Hakkında

Program BitBucket Git sistemi üzerinde yazılmış ve toplamda 12 sürümden meydana gelmiştir. Ödev dokümanında istenilen veri yapılarını ve sıralama işlemlerini gerçekleştirmektedir. Görsel ara yüzde, kuyrukları listelemek gibi birkaç küçük ek özellik daha sunulmak ile beraber çok fazla bir değişiklik yapılmamıştır.

Giriş

Programda 5 farklı sınıf bulunmaktadır; *Dosya*, *Dugum*, *Hata*, *IkiliAramaAgaci*, *Kuyruk.*

Program başladığında ilk olarak "*sayilar.txt*" adlı dosya okunur ve her satırı bir kuyruğa eklenir. Bu dosya adı bir sabitte tutulmuştur, çalışma esnasında kullanıcıdan alınmaz. Bunun nedeni ödev dokümanında istenilen dosya adının belirli olmasıdır. Sonrasında bu dosya adına göre ikili arama ağacı yapısı üzerinde bu veriler saklanır ve listeleme işlemi yapılmıştır.

Sayıların Dosyadan Okunması

Dosyalama işlemleri için, *Dosya* adında yeni bir sınıf oluşturulmuştur. Bu sınıftan oluşturulan bir nesne parametre olarak dosya adını alır. Nesne oluşturulduğunda dosyadan okuma işlemini gerçekleştirir ve sonrasında içeriğini kendi özel bir verisi olan *String* tipinde değişkende barındırır.

Dosyadan Okunan Sayıların Kuyruklara Aktarılması

Daha önceden bu dosya nesnenin *agacaAktar* metoduna parametre olarak gönderilen bir ağaç nesnesine bu dosya kuyruk yapılarına dönüştürülerek saklanır. Bu metot aynı zamanda bir kuyruk dizisinin adresini alır. Bunu nedeni dosyadan alınan sayıların farklı farklı kuyruklara eklenmesi gerekmesidir. Dışarıdan gönderilen nesnenin bellekten kaldırılması işlemi Ağaç yapısının sorumluluğunda olmadığından, eklenecek olan kuyruk nesneleri kullanıcı tarafından kontrol edilmelidir.

Oluşturulan Kuyrukların Ağaca Aktarılması

agacaAktar metodu kendi içinde *Dosya* sınıfının özel bir metodu olan *kuyrugaAktar* metodunu kullanarak dosyadan okunan sayıları tek tek kuyruk yapılarına aktarır. Sonrasında bu şekilde oluşturulan her bir kuyruk parametre olarak gelen ağaç nesnesine tek tek eklenir.

İkili Arama Ağacı

Düğüm

İkili arama ağacı için tasarlanmış olan düğüm yapısı standart bir bağıl listeden farklı olarak kendinden sonra gelen iki düğümün, sağ ve sol düğümlerin adreslerini tutar. Bu şekilde mantıksal bir dallanma meydana getirilir. Düğümler kuyruk yapısı barındıracak şekilde özelleştirilmiştir. Yine de *Kuyruk* sınıfına dair özel bir metot veya veri barındırmaz, yalnızca nesne tutma görevini yerine getirir. Bu sayede *Düğüm* yapısı kolaylıkla şablon haline getirilecek şekilde değiştirilebilir ve genel amaçlı olarak kullanılması sağlanabilir.

Ekleme İşlemi

Ekleme işlemi için aşırı yüklenmiş *elemanEkle* metodu kullanılır. Bunlardan birisi *public* alanda yer alan ve eklenecek olan *Kuyruk* nesnesini parametre olarak alan metottur. Dolayısıyla kullanıcıya sunulacak olan metot budur. Diğer *elemanEkle* metodu ise *private* alandadır ve *özyinelemeli* olarak çalışır. Yaptığı işlem sağ ve sol dallardan birisi boş oluncaya kadar ilerlemek ve bulduğu boşluğa yeni elemana eklemektir. Burada dikkat edilmesi gereken kriter her seferinde sağ dalın, o anki düğümden daha büyük, sol dalın ise daha küçük olması gerekmesidir. Bunu sağlamak için düğümlerde saklanan nesnenin karşılaştırma işlemini destekliyor olması gerekir. Bu projede ikili arama ağacı *Kuyruk* veri yapısı barındıran *Düğüm*ler ile çalışacak şekilde özelleştirilmiş olduğundan karşılaştırma işleminin desteklenmesi garanti edilmiş olur.

Silme işlemi

Bu projede silme işlemi kullanılmayacağından detaylı olarak anlatılmayacaktır. Yine de projede bunu sağlayan metot dahil edilmiştir. Aynı ekleme işleminde olduğu gibi aşırı yüklenmiş iki adet *elemanSil* metodu bulunur. Eleman silme işlemi için öncelikle verilen düğümün sağ dalından sola doğru veya sol dalından sağa doğru ilerlenmeli

ve sona ulaşılmalıdır. Bu sayede büyüklük derecesine göre verilen düğümdeki veriden bir büyük veya bir küçük veriye ulaşılmış olunur. Bu projede sağ dalın sol ucuna doğru gidilerek bu işlem gerçekleştirilmiştir.

Sonrasında ulaşılan düğümün alt dalları var mı diye bakılır. Hiç alt dalı olmaması, tek bir alt dalı olması ve iki tane alt dalı olması için ayrı ayrı işlemler yapılır. Aynı zamanda bu düğümün kök düğüm olup olmadığı da ayrıca denetlenir.

Listeleme / Gezme İşlemi

Her sıralama işlemi aşırı yüklenmiş iki adet metottan meydana gelir. Bunlar; *sirali*, *onSirali*, *sonSirali* metotlarıdır. Tıpkı ekleme ve silme işlemlerinde olduğu gibi bunlardan her biri *private* diğeri *public* alandadır. Kullanıcıya açık olan metot hiçbir parametre almaz ve geriye elemanların sıralanmış halini barındıran bir *String* döndürür. Bu projede barındırılan veriler *Kuyruk* tipinde özelleştirilmiştir. Dokümanda istenildiği gibi her bir kuyruktaki rakamların toplamı sıralanır. Bunun için her bir kuyruğun *rakamlarToplami* metodu ile toplamı elde edilir. Bu yönüyle tasarlanmış olan ağaç yapısı *Kuyruk* sınıfına bağımlı bir veri yapısıdır.

Sıralı Listeleme (inorder)

Sıralı listeleme işleminde her bir düğümün önce sol dalı ile özyineleme yapılır, sonrasında düğüm listeye aktarılır ve sağ dal ile özyineleme yapılır. Bu sayede sol yapraktan sağ yaprağa doğru sıralı bir şekilde gezilmiş listeleme işlemi yapılmış olunur.

1. Sol
2. Listeye aktar
3. Sağ

Ön Sıralı Listeleme (preorder)

Ön sıralı listeleme işleminde ilk olarak düğüm listeye aktarılır sonra sırasıyla sol ve sağ düğümler üzerinden özyineleme yapılır. Bu sayede düğümden sol dala ağırlıklı bir dolaşım ile listeleme işlemi yapılmış olunur.

1. Listeye aktar
2. Sol
3. Sağ

Son Sıralı Listeleme (postorder)

Son sıralı listeleme işleminde ise ilk olarak sol düğüm, sonra sağ düğüm üzerinde öz yineleme yapılır ve en son düğüm listeye aktarılır. Bu sayede sol daldan sağ dala ağırlıklı bir dolaşım ile listeleme işlemi yapılmış olunur.

1. Sol
2. Sağ
3. Listeye Aktar

Sonuç

Dosyadan oluşturulan kuyruklar ekrana listelenir, bu ödev dokümanında belirtilmemiş olan ek bir özellik olarak dahil edilmiştir. Sonrasında ağaç üzerinden her üç listeleme metoduyla da ekrana, ağacın düğümlerinin barındırdığı **kuyruk**larda yer alan rakamların toplamları ilgili şekilde yazdırılır.

11.12.2016
Emirhan Aydın
G1412.10018