



T.C.
SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
VERİ YAPILARI DERSİ ÖDEV RAPORU

03 NUMARALI ÖDEV

Grup Elemanları:

G161210027 - AYTEKİN ERLALE

G161210060 - OĞUZHAN ALİPEK

SAKARYA
ARALIK, 2017

Veri Yapılarının Avantajlarını Gösteren Proje

Aytekin/Erlale^{1*}, Oğuzhan/Alipek²

^a G161210027 2-B

^b G161210060 2-B

Özet

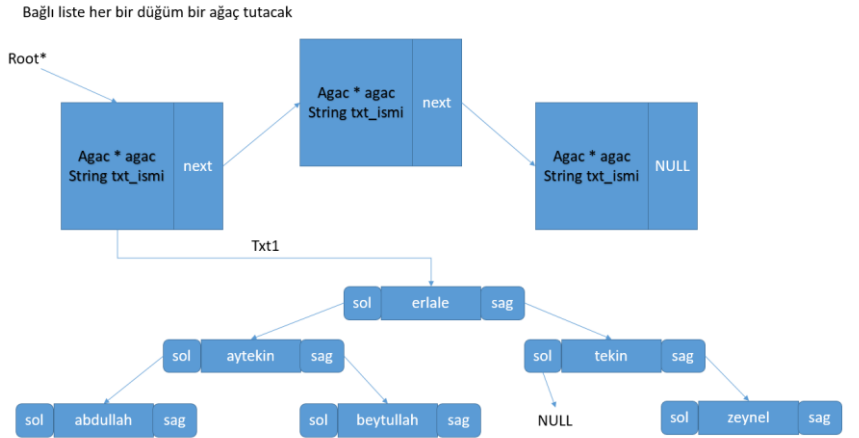
Bu yaptığımız projenin özeti, veri yapılarında ağaç ve liste veri yapılarının avantajlarını ve çalışma mantıklarını nerelerde kullanılması ve nerelerde kullanılmaması gerektiğini anlamaktır. Ödev başlamadan önce anladığımız kadarıyla Görsel 1-A'da görüldüğü gibi bir şablon çıkardık. Kısaca verilen projede sorun bir txt dosyasının içindeki verileri kelime kelime okuyup bu kelimeleri ikili arama ağacına (binary search tree) atmaktır. Bu işlem yapılırken verilerin sırasıyla ağaca atılmaması her seferinde farklı bir ağaç görmemize neden olabilir. Bu nedenle geliştirdiğimiz projede öncelikle 1 nolu referansta belirttiğimiz kaynaktan girilen bir yoldaki dosyaların nasıl okunulacağını öğrendik. Bu adımdan sonra her txt dosyasının verilerini sırasıyla okuyup, ikili arama ağaçlarına yerleştirdik dosya okuması bittiğinde o txt dosyasının bir ağacı olmuş oldu. Bu adımdan sonra projedeye başlamadan önce çizmiş olduğumuz bize göre projede bizden istenilen gereği tek yönlü bağlı listenin her düğümüne, string türünden txt dosyasının ismi ve dosya okuma sırasında oluşturduğumuz ağacı yerleştirdik projemizin mantığı da bu idi.

© 2017 Sakarya Üniversitesi.

Bu rapor benim özgün çalışmamdır. Faydalanmış olduğum kaynakları içerisinde belirttim. Her hangi bir kopya işleminde sorumluluk bana aittir.

Anahtar Kelimeler: İkili Arama Ağacı, Tek yönlü Bağlı Liste, Düğüm, Root

(Görsel 1-A)



* Ödev Sorumlusu. Aytekin Erlale, G161210027,
Mail Adresi: G1612.20027@sakarya.edu.tr

1. GELİŞTİRİLEN YAZILIM

Projede geliştirdiğimiz yazılım C++ dilinde olup amacı bir dizindeki txt uzantılı dosyalarının içindeki verileri okumak,daha sonrasında ise okuduğumuz bu verileri listelemek ve bu txt dosyaları içerisinde arama yapmaktır.Bunu yaparken ağaç veri yapısı ve liste veri yapısı kullanılmıştır.Ek olarak c dili kütüphanelirinden dirent.h'tan yararlanılmıştır.

2. PROJENİN İZLENEN YOL VE ÖNEMLİ NOKTALAR

Önce verilen yoldaki dosyalardan uzantısı txt dosyası olan dosyaları okuyan Islem sınıfında dosya okumasının c dili kütüphanelerinden dirent.h kütüphanesi ile gerçekleştiğini öğrendik ve bu kütüphaneyi Visual Studio 2017'ye include ettik ve yine Islem sınıfında her okunan dosya verisinin o txt dosyası için oluşturulan ağaca sırasıyla yerleştirdik.Dosyadaki verilerin tamamı okunduktan sonra o txt dosyasına ait bir ağaç elde etmiş olduk.Bu ağacı txt dosyasının ismiyle beraber tek yönlü bağlı listenin ilk düğümüne ekledik.Bu kısımdan sonra bir bağlı liste elde ettik.Elde ettiğimiz bağlı listenin her düğümünde bir ağaç ve ağacı elde ettiğimiz bir dosyanın ismi bulunmaktadır.

Projede en zorlandığımız kısım ise aranan kelimenin ağaçlarda bulunup bulunmamasını kontrol ettiğimiz kısımdı.Bu kısımda Görsel 2-A'da görüldüğü üzere Varmi fonksiyonuydu.Bu fonksiyon parametre olarak bir ağaç ve bir string almaktadır.Varmi fonksiyonunun amacı parametre olarak aldığı ağaçta,parametre olarak aldığı kelimenin bulunup bulunmamasıdır.Eğer parametre olarak gelen string değeri parametre olarak aldığı ağaç içerisinde mevcut ise geriye true ya da false değeri döndürecektir.Test.cpp dosyasında çağırdığımız liste sınıfının arama fonksiyonunda Varmi fonksiyonundan geri dönen değer bir düğüm için konrol edilip,true değeri geri dönerse düğümde tuttuğumuz dosya ismi ekrana bastırılacak.eğer false değeri geri döner ise listenin bir sonraki düğümü için bu için konrol edilecek. Bu işlem listedeki bütün düğümler için tekrar edilecek,eğer bağlı listede ki ağaçlarda hiç bir düğümde aranan kelime bulunamadıyda bir mesaj yazdırılacaktır.

(Gorsel 2-A)

```

61  bool Agac::Varmi(Agac * agac,std::string aranan)
62  {
63      //aranan kelime agactaki kelimeye eşitse
64      if (agac->kelime == aranan)
65      {
66          return true;
67      }
68      //aranan kelimenin alfabetik sırada agactaki kelimedden önce gelme durumu
69      else if (agac->kelime.compare(aranan) > 0)
70      {
71          if (agac->sol != NULL) {
72              return Varmi(agac->sol, aranan);
73          }
74          return false;
75      }
76      //aranan kelimenin alfabetik sırada agactaki kelimedden sonra gelme durumu
77      else {
78          if (agac->sag != NULL) {
79              return Varmi(agac->sag, aranan);
80          }
81          return false;
82      }
83  }
84  }
```

3. ALGORİTMA

3.1 Algoritma

- I. Girilen yoldaki dosyalar içerisinde txt dosyalarının içerisindeki veriler okunacak
- II. Okunan her veri sırası ile ağaca aktarılabacak her bir txt dosyası için bir ağaç oluşturulup, dosyada ki verileri okuma işlemi bitene kadar aynı ağaç kullanılmaya devam edecektir.
- III. Bütün dosyalar okundu, her dosya için verilerin bulunduğu bir ağaç elde edilecek, ve her dosya için bir düğüm içerisinde elde edilen ağaç ve dosya ismi yerleştirilecektir.
- IV. Oluşan listenin düğümlerinde ki verileri listelenecek ve bu düğümlerdeki ağaçlarda arama yapılacaktır

3.2 Kullanılan teknikler ve sebepleri

3.2.1 Kullanılan teknikler

- i. Girilen yoldaki dosyalar listelenirken c dili kütüphanelerinden **Dirent.h** ve dosyaların içerisindeki veriler okunurken **ifstream** nesnesi kullanılmıştır.
- ii. Okunan veriler **İkili arama ağaçlarında(binary search tree)** tutulmuştur.
- iii. Liste kısmında ise **tek yönlü bağlı liste** ve ekleme yapılırken ise **sona ekleme** tekniği kullanılmıştır.
- iv. Ağaçları Listelerken **İn-order** sıralama tekniği kullanılmıştır.

3.2.2 Kullanılan tekniklerin sebepleri

- i. Dirent.h kütüphanesinin kullanılma sebebi dosyalara erişmektir. İfstream nesnesinin kullanılma sebebidir dosyaların içerisindeki verileri okumaktır.
- ii. İkili arama ağaçlarına veri yerleştirilirken ilk elemanın solundaki değerler ondan küçük sağındaki değerler ondan büyüktür(alfabetik sıralamaya önce gelen kelime küçüktür). Bu bize arama yaparken kolaylık sağlamaktadır.
- iii. Tek yönlü bağlı liste çift yönlü bağlı listeye göre hafızada daha az alan tutmaktadır. Ekleme yaparken ise sona ekleme tekniğinin kullanılması işlem karmaşıklığından kurtulmak ve dizinden okunan dosyaların sırasıyla listede tutmaktır.
- iv. Sebebi proje dökümanında in-order sıralamasının kullanılması gerektiği yazmaktadır ayrıca in-order tekniği alfabeğe göre küçükten büyüğe sıralamadır. Arama yapılırken başta ikili arama ağacı bize kolaylık sağlamaktadır.

4. ÇIKTILAR

(Görsel 3-B)

Bu bilgisayar > Arşiv (D:) > veri

| Ad | Değiştirme tarihi | Tür | Boyut |
|-------------------------------------|-------------------|----------------------|-------|
| Yeni klasör | 20.11.2017 06:23 | Dosya klasörü | |
| txt1.txt | 4.12.2017 15:34 | Metin Belgesi | 1 KB |
| txt2.txt | 4.12.2017 15:33 | Metin Belgesi | 1 KB |
| txt3.txt | 4.12.2017 15:34 | Metin Belgesi | 1 KB |
| Yeni Microsoft Excel Worksheet.xlsx | 20.11.2017 06:23 | Microsoft Excel W... | 7 KB |

(Görsel 3-A) – Projeyi Analiz ederken kullanacağımız dizin.

(Görsel 3-B)

C:\Users\Aytekin\source\repos\VeriYapilari3.Odev\Debug\VeriYapilari3.Odev.exe

Klasör...: D:\\veri

(Görsel 3-B) – Projenin ilk çalıştığında bizi karşılayan ekranı bir klasör yolu girmemiz gereklidir.

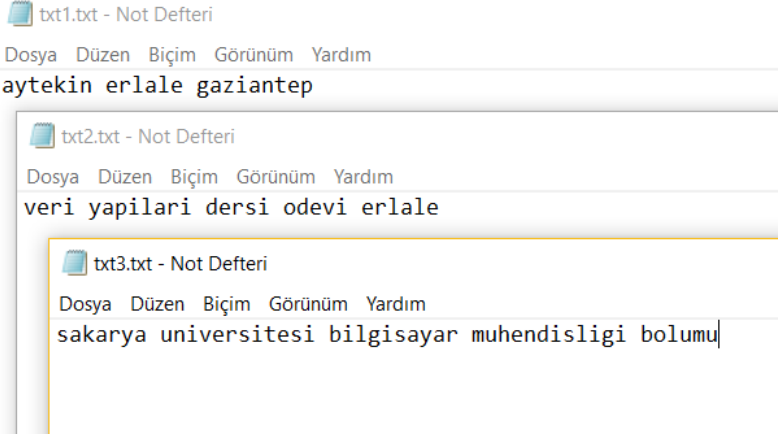
(Görsel 3-C)

C:\Users\Aytekin\source\repos\VeriYapilari3.Odev\Debug\VeriYapilari3.Odev.exe

```
1-Arama Yap
2-Listele
3-Cikis
>>>2
```

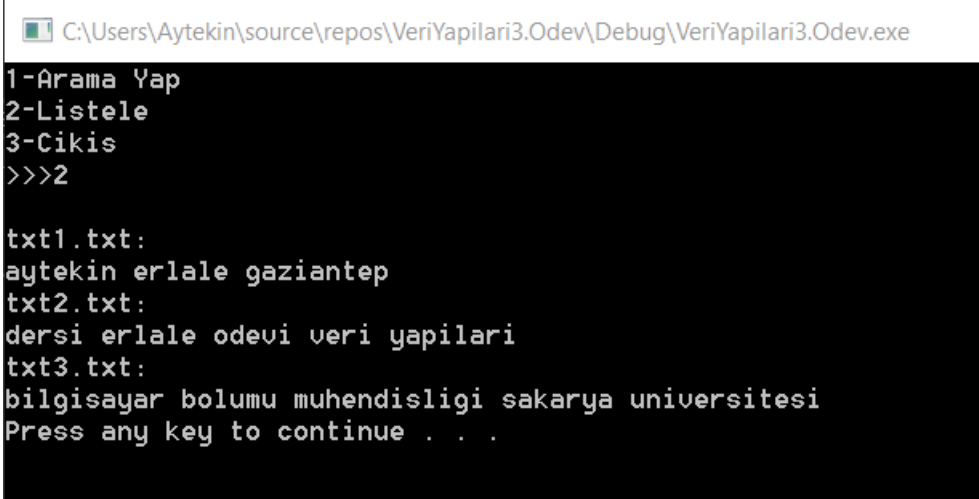
(Görsel 3-C) – Yolu girdikten sonra yapmak istediğimiz işlemler menüsü verileri listeletiyoruz

(Görsel 3-D)



(Görsel 3-D) – Dizindeki txt dosyalarının içerisindeki veriler

(Görsel 3-E)



(Görsel 3-E) – Dizindeki sadece txt dosyalarının içerisindeki verileri bize in-order sıralı liste olarak getirdi.

(Görsel 3-F)



```

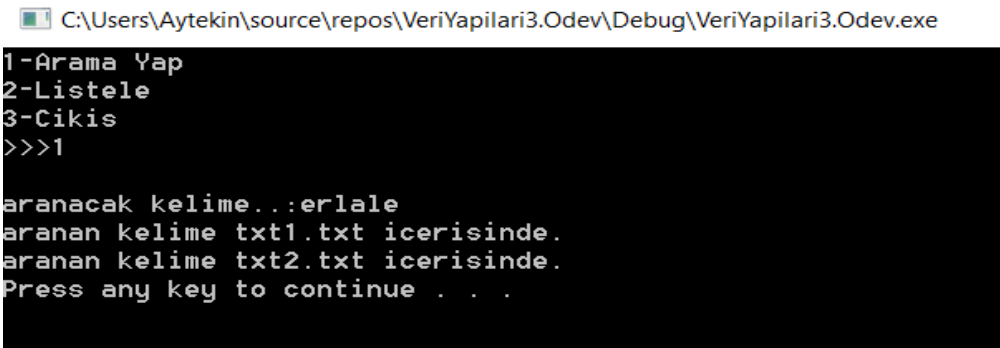
C:\Users\Aytekin\source\repos\VeriYapilari3.Odev\Debug\VeriYapilari3.Odev.exe
1-Arama Yap
2-Listele
3-Cikis
>>>1

aranacak kelime...:erlale

```

(Görsel 3-F) – Txt dosyaları içerisinde kelime aranacaktır

(Görsel 3-G)



```

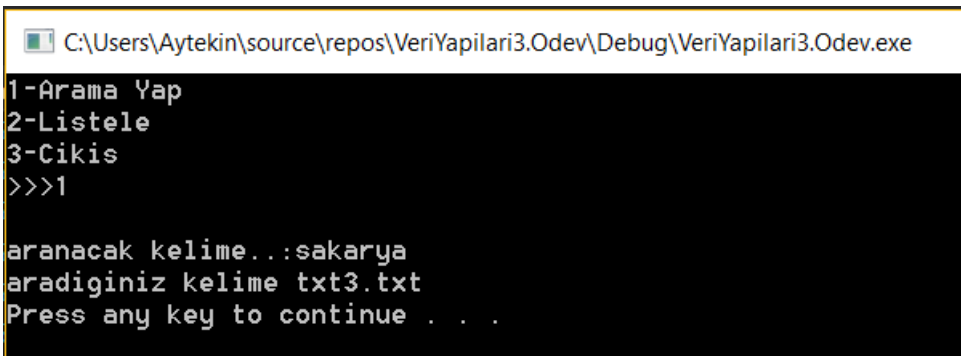
C:\Users\Aytekin\source\repos\VeriYapilari3.Odev\Debug\VeriYapilari3.Odev.exe
1-Arama Yap
2-Listele
3-Cikis
>>>1

aranacak kelime...:erlale
aranan kelime txt1.txt içerisinde.
aranan kelime txt2.txt içerisinde.
Press any key to continue . . .

```

(Görsel 3-G) – Erlale kelimesi arandığında doğru bir şekilde kelimenin bulunduğu txt dosyalarının ismini yazmıştır.

(Görsel 3-H)



```

C:\Users\Aytekin\source\repos\VeriYapilari3.Odev\Debug\VeriYapilari3.Odev.exe
1-Arama Yap
2-Listele
3-Cikis
>>>1

aranacak kelime...:sakarya
aradiginiz kelime txt3.txt
Press any key to continue . . .

```

(Görsel 3-H) – Sakarya kelimesi arandığında doğru bir şekilde kelimenin bulunduğu txt dosyasının ismini yazmıştır.

(Görsel 3-I)

C:\Users\Aytekin\source\repos\VeriYapilari3.Odev\Debug\VeriYapilari3.Odev.exe

```
Klasor...: D:dsada
Hata NO(2)D:dsada
Press any key to continue . . .
```

(Görsel 3-I) –Grililen dizinde hata varsa veya soya mevcut değilse hata döndürüp program kapanmıştır.

(Görsel 3-J)

C:\Users\Aytekin\source\repos\VeriYapilari3.Odev\Debug\VeriYapilari3.Odev.exe

```
1-Arama Yap
2-Listele
3-Cikis
>>>1

aranacak kelime...:dosya
aranan kelime bu dizinde bulunamadi
Press any key to continue . . .
```

(Görsel 3-J) –Aranan kelime txt dosyaları içerisinde bulunamadıysa ekrana bulunamadi mesajı yazmıştır.

5. SONUÇ

5.1 Kazanımlar

- a. Bu proje veri yapıları dersinin tam manasıyla gereğinin yapıldığı bir projeydi. Projenin bize bir çok katkısı olduğunu söylememiz gerekir. Ağaçları tam manasıyla özellikle ikili arama ağaçlarını tam manasıyla anladığımızı düşünüyoruz. Bağlı listelerde ekleme tekniklerini öğrendik. Dient.h, özyinelemeli fonksinlar hakkında bir çok şey öğrenerek ve bu projede pratik yaparak bu bilgileri pekiştirdik.
- b. Ayrıca grup olarak bu projeyi yürütmek, her ikimizinde takım uyumunu sağlama, takımla beraber hareket etme, yapılacak işleri bölmek ve kısa sürede yapmak gibi alışkanlıkları kazanmamıza yardımcı olmuştur

5.2 Projenin gerçek hayattaki faydaları

1. Bir veri grubunu ikili arama ağaçlarına atıp içerisindeki verilere hızla ulaşmak.
2. Ve bu verilerin bulundukları dosyaları bulmak ve bunu hızlıca yapmak.

Referanslar

[1] <https://arstechnica.com/civis/viewtopic.php?t=939142>

[2] <https://github.com/tronkko/dirent/blob/master/include/dirent.h>

[3] <https://codeyarns.com/2014/06/06/how-to-use-dirent-h-with-visual-studio/>