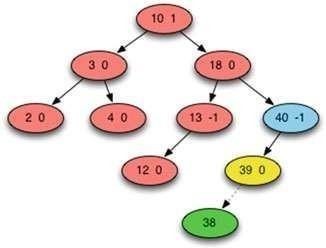
**Durum 2: Dengeleri Ayarla** Pivot düğüm mevcuttur. Ayrıca, yeni düğümün eklendiği pivot düğümün alt ağacı daha küçük yüksekliğe sahiptir. Bu durumda, arama yolu boyunca yeni düğümden pivot düğüme kadar olan düğümlerin dengesini değiştirin. Pivot düğümün üzerindeki düğümlerin dengeleri etkilenmez. Bu doğrudur çünkü pivot düğümde köklenen alt ağacın yüksekliği yeni düğümün eklenmesiyle değişmez.

Şekil 10.2 bu durumu göstermektedir. *Anahtarı* 12 olan öğe AVL tree eklenmek üzeredir. 18'i içeren düğüm pivot düğümdür. Eklenecek değer 18'den küçük olduğundan ve 18'i içeren düğümün bakiyesi 1 olduğundan, yeni düğüm muhtemelen ağacın daha iyi dengelenmesine yardımcı olabilir. AVL tree AVL tree olarak kalır. Pivota kadar olan düğümlerin dengesi ayarlanmalıdır. Pivotun üzerindeki dengelerin ayarlanması gerekmez çünkü bunlar etkilenmez. Şekil 10.3, tree 12 eklendikten sonra tree nasıl göründüğünü göstermektedir.

**Durum 3:** Pivot düğüm mevcuttur. Ancak bu kez yeni düğüm, pivotun daha büyük yükseklikteki alt tree (dengesizlik yönündeki alt ağaç) eklenir. Bu, yeni düğüm eklendikten sonra pivot düğümün -2 veya 2 dengesine sahip olmasına neden olur, bu nedenle tree artık bir AVL tree olmayacaktır. Burada, tree AVL durumuna geri getirmek için *tek* bir *döndürme* veya *çift döndürme* gerektiren iki alt durum vardır. Dengesizlik yönündeki pivot düğümün çocuğunu *kötü çocuk* olarak adlandırın.

**Alt Durum A: Tek** **Rotasyon** Bu alt durum, yeni düğüm dengesizlik yönünde olan kötü çocuğun alt ağacına eklendiğinde ortaya çıkar. *Olasılık*

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



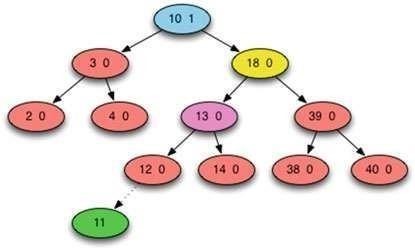
**Şekil 10.3** AVL Tree Vaka 3A-Tek Rotasyon

pivot düğümde dengesizliğin tersi yönde bir döndürmedir. Döndürme işleminden sonra ağaç hala bir ikili arama tree’dir. Buna ek olarak, pivotta köklenen alt tree bir kez daha dengelenir ve toplam yüksekliği bir azalır.

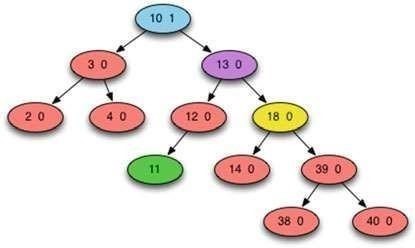
Şekil 10.3 bu alt durumu göstermektedir. 38 değeri, 39'u içeren düğümün solundaki tree eklenecektir. Ancak bunu yapmak, 40'ı içeren düğümün o1 dengesinin -2'ye düşmesine neden olur ki bu da uygunsuz dengeye sahip en yakın ata ve pivot düğümdür. Sarı düğüm *kötü çocuktur.* Buna ek olarak, 38 dengesizlikle aynı yönde yerleştirilmektedir. Dengesizlik sol taraftadır ve yeni değer sol tarafa eklenmektedir. Çözüm, 40'ta köklenen alt tree’yı sağa döndürmektir, bu da Şekil 10.4'te gösterilen tree ‘le sonuçlanır.

**Alt Durum B: Çift Döndürme** Bu alt durum, yeni düğüm dengesizliğin ters yönünde olan kötü çocuğun alt tree’ye eklendiğinde ortaya çıkar. Bu alt durum için, kötü çocuğun arama yolu üzerinde bulunan çocuk düğümünü *kötü torun olarak* adlandırın. Bazı durumlarda, kötü torun olmayabilir. Şekil 10.4'te kötü torun mor düğümdür. Çözüm aşağıdaki gibidir:

1. Kötü çocukta dengesizlik yönünde tek bir rotasyon gerçekleştirin.
2. Pivotta dengesizlikten uzağa doğru tek bir dönüş gerçekleştirin.



**Şekil 10.4** AVL Vaka Tree 3B-Çift Rotasyon [www.it-ebooks.info](https://www.it-ebooks.info)

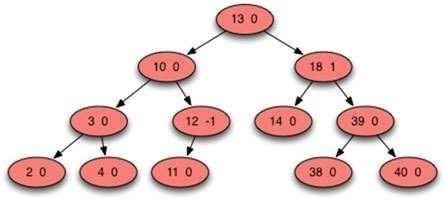


**Şekil 10.5** AVL Tree Vaka 3B Adım 1 Döndürme Yönü

Yine, tree hala bir ikili arama tree’dir ve orijinal pivot düğümün konumundaki alt tree’nin yüksekliği çift döndürme ile değişmez. Şekil 10.4 bu durumu göstermektedir. Bu durumda pivot tree’nin köküdür. 18'i içeren düğüm kötü çocuktur. Kötü torun ise 13'ü içeren düğümdür (Şekil 10.5).

Ağaçtaki dengesizlik pivotun sağındadır. Yine de 11 kötü çocuğun soluna yerleştirilmektedir. İlk adım, kötü çocukta sağa doğru bir rotasyondur. Bu, 11'*i yukarı getirerek tree’nin* sağ tarafını dengelemeye bir *şekilde* yardımcı olur. Şekil 10.6'da gösterilen ikinci adım, pivotta sola dönerek tüm ağacı tekrar dengeye getirir.

Bu algoritmanın en zor kısmı bakiyeleri doğru şekilde güncellemektir. İlk olarak, pivot, kötü çocuk ve kötü torun değişebilecek bakiyeleri içerir. Eğer kötü torun yoksa, pivotun ve kötü çocuğun bakiyeleri sıfır olacaktır. Burada olduğu gibi kötü bir torun varsa, pivot ve kötü çocuğun bakiyelerini belirlemek için biraz daha fazla iş vardır. Kötü torun mevcut olduğunda, çift döndürmeden sonra bakiyesi 0 olur. Kötü çocuğun ve pivotun bakiyeleri dönüşün yönüne ve yeni öğenin ve kötü torunun öğesinin değerine bağlıdır. Bu durumlarda hem pivotun hem de kötü torunun bakiyelerini belirlemek için bu durum vaka bazında analiz edilebilir. Bir sonraki bölümde bakiyelerin nasıl hesaplandığını inceleyeceğiz.



**Şekil 10.6** AVL Tree Vaka 3B Adım 2 Uzağa Döndür

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