Veri yapıları Ödev 5 Raporu

Yazarlar: İsmet GÜZELGÜN(b121210025@sakarya.edu.tr)
İbrahim AKDAĞ(b121210027@sakarya.edu.tr)

Özet

Ödevde C++ programlama diliyle, pointerlar yardımıyla nesneye dayalı programlama paradigmasına uygun bir şekilde huffman agaci ve hash tablosu konularına istinaden bir çalışma yapmamız istendi.Tüm bunları kapsaması açısından 5 adet sınıf oluşturduk.Ödevde bizden istenilenleri kısmen gerçekleştirdiğimize inanıyoruz.Detaylar aşağıdadır.

HuffmanDugumu.cpp

Huffman bir agac yapısı olarak kullanıldığı ve ağaçların düğümlere ihtiyacı olduğu için burada 2 kuruculu bir sınıf olan huffman dugumu sınıfı oluşturuldu.Oluşturulan sınıf içerisinde hem bildiğimiz anlamda dugum işlemleri yapılmakta hem de bazı operatörler cout bu sınıftan oluşturulacak nesnelerin varlığını önceden bilmediği için aşırı yüklenmiştir.Dosyadan okunacak her string ifade karakterlerine parçalanıp bu şekilde huffman dugumlerinde tutulacaktır.Bu sebeple dugumler karakter frekans ve huffman kodkarsiligi barındırmaktadır.

HuffmanAgaci.cpp

Huffman agaci bilindiği üzere bir sıkıştırma algoritması şeklinde çalışmaktadır.Bu sınıf bu algoritma için gerekenleri sağlamaktadır.Yazılırken Fatih hocanın kaynak kodlarından yararlanılmıştır.Zira cpp programlama diliyle yazılmış STL kütüphaneleri dışında belkide en efektif en bizden istenilene uygun fonksiyonlar burada yerini almış durumdadır.Ek olarak DugAra fonksiyonu ile önce Kodla ve DosyaOku fonksiyonları ile oluşturulmuş ağaç ve kodlanmış değerlere main de tekrar aynı dosyadan veriler alınarak alınan verilerin string haliyle totalde kaç bitten oluştuğunun hesaplanmasına yardımcı olunmuştur.Bu fonksiyon yazılırken huffman agaci dugumleri nasıl gezilmeli bilinemediğinden epey uğraştık.Fakat sonunda bir çözüm geliştirmeyi başarabildik.Root düğümden başlamak kaydıyla tüm ağaç gezilerek uygun char ifadeye denk gelen huffmankodu ağaçtan toplanıp return ifadesi ile gerekli diğer yapılara yollanarak gayet efektif bir kullanım geliştirdik.

Kisi.cpp

Hash tablosu da yine huffman agaci gibi dugumlere sahiptir ve bu düğümler içerisinde Kisi nesneleri tutar.Kisi sınıfı huffman agac yapısından çok hash tablosu için önemli bir rol oynamaktadır.Kurucusu ile aldığı string ifadeleri kendi içerisinde tutarak daha sonra hash tablosu oluşturulurken devreye girecektir.

HashDugum.cpp

Bu sınıf bir çok anlamda standart bir dugum sınıfından farklı degildir.Veri olarak kisi nesneleri tutar ve kisi nesnelerinin dugAra gibi fonksiyonlardan aldığı string ifadeleri veri olarak saklar.Daha sonrasında HashTablosu oluşturulurken aynı hash indeks değerine sahip veriler için ağaç oluşturulması gerektiğinde kullanılacaktır.Her yeni değer eklendiğinde önce hashtablo sınıfı içerisindeki bul fonksiyonu yardımıyla kontrol edilip sonrasında ekle fonksiyonu ile bu sınıfın nesnelerine değerler eklenmektedir.

HashTablo.cpp

Bütün işin yapıldığı sınıf burasıdır.Burada özellikle hesaplaVeEkle(string) fonksiyonu çok çok önemlidir.Kodumuzun çalışma mekanizması hash dizi indeksinin her elemanı için bir kişi nesnesi oluşturmaya dayanıyordu ama buradaki sorun.kisi nesnelerinin içerilerinde huffman agacı tutuyor olmalarıydı.Yani her kişi nesnesi içerisinde hem bir huffman agacı hemde ayrıca yine agac oluşturulması için kullanılan text dosyasından okunan değerler vardı.Burada önce ağaçların oluşturulması sonrasında hash tablosuna gereken eklemeler yapılması imkansız ve hatalı bir hal alıyordu.En baştaki bu yanlış tasarımdan ötürü çok vakit kaybettik.Fakat şu anki mevcut haliyle nesne olarak oluşturulan her hash tablosu içerisinde düğümlere ve bu düğümlerin içerisinde tuttuğu kişi nesnelerinden başka bir tane de huffman agacı oluşturuyordu.Yani önce hashtablo nesnesi içerisindeki huffman agacına dosya okutuyor sonra yine aynı agaca kodlatıyoruz ki daha sonra değer eklerken okunacak her stringin karakteri için bir kod hali hazırda oluşturulmuş olsun ve bize sadece bu binary string ifadeyi desimal integer sayıya dönüştürme kısmı kalsın.İşte bunu hesaplaVeEkle(string) fonksiyonuyla mainde önce huffman agacina kodlama ve dosya okuma yaptırtıp sonra aynı text dosyasını tekrar okutarak hesaplatıp eklettirerek becerebildik.HesaplaVeEkle fonksiyonu içerisinde hashtablosunun ekle fonksiyonunuda içerdiği için değer eklemek bu şekilde sorun olmadı.Ve ancak ödevi son haline getirebilmiş olduk.

Sonuç

Ödevde yine en başta yaptığımız yanlış tasarımdan ötürü sıkıntı yaşadık.Fakat tekrar başına oturup bu tasarımı değiştirebilince bir şekilde üstesinden geldik.Aslında ödevde istenilen tek bir şey dışında her şeyi gerçekleştirebiliyoruz.Gerçekleştiremediğimiz tek şey indeks hesapla sırasında binarystringleri desimal integerlara dönüştürürken çok büyük sayıları integer da depolayamadığımız için negatif değerlerle dönen bazı değerlerin modu alındığında – birer değer döndürüyorlar.Hash tablosu indeks olarak eksi değer kabul etmediği için bu gibi durumlarda program terminate oluyor.Bizde bunu hashtablo.cpp içerisindeki desimalHesapla fonksiyonunda eksi değer dönerse pozitife çevirmesi şeklinde bir komut vererek değiştirdik.Şuan ki haliyle 500 ismide okuyup ağaç oluşturabilmekte ve hepsi için bir huffman kodu oluşturabilmekteyiz.Fakat hashtablosu geri dönen bu eksi değerleri tam anlamıyla modunu alamadığımız için hatalı bir yerleştirme yapıyor olabilir.

Bu sene bizim için oldukça verimli geçen bu dersle alakalı verilen ödevler ve kazandırılmaya çalışan program yazma alışkanlıkları için lütfen teşekkürlerimizi kabul edin.

Saygılarımızla

İsmet GÜZELGÜN

İbrahim AKDAĞ