Ömer Can ÇALIŞIR - Melih Ensar BARIŞIK

[redacted] 1A - [redacted] 1A

Veri Yapıları 3. Ödev Raporu

Öncelikle ödev dökümanında da açıklandığı gibi program çalıştırıldığında 2 farklı dosya okunacaktır. Okunan bu dosyaların içerisinde rastgele sayılar bulunmaktadır. Dosyalardan sayıları okuyacak olan program iki farklı arama ağacına bu sayıları yerleştirecektir. Ardından dökümanda kuralları belirtilen oyun oynatılacaktır.

İşlemleri yapmak üzere tanımlanacak sınıfların; Node sınıfı, BinaryTree sınıfı ve Oyun sınıfı olmasına karar verdik.

Tasarımımızın uygulamasına Node sınıfını oluşturarak başladık. Node sınıfını oluştururken, daha sonra ikili arama ağacında kullanılacakları için bunu göz önünde bulundurarak; sağ düğümü tutan bir right değişkeni, sol düğümü tutan bir left değişkeni, sayıları tutacak bir data değişkeni ve her bir düğümün soy sayısını tutacak olan soySayisi değişkeni oluşturduk. Node sınıfının tasarımı böylelikle bitmiş oldu.

Sonrasında Benim.txt ve Rakip.txt dosyalarından okunacak sayıların tutulacağı yapı olan ikili arama ağacı(BinarySearchTree) sınıfımızı oluşturmaya başladık. İlk olarak ikili arama ağacı yapımıza düğüm ekleyebilmek adına recursive çalışan bir insert(ekleme) fonksiyonu yazdık. Fonksiyon basitçe kendisine parametre olarak verilen datayı ikili arama ağacının özelliği olan küçük veriler sola, büyük veriler sağa kuralına uygun olarak eklemeye çalışıyor. Öncelikle fonksiyon ikili arama ağacı yapısının boş olup olmadığını kontrol ediyor. Boş ise köke yeni bir eleman ekliyor. Şayet doluysa kurallar işletilmeye başlanıyor. Öncelikle verilen data ile yapıdaki mevcut data karşılaştırılıyor. Eğer verilen data mevcut datadan büyükse bu seferde sağ düğümün boş olup olmadığına bakılıyor. Boşsa eleman ekleniyor degilse fonksiyon kendisini mevcut düğümün sağını parametre alarak tekrar çağırıyor. Bu işlem recursive olarak uygun konum bulunana kadar devam ediyor. Aynı prosedür verilen data mevcut datadan küçük olduğu durumda da işletiliyor. Ardından ağaç üzerindeki en büyük değeri bulabilmek adına maxValue isimli bir recursive fonksiyon yazdık. Kendisine parametre olarak verilen düğümün sağı boş olana kadar kendisini çağırıyor ve en sağ konuma ulaştığında o konumdaki datayı geri döndürüyor. Sonrasında her bir düğümün datasını ve soy sayısını bastırabilmek adına post order sıralamasını kullanan recursive bir postOrder

fonksiyonu yazdık. Fonksiyon sırasıyla left, right ve kök düğümleri üzerinde hareket ederek bunların datalarını ve soy sayılarını bastırıyor. Bu fonksiyonun ardından her bir düğümün soy sayisini hesaplamak adına verilenDugumunSoySayisiniHesapla isimli recursive bir fonksiyon yazdık. Fonksiyon basitçe verilen düğümden başlayarak alt düğümlere gidiyor ve verilen düğümün soy sayisini hesaplıyor. Ardından tüm ağacın soy sayısını hesaplamak adına, bütün ağaç üzerindeki düğümleri, oluşturduğumuz verilenDugumunSoySayisiniHesapla fonksiyonuna yollayacak ve dönen değerleri toplayacak bir fonksiyon yazdık. Böylelikle ağacın toplam soy sayısını bulan fonksiyonu tamamladık. Bu fonksiyondan sonra istenen düğümü parametre olarak alan ve silen bir fonksiyon yazmaya basladık. Bu fonksiyon öncelikle solda çocuk yoksa sağ düğümü silinen düğüm yerine, sağda çocuk yoksa da soldaki düğümü silinen düğüm yerine koyacak şekilde çalışıyor. Bu iki durumun sağlanmadığı 2 çocuklu olma durumunda ise solun en sağına gitme kuralı gözetilerek gerekli işlemler yapılıyor. Tüm bu işlemlerin ardından düğüm ağactan siliniyor. Daha sonra AraVeSil fonksiyonunu yazmaya başladık. Bu fonksiyon istenen değerin ağac üzerinde aranması ve sil fonksiyonu yardımıyla silinmesi şeklinde çalışıyor. Basitçe aranan değeri yakaladığı durumlarda bu değerin bulunduğu düğümü sil fonksiyonuna yolluyor ve silme işlemini gerçekleştiriyor. Aranan değerin mevcut düğüm değerine eşit olmadığı durumlardaysa iki durum değerlendiriliyor. Aranan değer mevcut değerden küçükse fonksiyon kendini mevcut düğümün solunu parametre alarak yeniden çağırıyor, tersi durumdaysa mevcut düğümün sağını parametre alarak yeniden çağırıyor ve recursive şekilde aranan değer bulunana dek işlem tekrar ediliyor. Bu fonksiyonla birlikte ikili arama ağacı(BinarySearchTree) sınıfımızın tasarımı bitmiş oldu.

Son olarak Oyun sınıfını oluşturmaya başladık. Oyun sınıfında ağaçların sahip olduğu değerleri okuyan bir degerleriOku fonksiyonu, ekranda gördüğümüz çıktıları oluşturan bir yazdır fonksiyonu, karşılaştırma sonucunda oluşan kazanma kaybetme ya da berabere kalma olaylarına karar veren kimKazandi fonksiyonu, oyunun her bir elinin oynandığı oyunuBaslat fonksiyonu ve oyunuBaslat fonksiyonunu kendi içinde çağıran gerekli skorlara ulaşana kadar oyunuBaslat fonksiyonunu sürekli çağıran bir oyunuOynat fonksiyonu tanımladık.