Ömer Can ÇALIŞIR - Melih Ensar BARIŞIK

[redacted]1A - [redacted] 1A

Veri Yapıları 2. Ödev Raporu

Öncelikle ödev dokümanında da şeklen tasvir edildiği gibi bir .text dosyasından okunan kelimelerin her biri oluşturacağımız iki yönlü bağıl listedeki düğümlere yerleştirilecektir. Fakat yerleştirme yapmadan önce aynı kelimenin önceden eklenip eklenmediğine dikkat edilecek , eğer eklendiyse kaç adım ilerisine eklendiği tutulacaktır. Okunan kelime düğüm içerisinde yoksa düğüme eklenecek ve ayrı bir veri olarak sıfır değeri tutulacaktır. Tüm bunların sonunda bu işlemlerden elde edilen byte cinsinden kazanç hesaplanacaktır.

İşlemleri yapmak üzere tanımlanacak sınıfların; Node sınıfı, Iterator sınıfı, DoubleLinkedList sınıfı ve FileRead sınıfı olmasına karar verdik.

Tasarımımızın uygulamasına Node sınıfını oluşturarak başladık. Node sınıfını oluştururken, daha sonra iki yönlü bağıl listede kullanılacakları için bunu göz önünde bulundurarak; önceki düğümü tutan bir prev değişkeni, sonraki düğümü tutan bir next değişkeni ve kelimeleri tutacak bir data değişkeni oluşturduk. Node sınıfının tasarımı böylelikle bitmiş oldu.

Sonrasında düğümler arasında kolayca dolaşmak ve kodun okunabilirliğini artırmak amacıyla Iterator sınıfını tasarladık. Iterator ın gösterdiği düğümün ilerisine ve gerisine hareket edebilmek amacıyla next ve prev seklinde iki basit fonksiyon yazdık. Iteratorun gösterdiği düğümün datasına ulaşmak için getCurrent isimli bir fonksiyon ve Iteratorın düğümler üzerinde hareket ederken baş ve son düğüme ulaştığını kontrol edebilmek adına canMove adlı bir fonksiyon yazdık. Iterator sınıfını bitirmiş olduk.

Daha sonra kelimeleri düğümler kullanarak tutacağımız yapı olan iki yönlü bağıl liste(DoubleLinkedList) sınıfımızı oluşturmaya başladık. İlk olarak istenilen konumdaki düğümden bir önceki düğüme ulaşmamızı sağlayan OncekiniKonumuIleBul isimli bir fonksiyon yazdık. Bu fonksiyonda oluşturduğumuz iterator nesnesi ile istenilen düğümden bir önceki düğüme iteratorımızı hareket ettiren bir while döngüsü yazdık. While döngüsünün sonunda bu düğümü geri döndürdük. Ardından addLast isimli bir fonksiyon tanımladık. Bu fonksiyonda iki yönlü bağıl listenin sonuna yeni bir düğüm ekleme işlemini yaptık. Fonksiyon içerisinde tanımlanan iterator nesnesi, OncekiniKonumuIleBul fonksiyonu yardımıyla son

düğüme gidiyor ve sonrasında bu düğüme yeni bir düğüm bağlama işlemini gerçekleştiriyor. Fonksiyon işlemlerini bitirdikten sonra çöp oluşmaması adına delete fonksiyonunu kullandık. Ardından compare isimli bir fonksiyon yazdık. Fonksiyon aldığı kelimeye gerekli düzenlemeleri yaptıktan sonra düğüm içerisinde önceden mevcut olup olmadığını kontrol ediyor. Eğer aldığı kelime düğümde önceden mevcutsa ekleneceği konum ile düğümde mevcut olduğu konum arasındaki mesafe düğüme yazılıyor. Böylelikle byte kazancı sağlanmış oluyor. Bu fonksiyonunda ardından bahsettiğimiz byte kazancını hesaplamak adına kazancHesapla isimli bir fonksiyon yazdık. Bu fonksiyon yine tekrar eden kelimeleri tespit edip bu kelimelerin uzunluğunu buluyor. Tekrar eden düğümlerin datasına yazdığımız sayının kaç karakter yer kapladığını hesaplıyor ve daha sonra kârları topladığımız değişkene atıyor. Print fonksiyonu içerisinde düğümün konumuna göre çıktılar alıyoruz. Düğümün başlangıçta veya sonda olduğunu head nesnesinin öncesinde ve sonrasında tuttuğu alanların NULL olup olmadığına bakarak karar veriyor ve istenen şekle uygun çıktılar veriyoruz. Eğer bu iki durum sağlanmıyor ise düğümün orta kısımlarında olduğumuzu anlayarak istenildiği gibi düğümün içinde tutulan bilgiyi ekrana yazdırıyoruz. Kazanç yazdır fonksiyonu kazanç hesapla fonksiyonu ile elde edilen toplam byte kârını ekrana yazdırıyor.