Veri Yapıları Ödevi Raporu

1. Giriş

Bu rapor, veri yapıları dersi kapsamında gerçekleştirdiğimiz ödevin detaylarını, zorlukları, ödevde yaptıkla rımızı ve öğrendiklerimizi içermektedir. Ödev, AVL ağacı ve yığıt kullanarak bir C++ programı yazmamızı i çermekteydi. Bu program, belirli operasyonları gerçekleştirebilen bir veri yapısını oluşturmayı amaçlamakt adır.

2. Ödevin Tanımı

Öncelikle, ödevde istenilen başlıca özellikleri hatırlayalım. Programımız, AVL ağacı kullanarak Veri.txt dos yasından okuduğu her satır elemanları sıralı bir şekilde depolamalı ve ve depolanan her bir avl ağacındaki yaprakları o AVL ağacıyla bağlantılı olan yığıta atmalıdır. AVL ağaçlarındaki yaprak haricinde kalan düğü mler de ayrıca sayısal olarak toplanıp ascii değerler elde edilmeli ve bu ascii değerlerin char karşılıkları ek rana yazdırılmalıdır. Bunlara ek olarak yığıtlardaki
yaprak değerleri birbirleriyle kıyaslanıp bir en küçük bir en büyük olacak şekilde yığıtlardan çıkarılmalıdır. Bu process devam ederken ilk boşalan yığıta ait olan AVL ağacı silinmeli ve yığıtlar aynı verilerle yeniden doldurulup tekrar aynı işlemler yapılmalıdır. Bu süreç bir AVL ağacı kalana kadar devam etmelidir.
Çıktı olarak bu son kalan AVL ağacındaki ascii'den dönüştürülen char değeri ve bu AVL ağacının kaçıncı satır ait verileri taşıdığı ekrana belli bir formatta yazdırılmalıdır.

3. Zorluklar ve Çözümler

Ödevi gerçekleştirirken karşılaştığım zorluklardan biri, nasıl bir süreçle yoğıt ve AVL ağacını bağlayacağı mı düşünme aşamasıydı. Ben AVL ağacı içerisinde yığıtı tutmayı tercih ettim çünkü eğer AVL ağacı dışın da bir yığıtta tutsaydım bu druumda bunlarım birbirlerine bağlı olduğunu belirtecek dict yapısında yeni bir t ype oluşturmam gerekecekti ben bunun daha uzun ve zorlayıcı olacağını düşündüğümden direkt AVL ağa cı içerisinde yığıt tutmayı tercih ettim.
İkinci zorlayıcı durum ise boşaltılan yığıtların tekrar eski hallerine dönüştürülmesiydi, bunu da AVL ağacı i çerisinde biri real diğeri temp olmak üzere iki yığıt tanımlayarak çözdüm. İşlem şu şekilde gerçekleşiyor; real yığıttan bir veri çıkartıldığında temp yığıta ekleniyor ve eğer real yığıt tamamen boşalmadıysa ve refill fonksiyonu çalıştırıldıysa bu durumda temp yığıta atılan elemanlar geri real yığıta ekleniyor. LIFO kullandı ğı için sırada da bir karışma yaşanmıyor.
Üçüncü karşılaştığım sorun ise .hpp ve .cpp dosyalarını birbirinden ayırma ve makefile oluşturma sorunuy du. Bunu da birçok deneme sonucunda gerçekleştirmeyi başardım. Bu konuda internetten hataların ne old uğunu araştırarak ve tekrar düzelterek ilerledim.
Dördüncü zorluk ise şablon kullanmadan bu veri yapılarını oluşturmak ve kullanmak oldu. Bunu deneyimle mek de bana çok şey kattı.

4. Öğrenilenler ve Deneyimler

Öğrendiğim şeyler AVL ağacının ve Yığıt'ın çalışma mantığını daha iyi kavramış ve deneyimlemiş oldum. Her ne kadar derslerden öğrenmiş olduğumu düşünsem de kullanırken yaşadığım problemleri çözerken n asıl çalıştıklarına dair daha net fikirlerim oluştu.
Makefile oluşturma .hpp ve .cpp doyaları ayırma konusunda çok şey öğrendim. Daha öncesinde sadece . hpp yazıp çalıştırıyordum ve ayırmayı tam olarak bilmiyordum.

5. Sonuç

Veri yapıları dersi kapsamında gerçekleştirdiğimiz bu ödev, teorik bilgilerin pratiğe dökülmesi açısından ol dukça değerli bir deneyimdi. AVL ağacı ve yığıt gibi temel veri yapılarına dair derinlemesine bir anlayış ka zandım ve C++ dilinde daha yetkin bir şekilde program yazma becerilerimi geliştirdim. Bu öğrenim süreci, gelecekte karşılaşacağım benzer problemleri çözmek adına önemli bir temel oluşturdu.