
**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

# VERİ YAPILARI ÖDEVİ

**Öğrenci Adı: Mehmet**

**Öğrenci Soyadı: ATAŞ**

**Öğrenci Numarası: G201210382**

**Öğrenci E-Posta: mehmet.atas5@ogr.sakarya.edu.tr**

**Grup: 2/A**

**Öğretim Görevlisi: Dr. Muhammed Fatih ADAK**

Ödevde ilk olarak “DogruKuyrugu.hpp” adında bir dosya oluşturdum. Bu dosyada ödevde kullandığım, öncelikli kuyruk olan “DogruKuyrugu” sınıfını tanımladım ve metotlarını yazdım. Bu sınıfa genel olarak kurucu, yıkıcı, ekleme, silme, çıkarma gibi metotları ve öncelikli çıkarma, öncelikli silme metotlarını ekledim. Ödevde istenen öncelikli yapıyı kuyruğa ekleme yaparken değil, kuyruktan çıkarma işlemi yapılırken sağladım. Bu şekilde çıkarma ve silme işlemi yapılırken en küçük orijin değerine sahip düğümün verisi öncelikli olmuş oldu. Bu dosyayı oluştururken derste öğrendiğim öncelikli kuyruk veri yapısından yararlandım. Şablon veri yapısı kullanmadım, oluşturduğum sınıfı başlık ve kaynak dosyasına ayırdım. Daha sonra arama ağacı (AVL) işlemlerini sağlayacak olan “AVL.hpp” dosyasını oluşturdum, bunun içerisinde sınıfları ve fonksiyonları oluşturdum.

“AVL” sınıfına kurucu, yıkıcı, ekleme, temizleme, ağacı postorder dolaşma, ağacın yüksekliğini bulma, ağacı dengeleme işlemlerini yapacak fonksiyonları tanımladım. Ekleme yapılırken dengeleme işlemleri doğru kuyruğunun toplam uzunluğuna göre yapıldı ve dengeleme sağlandı. Daha sonra dengelenen ağacın her düğümünü postorder olarak dolaşarak her düğümdeki “DogruKuyrugu” nesnelerini ekrana çıkartmamı sağlayacak olan “postorder” metodunu oluşturdum. Bu sınıfı oluştururken de şablon veri yapısından faydalanmadım. Oluşturduğum sınıfı “AVL.hpp” başlık dosyası ve “AVL.cpp” kaynak dosyasına ayırdım.

Bu işlemlerden sonra dosyadan veri okuma işlemini gerçekleştirmek için “DosyaOku” adında bir sınıf tanımladım. Bu sınıfı da “DosyaOku.cpp” kaynak dosyası ve “DosyaOku.hpp” başlık dosyasına ayırdım. “DosyaOku” sınıfı içerisinde bulunan “NoktaOlustur” fonksiyonu bir noktanın(x,y,z) orijine olan uzaklık değerini bulur, “dosyaOku” fonksiyonu  ise dosyayı satır satır okur ve bu satırları işlemlere sokar.

→Bu işlemler:

1)İlk olarak satırdaki sayı adedi bulunur.

2)Bulunan sayı adedi uzunluğunda bir dizi oluşturulur.

3)Oluşturulan diziye satırdaki sayılar atılır.

4)For döngüsü kullanılarak satırdaki noktalar arası toplam uzunluk bulunur.

5)Başka bir for döngüsü yardımıyla her bir noktanın orijine olan uzaklık değeri bulunur ve bu değerler kuyruk nesnesinin düğümlerine toplam uzunluk değeriyle birlikte atılır.

6)Oluşturulan kuyruk nesneleri AVL ağacına eklenir.

Her satır için bu işlemler gerçekleştirilir ve bir AVL ağacı oluşturulmuş olur. Bu kısım tamamlandıktan sonra AVL ağacı “postorder” olarak dolaşılır ve ağacın düğümlerinde bulunan “DogruKuyrugu” nesnelerindeki orijin değerleri öncelikli olarak (orijin değeri küçükten büyüğe doğru öncelikli) ekrana yazdırılır ve uygulama son bulur.

Ödevdeki öncelikli kuyruk ve AVL ağacını, genel olarak derste oluşturduğumuz öncelikli kuyruk ve AVL ağacı sınıflarında gerekli düzenlemeleri ve eklemeleri yaparak oluşturdum. Dosya okuma kısmını ise adım adım, hatalar alıp aldığım hataları düzelterek tamamladım. Bu ödev öncelikli kuyruk yapısını, AVL ağacını (genel olarak arama ağaçlarını) ve dosya okuma işlemlerini daha iyi kavramamı sağladı.