Veri Yapıları & Algoritmalar

Sivas Cumhuriyet Üniversitesi

AVL Ağacına İteratif Ekleme – Rotasyonlar - AVL Ağacına Recursive Ekleme

Himmet Can Umutlu – 2023481048 | Bilişim Sistemleri ve Teknolojileri

10.3.4 - AVL Tree Iterative Insert / AVL Agacı Yinelemeli Olarak Dügüm Ekleme

Giriş

Son bölümde açıklandığı gibi, yükseklik dengeli(height balanced) AVL ağaçları için ekleme algoritmasının iki çeşidi vardır.

Ekleme yinelemeli(iterative) veya özyinelemeli(recursive) olarak gerçekleştirilebilir.

Denge(balance) ayrıca açık bir şekilde saklanabilir veya her bir alt ağacın(subtree) yüksekliğinden hesaplanabilir.

Bu bölümde, her bir düğümün yüksekliğini muhafaza etmeden dengenin açık bir şekilde nasıl muhafaza edileceği açıklanmaktadır.

(değer, bakiye, balance eş anlamlı)

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

Yeni düğümün hedefine giden yol boyunca ilerlerken, karşılaştığımız tüm düğümleri yol yığınına(path stack) 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(merkez nokta) 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 ağaç için yeni bir kök düğüm olabilir. Bu durumda, pivotun ebeveyni(parent) yol yığınındaki bir sonraki düğümdür ve yeni alt ağaca bağlanabilir. Pivot patlatıldıktan sonra yol yığını boşsa, ağacın kökü pivottur. Bu durumda, AVL ağacının kök düğümü, ağaçtaki yeni kök düğümü gösterecek şekilde yapılabilir. Belirtildiği gibi, yukarıda, ağaca yeni bir değer eklerken üç durumdan biri ortaya çıkacaktır.

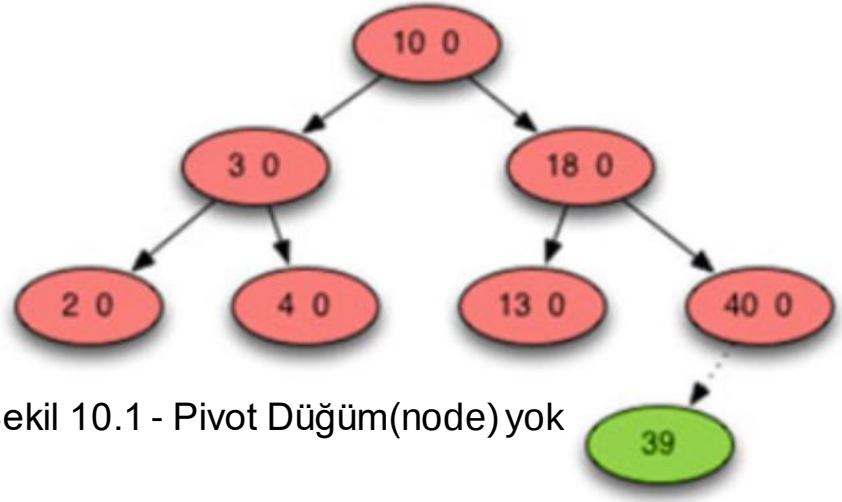
Durum 1

Pivot düğüm yoktur.

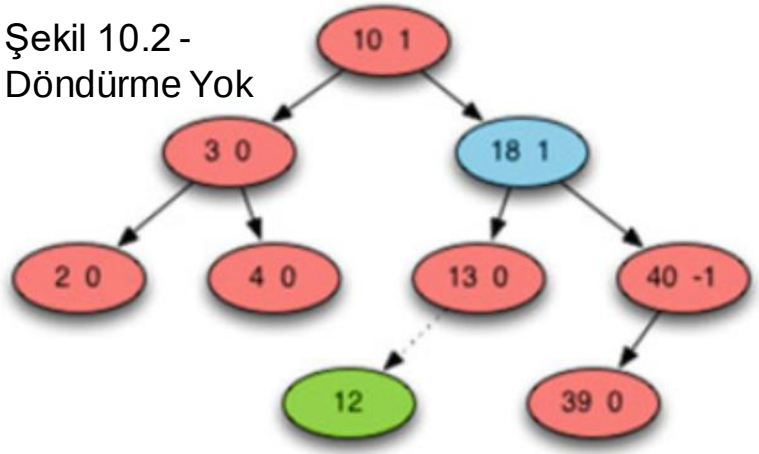
Diğer bir deyişle her bir düğümün dengesi(balance) 0'a eşittir.

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(path stack) kullanabilirsiniz.

Bu durum AVL ağacına 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(path stack) itilir. 39'u içeren yeni düğümün bakiyesi 0 olarak ayarlanır. 40'ı içeren düğümün yeni bakiyesi -1. 18 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 10.2 yeni değerin eklenmesiyle değişen ağacı göstermektedir.



Şekil 10.1 - Pivot Düğüm(node) yok



Şekil 10.2 - Döndürme Yok

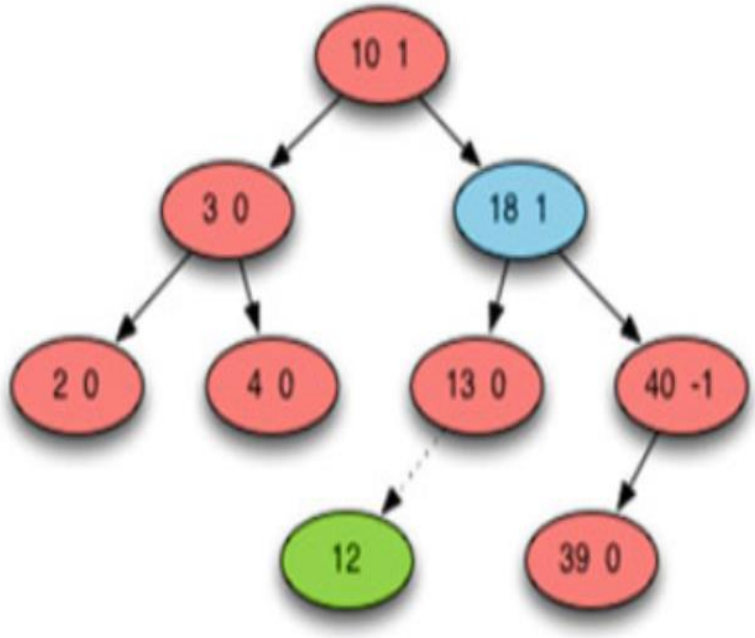
Durum 2

Pivot düğüm mevcut, dengeleri ayarla.

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. Bu, pivot düğümün üzerindeki düğümlerin dengeleri etkilenmez. Bu durum doğrudur çünkü pivot düğümde köklenen alt ağacın yüksekliği yeni düğümün eklenmesiyle değişmez.

Şekil 10.2 bu durumu göstermektedir. Anahtarı 12 olan öğe AVL ağacına 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 ağacı AVL ağacı olarak kalır. Pivota kadar olan düğümlerin dengesi ayarlanmalıdır. Pivotun üzerindeki dengelerin ayarlanması gerekmez çünkü bunlar etkilenmez. Şekil 10.3, ağaca 12 eklendikten sonra ağacın nasıl göründüğünü göstermektedir.

Şekil 10.2 - Döndürme Yok



Durum 3

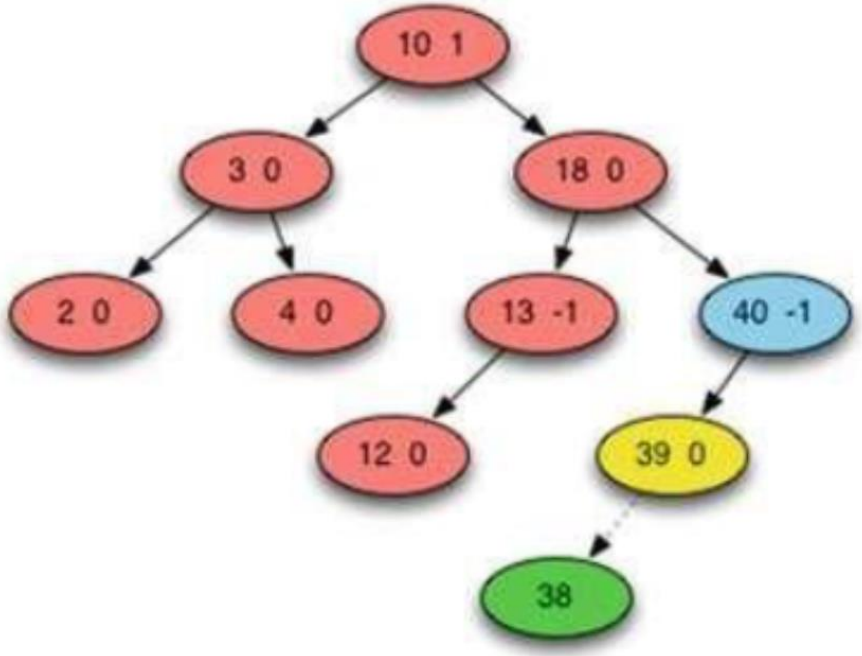
Pivot düğüm mevcuttur. Ancak bu kez yeni düğüm, pivotun daha büyük yükseklikteki alt ağacına (dengesizlik yönündeki alt ağaç) eklenir. Bu, yeni düğüm eklendikten sonra pivot düğümün -2 veya 2 dengesine sahip olmasına neden olur, bu nedenle ağaç artık bir AVL ağacı olmayacaktır. Burada, ağacı AVL durumuna geri getirmek için tek bir döndürme(single rotation) veya çift döndürme(double rotation) gerektiren iki alt durum vardır. Dengesizlik yönündeki pivot düğümün çocuğunu kötü çocuk(bad child) olarak adlandıracağız.

Alt Durum A: Tekli Rotasyon(Single Rotation)

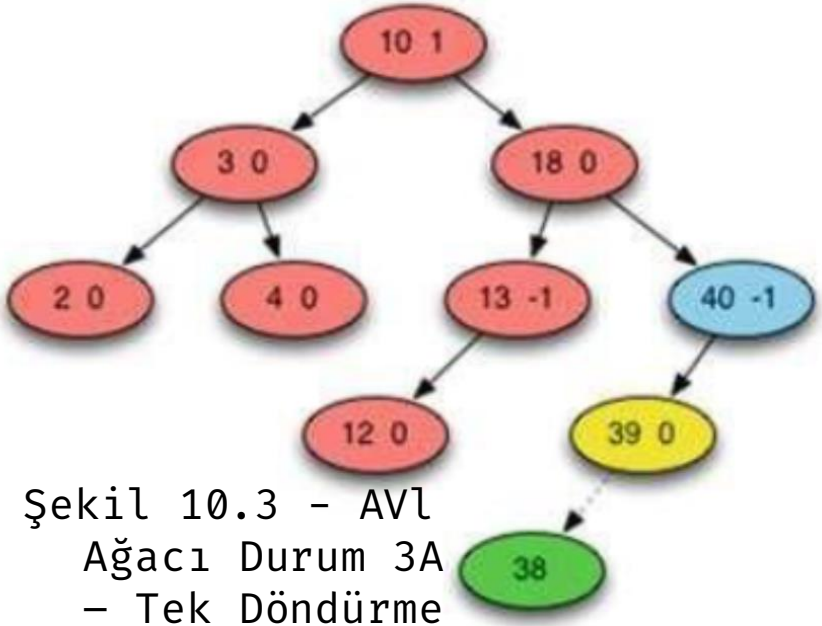
Bu alt durum, yeni düğümün dengesizlik yönünde olan kötü çocuğun alt ağacına eklendiğinde ortaya çıkar. Çözüm, dengesizliğin ters yönünde dönüş yapmaktır. Dönüşten sonra ağaç hala bir ikili arama ağacıdır(binairy search tree). Ayrıca, döndürme işlemi sonrasında döndürme düğümünde köklenen alt ağaç yeniden dengelenir, bu da genel yüksekliğini bir azaltır.

Şekil 10.3 bu alt durumu göstermektedir. 38 değeri, 39'u içeren düğümün solundaki ağaca eklenecektir. Değer 38'in, 39 içeren düğümün soluna eklenmesi gerekmektedir. Ancak bunu yapmak, 40’ı içeren düğümün dengesini -2'ye düşürür ve bu, dengesizlik içeren en yakın üst soydaki düğüm ve pivot düğümüdür. Sarı düğüm, kötü çocuktur(bad child).

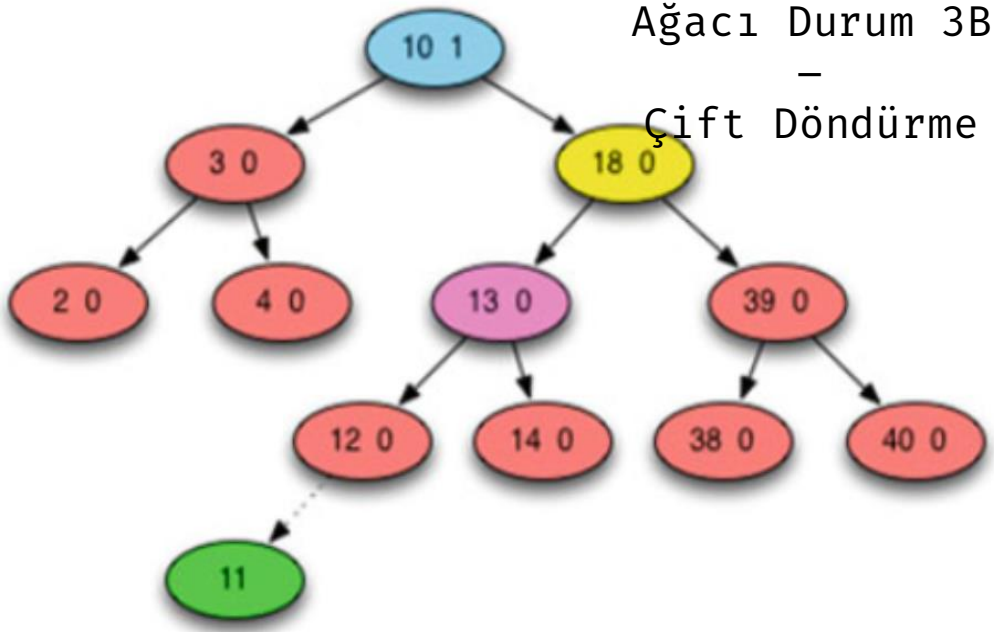
Şekil 10.3 - AVL Ağacı
Durum 3A – Tek Döndürme



Ek olarak, 38, dengesizliğin olduğu yönde eklenmektedir. Dengesizlik solda ve yeni değer solda eklenmektedir. Çözüm, 40'ta köklenen alt ağacı sağa döndürmektir, bu da Şekil 10.4'te gösterilen ağaca neden olur.



Şekil 10.3 - AVl Ağacı Durum 3A – Tek Döndürme

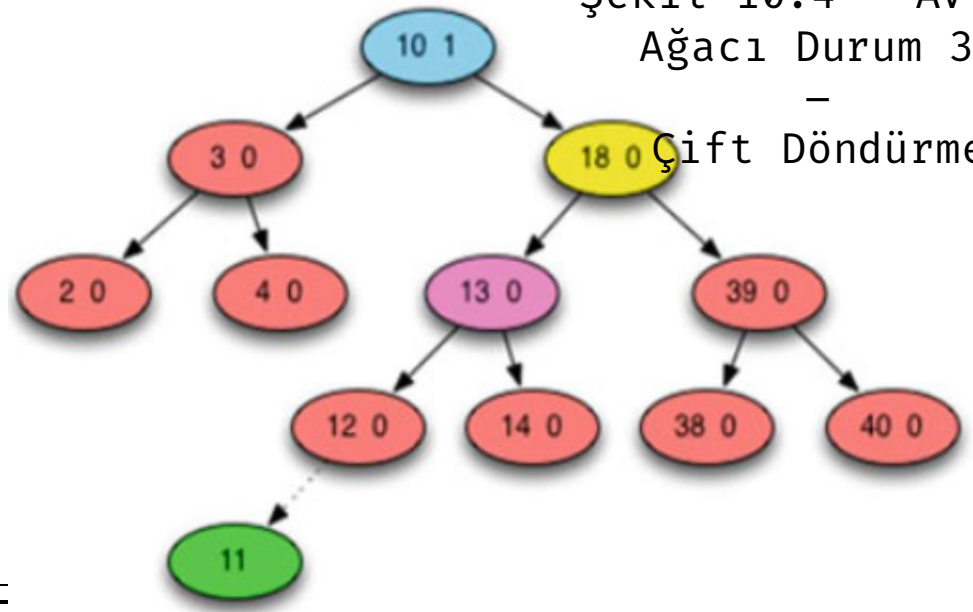


Şekil 10.4 - AVl Ağacı Durum 3B – Çift Döndürme

Alt Durum B: Çift Döndürme(Double Rotation). Bu alt durum, yeni düğümün dengesizliğin ters yönünde olan kötü çocuğun alt ağacına eklendiğinde ortaya çıkar. Bu alt durum için, kötü çocuğun arama yolu üzerinde bulunan çocuk düğümünü kötü torun(bad grandchild) olarak adlandırırız. 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.

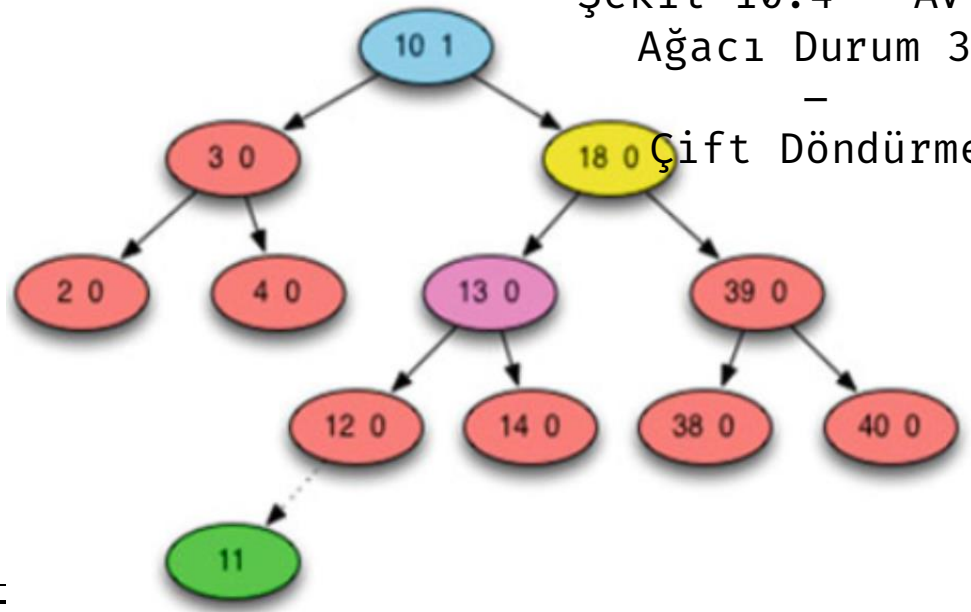


Şekil 10.4 - AVl Ağacı Durum 3B – Çift Döndürme

Yine, ağaç hala bir ikili arama ağacıdır ve orijinal pivot düğümün konumundaki alt ağacın yüksekliği çift döndürme ile değişmez. Şekil 10.4 bu durumu göstermektedir. Bu durumda pivot, ağacın köküdür.

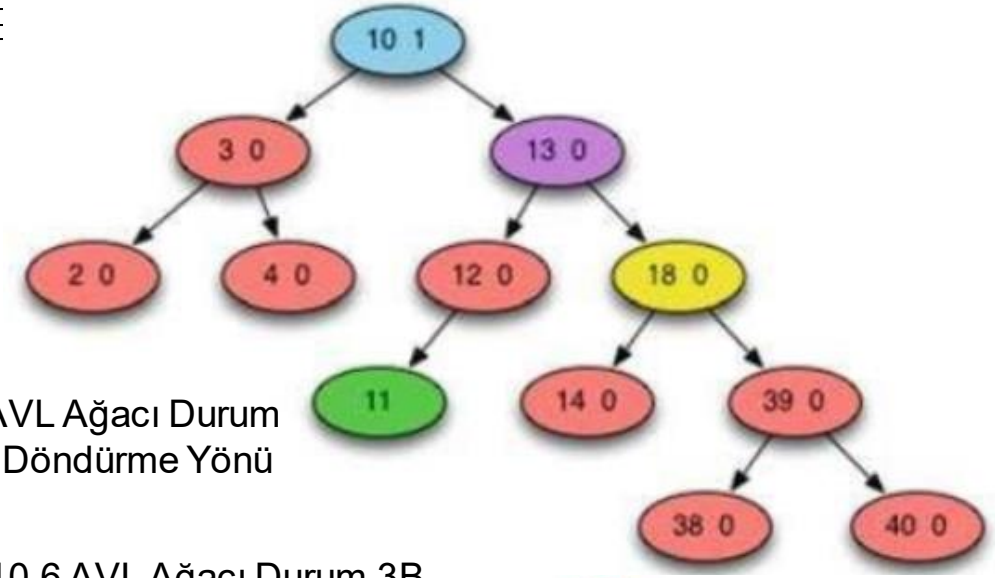
1. Kötü çocukta dengesizlik yönünde tek bir rotasyon gerçekleştirin.

2. Pivotta dengesizlikten uzağa doğru tek bir dönüş gerçekleştirin.

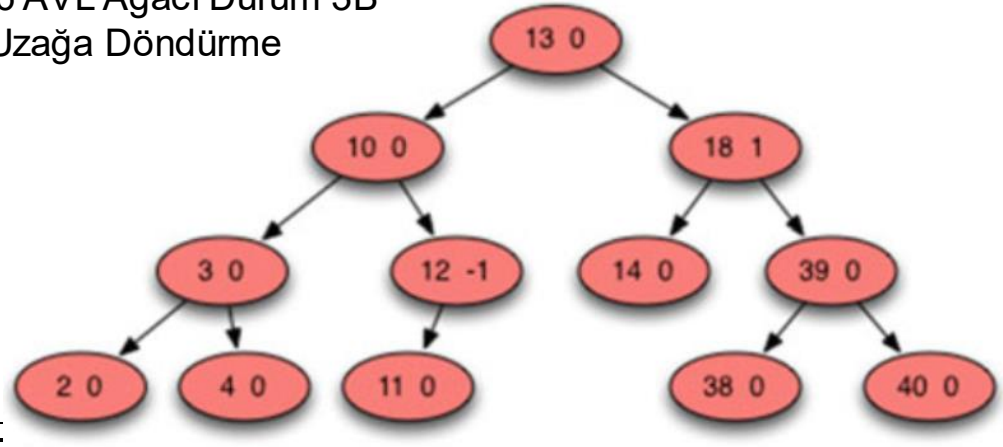


Şekil 10.4 - AVl Ağacı Durum 3B – Çift Döndürme

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 ağacın sağ tarafını dengelemeye bir şekilde yardımcı olur. Şekil 10.6'da gösterilen ikinci adım, pivotta sola dönerek tüm ağacı tekrar dengeye getirir.



Şekil 10.5 AVL Ağacı Durum 3B Adım 1, Döndürme Yönü



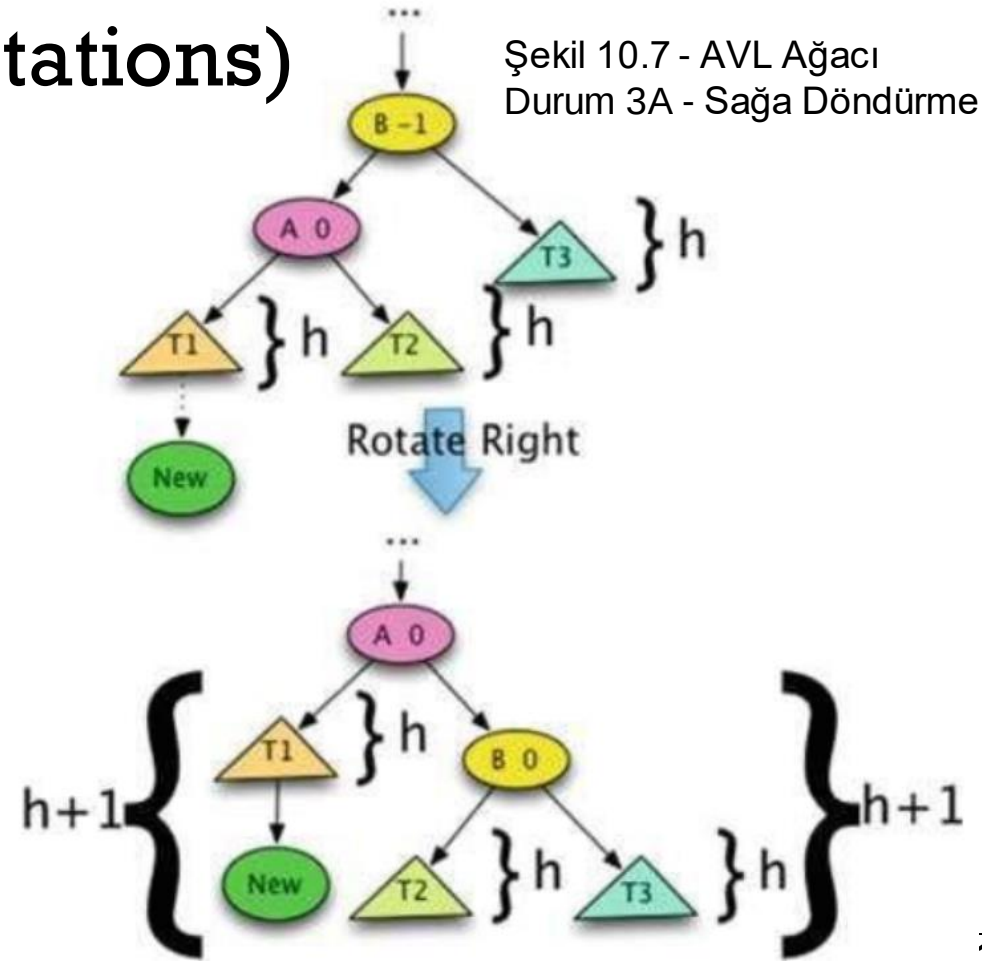
Şekil 10.6 AVL Ağacı Durum 3B Adım 2, Uzağa Döndürme

Bu algoritmanın en zor kısmı değerleri (balances) 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.

10.3.5 Rotasyonlar (Rotations)

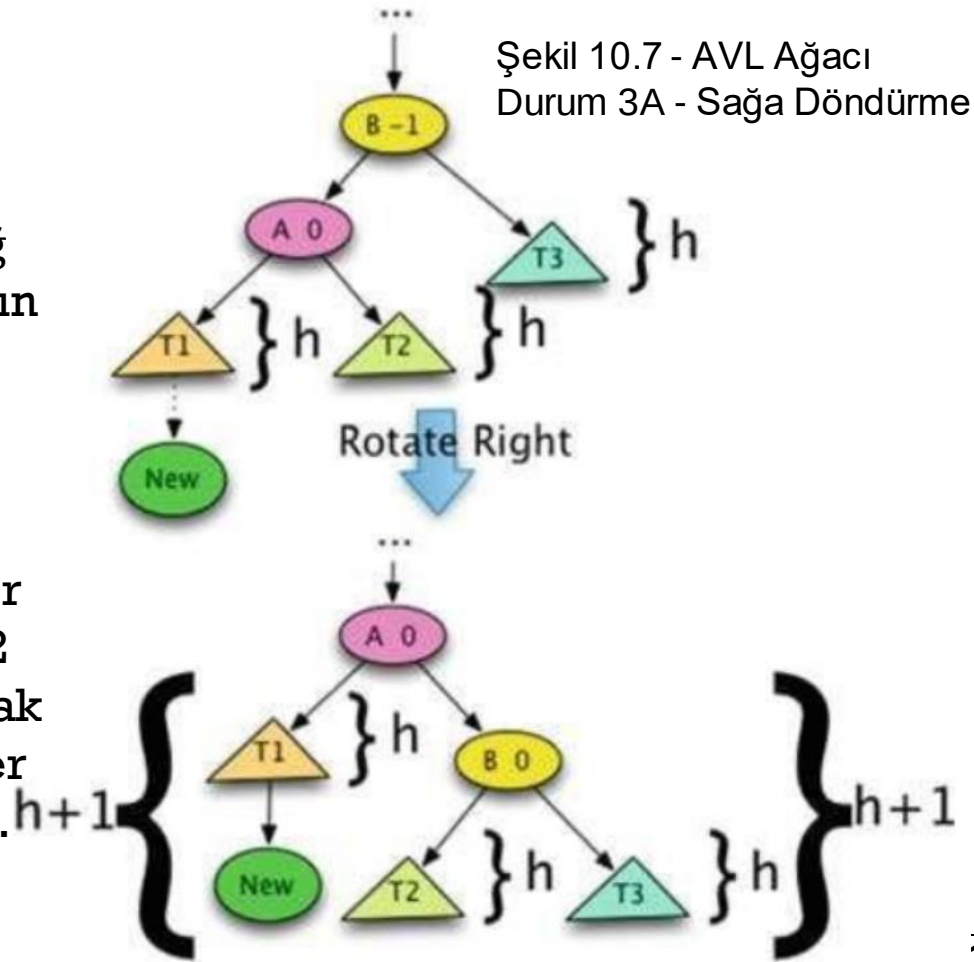
1. ve 2. durumlar da sadece bakiyeleri ayarlamak anlamına geldiği için basit bir şekilde uygulanabilir. "Durum 3 A" için ağaç, ağaca yeni bir düğümün ekleneceği bir durumdadır ve bu da ele alınması gereken bir dengesizliğe neden olur. İki ihtimal vardır Şekil 10.7 bu olası durumlardan ilkini göstermektedir. Yeni düğüm, pivot düğüme bağlı alt ağaç zaten sola ağırlıklıyken, kötü çocuk A'nın soluna eklenebilir.



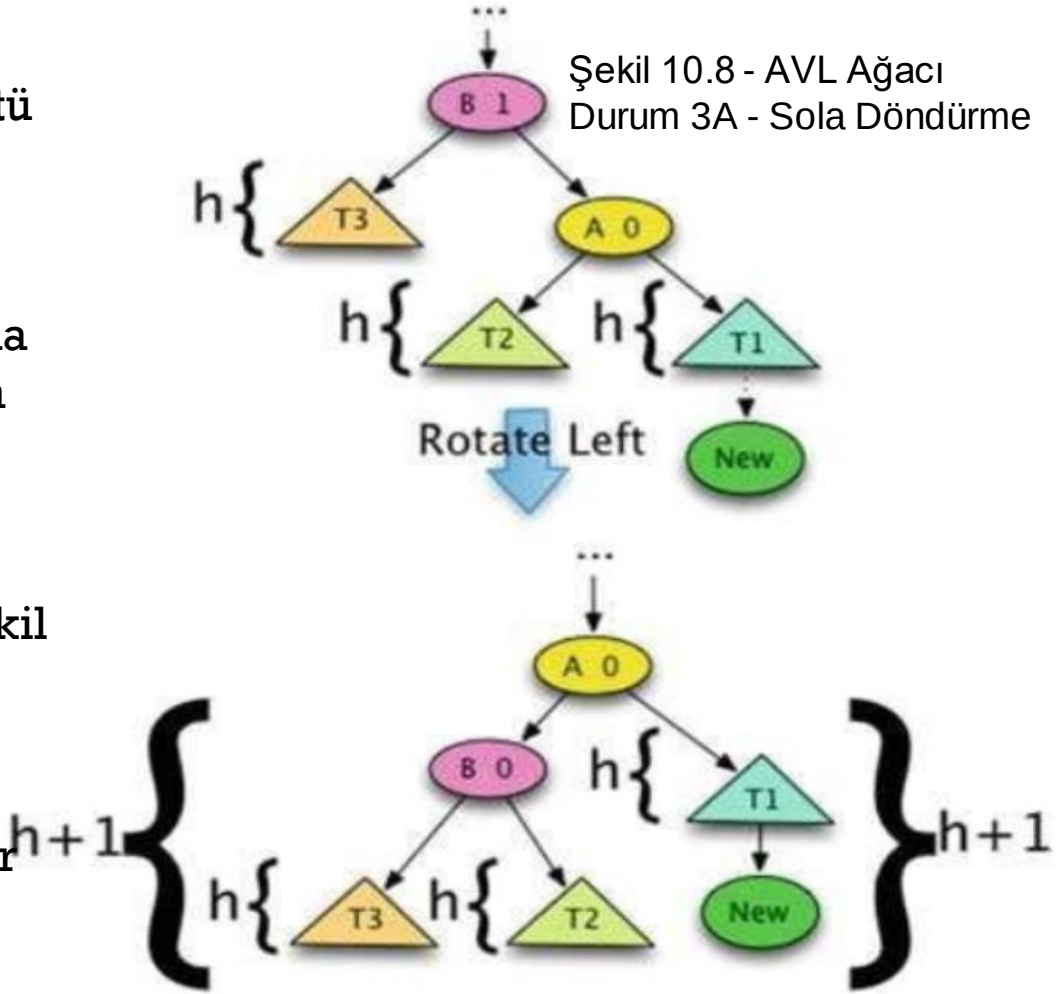
Şekil 10.7 - AVL Ağacı
Durum 3A - Sağa Döndürme

Pivot düğüm olan B, sıfır olmayan bir dengeye sahip en yakın atadır. B düğümünün yeni düğümü eklemeden önce -1 bakiyeye sahip olması için sağ alt ağacının yüksekliği ñ, sol alt ağacının yüksekliği ise h + 1 olmalıdır. Yeni düğümün kötü çocuğun alt ağacına eklenmesi, pivotun -2 bakiyeye sahip olmasına neden olur ki buna izin verilmez. Sağa döndürme sorunu çözer ve ikili arama ağacı özelliğini korur. T2 alt ağacı rotasyonda hareket eder ancak rotasyondan önce T2'deki tüm değerler B'den küçük ve A'dan büyük olmalıdır.

Şekil 10.7 - AVL Ağacı
Durum 3A - Sağa Döndürme

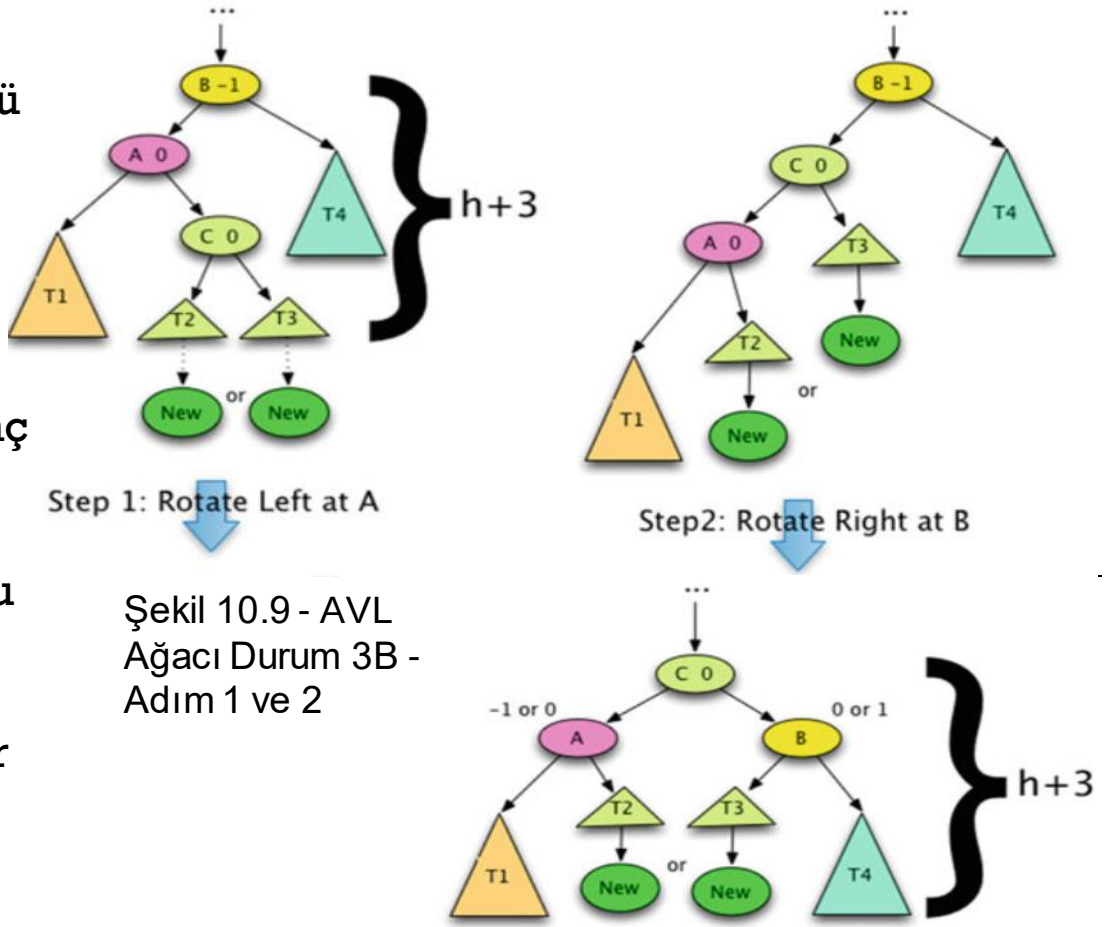


Dengesizlik sağda olduğunda kötü çocuğun sağına bir değer eklemek, sola döndürme gerektiren benzer bir durumla sonuçlanır. Her iki rotasyonda da A ve B düğümlerinin dengesinin sıfır olduğuna dikkat edin. Bu yalnızca 3 A durumu için geçerlidir ve çift döndürme durumunda geçerli değildir (Şekil 10.8). Yine, her iki düğümün, pivot ve kötü çocuğun dengesi, her iki yöndeki dönüşten sonra sıfır olur. Durum 3A başka hiçbir koşul altında mümkün değildir.

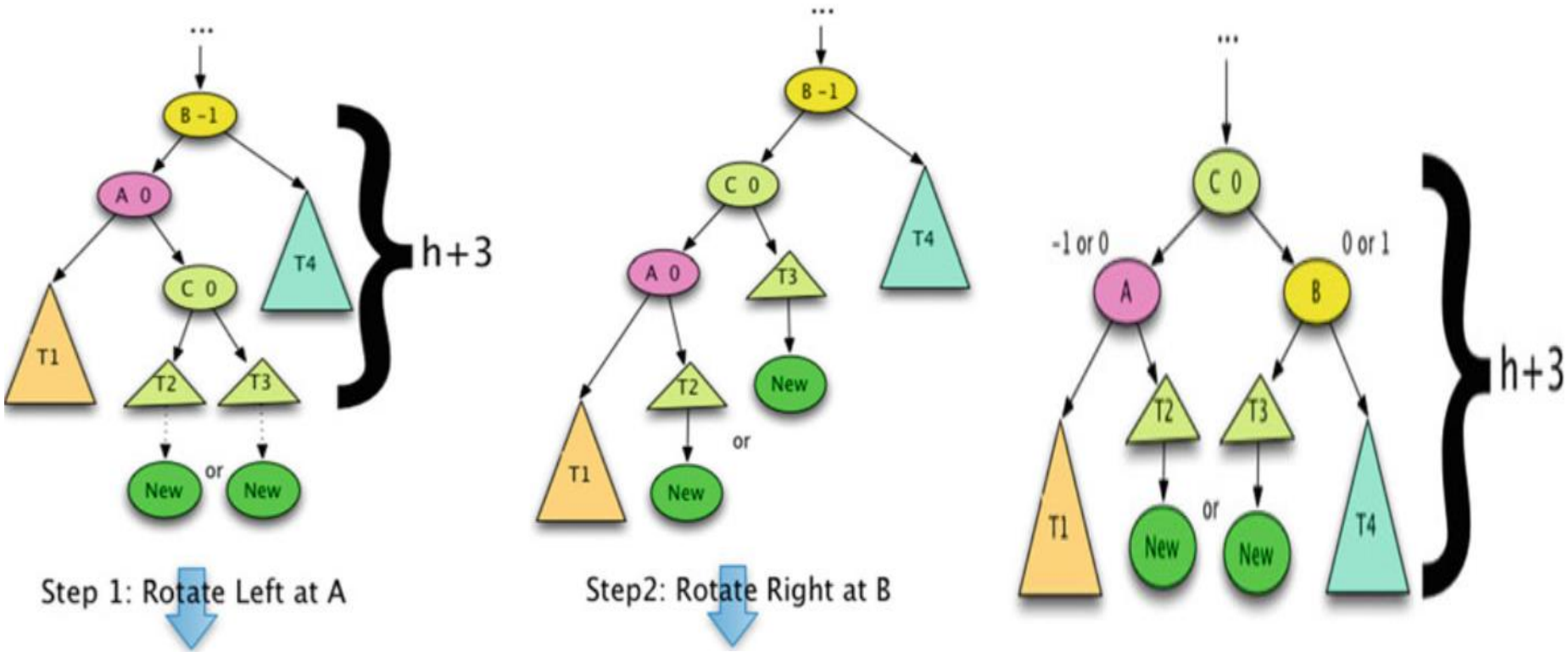


Şekil 10.8 - AVL Ağacı
Durum 3A - Sola Döndürme

Durum 3B için sadece bir pivot ve kötü çocukla değil, aynı zamanda kötü bir torunla da ilgilenmeliyiz. Önceki bölümde açıklandığı gibi, bu durum bir dengesizliğin tersi yönünde kötü bir çocuğun altına yeni bir değer ekler. Örneğin, Şekil 10.9'daki alt ağaç sola ağırlıklandırılmış ve yeni düğüm kötü çocuğun sağına eklenmiştir. Benzer bir durum, alt ağaç sağa doğru ağırlıklandırıldığında ve yeni düğüm kötü çocuğun sol alt ağacına yerleştirildiğinde meydana gelir. Her iki durum da gerçekleştiğinde, dengeyi yeniden sağlamak için çift döndürme gerekir.



Şekil 10.9 - AVL Ağacı Durum 3B - Adım 1 ve 2



Şekil 10.9 - AVL Ağacı Durum 3B - Adım 1 ve 2

10.3.6 AVL Ağacı Yinelemeli(Recursive) Ekleme

Özyinelemeli bir işlevi uygularken, bir sınıfın yöntemi yerine tek başına bir işlev olarak yazmak çok daha kolaydır. Bunun nedeni, tek başına bir yöntemin hiçbir şey (yani Python'da None) üzerinde çağrılabilmesiyken, bir yöntemin her zaman null(boş değer) olmayan bir self(kendini gösteren) referansına sahip olması gerektiğidir. Özyinelemeli fonksiyonları metot olarak yazmak self için özel durumlara yol açar.

Örneğin, insert yöntemi özyinelemeli olarak yazılırsa, özyinelemeli işlev olarak insert'i çağırırsa uygulaması daha kolaydır. Bölüm 10.2.1'deki insert fonksiyonu yüksekliği dengeli AVL ağaçları için yeterli olmayacaktır. Ekleme algoritması ağacın mevcut dengesini dikkate almalı ve önceki bölümde sunulan üç durumda tartıştığımız gibi dengeyi korumak için çalışmalıdır.

Sorular

Review Questions | İnceleme Soruları

4. Pivot düğüm(pivot node) nedir?

Pivot düğüm, AVL ağaçlarında dengeyi yeniden sağlamak için kullanılan bir tekniktir; bir düğümün alt ağaçları arasındaki yükseklik farkı belirli bir sınırı aştığında, bu düğüm pivot olarak belirlenir ve sola veya sağa doğru döndürme işlemi uygulanarak denge sağlanır, böylece ağaçın performansı korunur.

Sorular

Review Questions | İnceleme Soruları

5. AVL ağaçlarıyla ilişkili olarak kötü çocuk(bad child) nedir?

"Bad Child", AVL ağaçlarında dengeyi sağlamak için kullanılan bir terimdir; bir düğümün alt ağaçlarından birinin yüksekliği diğerinden 2 birim fazla olduğunda, bu "kötü çocuk/bad child" olarak adlandırılır ve rotasyon işlemlerinde öncelikli olarak ele alınır.

Sorular

Review Questions | İnceleme Soruları

6. Yol yığını(path stack) nedir ve ne zaman gereklidir?

Yığın, bir düğümü bir ata veya torunlarına kadar izleyen bir veri yapısıdır. Bir düğüm dengesiz hale geldiğinde, yığın, dengesizlik nedeniyle etkilenen düğümleri saklamak için kullanılır. Bu yığın, AVL ağacındaki dengesizlikleri gidermek için gerekli olan rotasyonları yapmak için gereklidir.

Özetle yığın, bir dengesizlik algılandığında, bu dengesizlikten etkilenen düğümleri takip ederek işlem yapmayı kolaylaştırır. Bu nedenle önemli bir veri yapısıdır.

Sorular

Programming Problems | Programlama Problemleri

3. Her düğümde yükseklikleri koruyan ve özyinelemeli(recursively) olarak ekleme yapan bir AVL ağacı uygulaması yazın. Programınızı rastgele oluşturulmuş bazı veriler üzerinde kapsamlı bir şekilde test etmek için bir test programı yazın.

progsoru-3.py

Sorular

Programming Problems | Programlama Problemleri

4. Her düğümde yükseklikleri koruyan ve yinelemeli(iteratively) olarak ekleme yapan bir AVL ağacı uygulaması yazın. Programınızı rastgele oluşturulmuş bazı veriler üzerinde kapsamlı bir şekilde test etmek için bir test programı yazın.

progsoru-4.py

Programming Problems | Programlama Problemleri

Soru 3 ve 4 arasındaki fark:

İteratif (yineleyici) yaklaşım, bir döngü kullanarak bir işlemi tekrarlayarak gerçekleştirirken, rekürsif (özyinelemeli) yaklaşım, bir işlemi kendini çağırarak tekrarlar.

Dinlediğiniz İçin Teşekkürler

Himmet Can Umutlu
2023481048@cumhuriyet.edu.tr