Programlama Laboratuvarı III. Proje Raporu

Umut YİĞİT
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi 2022
200201041

Betül BODUR
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi 2022
210201069

I. Özet

Bu dokümanda Programlama Laboratuvarının 3. projesi olan Soy Ağacı oluşturup kullanıcıya çeşitli bilgilerin sağlanması için oluşturulan algoritmalar açıklanmıştır . Dokümanda problemin tanımı verilip bu problemler hakkında araştırmalar ve yöntemler sunulmuştur.Araştırmalar ve yöntemlerde bahsedilen class ve metotların ayrıntılı açıklaması bulunmaktadır.

***Index Terms*—soy,nesne,metot,ağaç,kişi,arraylist**

II. Problem Tanimi

Bu projede verilen 4 sayfalı, içerisinde kişisel bilgilerin (ad. soyad, eş bilgisi vb.) ve her kişi için özel bir id olan `.csv` uzantılı bir dosyanın içerisindeki bilgilerin okunması, kişilerin birbiri ile olan ilişkilerin anlanması ve uygun ağaç yapısının oluşturulması hedeflenmektedir.Bu proje java programlama dili ile yazılmıştır

Kişilerin bilgilerinin bulunduğu `.csv` uzantılı bir dosya okunup içerisinde bulunan bilgiler yardımıyla soy ağacı oluşturulmalıdır.Kişi adında tc no (id), ad, soyadı, doğum tarihi, anne adı, baba adı, kan grubu ve meslek, kızlık soyadı ve cinsiyet bilgilerinin bulunduğu sınıf yapısı oluşturulmalıdır.Ağaç yapısı oluşturulduktan sonra; Çocuğu olmayan düğümler depth first algoritması ile bulunup yaş sıralamasına göre kayıt edilmeli ve sıralama adımları gösterilmeli, üvey kardeşler breath first algoritması ile bulunarak harf sıralamasına göre kayıt edilmeli ve sıralama adımları gösterilmeli, Kullanıcıdan alınacak kan grubuna sahip bireyler gösterilmeli,kişiden alınacak kişi bilgisi ile kişinin soyunda aynı mesleği yapan çocukları ve torunları gösterilmeli,soy ağacında aynı isme sahip bireylerin ismi ve yaşları gösterilmeli,kullanıcdan alınacak 2 tane isim girdisinden sonra büyük olan kişinin küçük olan kişiye yakınlığı gösterilmeli,kullanıcıdan alınacak kişi bilgisi ile o kişiye ait soy ağacı gösterilmeli ve yakınlığı yazılmalı,soy ağacının kaç nesilden oluştuğu bulunmalı,kullanıcıdan alınan isim girdisinden sonra kaç nesil geldiği bulunmalıdır.

III. Araştırmalar ve Yöntemler

Projeye ilk başta verilen csv uzantılı dosyalarda bulunan bilgilerin düzenli bir şekilde alınması ile başlanmıştır.

Programın başlangıcında verilen bir adet 4 sayfalı `.csv` uzantılı dosya mevcuttur. Dosya okuma konusunda sorun olmaması için öncelikle csv dosyası Excel yardımı ile ayrı ayrı 4 tane `.csv` uzantılı dosya olarak kayıt edilir.

Program çalışırken, verilen csv dosyalarındaki eksiklikler sebebiyle doğru sonuçlar alınamamıştır.Bu eksikliler bazı kişilerin eş adı sütununda eşinin soyadının yazmamasından,eşinin soyadından farklı olmasından ve eşinin csv uzantılı dosyada bulunmamasından kaynaklanmaktadır.Bunun için SoyadDuzelt() metodu ve esDugumKontrol() metotları yazılarak düzeltilmiştir.

Projede ağaç yapısı oluşturulduktan sonra ağacı oluşturan kişilere göre 1920*1080 çözünürlüklü ekrana sığmamasından dolayı ağaç yapısının tamamı gözükmemektedir.Bunu çözebilmek için ekranımızın çözünürlüğü arttırılıp sorun çözülmüştür.

Kodun çalıştırılabilir halinde (jar dosyası) programın içine eklenen görüntülerin gözükmesi için projede kullanılacak görüntüleri URL tipinde tanımlayıp getClasses() metodu,getResource() metotları kullanılarak,jar dosyası çalıştırıldığında görüntülerin kaybolması engellenmiştir. Örnek kod;

URL duz = this.getClass().getResource("/Excel/duz.png");

IV. Geliştirme Ortami

- Bu proje `JAVA` programlama dili ile yazılmıştır.
- NetBeans 15 kullanıldı.
- JDK Sürümü: 19

V. Kod Bilgisi

A. Class'lar

Programın sınıfları şu şekilde özetlenmiştir:

Kisi Sınıfı:

Kisilere ait bilgilerin tutulduğu sınıftır.Kişinin ; ad, soyad, dogumTarihi, esi, anneAdi, babaAdi, kanGrubu, meslek, medeniHal, kizlikSoyadi, cinsiyet'i burada tutulmaktadır.

FamTree Sınıfı:

Ana class'tır.Programın başlangıcında yapılması gereken csv uzantılı dosyaları okuma,düzenleme işlemi bu sınıfta yapılmaktadır.Bu sınıfta main metodu,esDugumKontrol metodu ve SoyadDuzelt metodu bulunmaktadır.

- main metodu

`CSV` (Comma Seperated Variables) dosyaları içerisine bilgiler belirli özel karakterler ile (örneğin ',' veya ';' gibi) ayrılarak kayıt edilir. Dosya içerisindeki veriler okunurken bu *ayraçlara göre* okunması gerekmektedir.
Dosyaların okunma işlemi program başında 4 kere çalışan bir döngü sayesinde yapılmaktadır. Döngü içerisinde i değişkeni 1'den başlar ve 4 olana kadar devam eder.(page1 , ... , page4 olması sebebiyle)

```
String file = "./src/Excel/page"+i+".csv";|
```

Döngü her çalıştığında dosyanın dizini için kullanılan file String'i değişir ve sırayla 4 adet CSV dosyasının okunması sağlanır. Her okunan satır `regex` yardımı ile ' ; ' karakterine göre ayrılır ve bir string dizisi içerisine kayıt edilir.

Bu aşamadan sonra okunan satırdaki veriler *ayrılarak* elde edilmiş olur. Bir sonraki işlem olarak for each döngüsü ile string dizisi içindeki her bir veri, sırayla okunarak yeni bir kişi için oluşturulan yeniKisi nesnesinin gerekli değişkenlerine atanır ve yeniKisi nesnesi kişileri tutmak için içerisinde Kisi sınıfından nesneler tutan kisiler ArrayList'i içerisine kayıt edilir.

```
  ArrayList <Kisi> kisiler=new ArrayList
<Kisi>();| Kisi yeniKisi = new Kisi();|
```

Artık csv dosyalarındaki kişilerin oluşturduğu, içerisinde Kisi sınıfından nesneler tutan bir ArrayList `(kisiler)` mevcuttur ancak verilen aile listelerinde bazı kişiler diğer aileler ile bağlantılı olduğundan bir kişi birden fazla kez ArrayList içerisine kayıt edilmektedir. Bu yüzden kayıt işleminden sonra bir karmaşıklığı $n\hat{2}$ olan metodun içinde indis ve bir sonraki indiste ilerleyerek id'si eşit olan kişileri bulup, kişiler arraylistin bir metodu olan remove() ile kisiler arraylistinden kaldırılmıştır.

Projenin tamamlanabilmesi için CreateTree sınıfı oluşturulmuştur.main metodunda bu sınıfa ait bir nesne oluşturulmuştur. Ve bu nesne ile CreateTree sınıfın Create() metodunu çalıştırmıştır.Bu metodun karmaşıklığı $n\hat{3}$'tür

- SoyadDuzelt metodu

Okunan Csv dosyalarında bazı kişilerin eşlerinin soyadlarının olmadığı ve bazı kişilerin de eşi ile soyadının farklı olduğu görüldüğünde sorunu çözmek için bu metoda ihtiyaç duyulmuştur. Bu metot kisiler listesinde bulunan herkesi gezer ve eşlerinin adında bulunan boşluk karakterinin indisine göre soyadının olup olmadığına karar vermektedir.

int indis = kisiler.get(i).esi.indexOf(" ")

- *indis* değeri sıfır ise eşi olmadığı anlamına gelir ve herhangi bir işlem yapılmaz.
- İndis değeri sıfır değil ise kişinin eşinin adı kişinin eşinin adı + kişinin soyadı şeklinde güncellenir.
- İndis değeri sıfırdan küçük ise kişininin eşinin sadece adının olduğu ve soyadının yazılı olmadığı anlamına gelir. İndis değeri kişinin eşinin isim uzunluğu olarak güncellenir ve tekrardan kişinin eşinin adı kişinin eşinin adı + kişinin soyadı şeklinde güncellenir.

- İsim güncelleme işlemi *indis değeri sıfırdan büyük ise* yapılmaktadır.Bu metodun karmaşıklığı $n\hat{2}$'dir

- esDugumKontrol metodu

Okunan Csv dosyalarında bazı kişilerin eşi olduğu halde eşlerine ait bilgilerin csv dosyalarında bulunmadığı fark edildiğinde sorunu çözmek için bu metoda ihtiyaç duyulmuştur. Bu metot iç içe iki adet *for döngüsü* ile çalışmaktadır. İlk döngüde (i) tutulan kişinin eş bilgisi ile ikinci döngüden (j) gelen kişinin ad ve soyad bilgileri kontrol edilir. Eğer eşleşme sağlanırsa herhangi bir işlem yapılmaz ancak eşleşme sağlanamaz ise (ilk döngüdeki kişinin eşi listede yok ise) kisiler listesinin en sonuna eş ad ve soyad bilgisi ile yeni bir kişi oluşturulur.Bu metodun karmaşıklığı n'dir.

CreateTree Sınıfı:

Create sınıfı içinde düğüm yapıları oluşturulmak için Node adında bir sınıf oluşturulmuştur.Bu sınıfta anne, baba,çocuk,kardeş,eş düğümleri oluşturulup kişi ye aşağıda anlatılacak olan metodlarla bağlanmıştır.Kişinin bağlantılarının kontrol edilmesi amacıyla kontrol değişkenleri eklenmiştir.Düğümlerin tutulması için içerisinde Node sınıfından nesneler tutan bir arrayList oluşturulmuştur.

- Create metodu

Create metodunda, her kişinin düğüm olmasından dolayı, her kişinin düğümü oluşturulup düğümleri tutan arrayListe eklenmiştir.Ve kişilerin eş , kardeş ve çocuk düğümleri null ise karşılıklı olarak eşitlemek için CocukBul(),KardesBul() ve EsBul() metotları uygulanır.

Döngü ile kişiler gezilir ve kişinin kardeşi varsa; kendi annesini ,kardeşinin annesine ve kendi babasını ,kardeşinin babasına bağlar.
Tekrar bir döngü ile kişiler gezilir ve kişinin üvey kardeşi varsa; kişinin anne adı ve üvey kardeşin anne adı aynı ve babas adı ile üvey kardeşinin baba adı farklıysa kişinin annesi üvey kardeşin annesine bağlanır veya kişinin baba adı ve üvey kardeşin baba adı aynı ve anne adı ile üvey kardeşinin anne adı farklıysa kişinin babası üvey kardeşin babasına bağlanır.Bu üvey kardeş düğümünde ki kişiler UsiblingsID adındaki bir arraylistte tutulur.
Kişilere ait çocukları tutan bir arraylist oluşturulur.Bu arraylist hem arayüz oluşturulmasında hem de çocuğu olmayan düğüm bulmak için kullanılacaktır.
assignXY metoduna dugumler parametresiyle gönderilir.
Verilen aileler arasında direkt olarak bir bağlantı olmadığından arayüz üzerinde ağaç gösterimi yapılırken bazı kişilerin düğümlerinin çakıştığı tespit edildi.Çözüm olarak eş ve çocuk düğümüne herhangi bir kişi bağlı olmayan kişilerin x değerleri karşılaştırılarak ilk köke ait ağacın en sağdaki düğümü bulundu.Daha sonrasında gelen ailenin ağacı yeni bir kökten çıkacağı için y değeri 1 den başlamaktadır. X Max değeri bulunduktan sonra y değeri 1 olan ve x değeri maksimum x değerinden (ilk ağacın en sağı) küçük olan düğüm bulunursa o düğümün (çakışan düğümler) x değeri 3 arttırılır ve x ve y değer atama fonksiyonu tekrar çağrılır.Güncel x ve y değerleri ağaç çakışma olmadan tekrar çizilmiş olur.
Kullanıcıdan isim girdileri istenir ve iliskiBul() metodu çağrılır.Düğüm döngüsünde kişilerin eşi varsa ve cinsiyeti yazılmamışsa kişinin cinsiyeti equals() metodu ile bulunur ve kişinin eşinin cinsiyetide , kişinin tam tersi olacak şekilde yazılmaktadır.
Bazı kişiler listede bulunmamaktadır.Bulunmayan kişileri düğüme kattığımız zaman çoğu bilgisi boş kalmaktadır.Özellikle algoritmanın bakması gereken doğum tarihi yeni kişilerde bulunmadığından hataya yol açmaktadır.Bazılarında iste ay veya gün tek haneli yazılmıştır bu da hataya yol açmaktadır.Bu çözmek için ise eğer doğum tarihinin uzunluğu 10 değil ise başına 0 ekle eğer kişinin doğum tarihi 0 ise kişinin doğum tarihini 00.00.0000 olarak değiştirir.

Ağaçta isimleri aynı olan kişileri bulmak için iç içe 1 for tanımlanır.İlk for içinde kişinin doğum tarihi alınır ve doğum tarihini string veri tipinden integer veri tipinde çevirir.Ve yaşını bulmak için bulduğu tarihi 2022'den çıkararak kişinin yaşını bulur.İçerdeki for'da ise yine aynı işlemler yapılıp ikinci kişinin yaşını bulur. Ve bi if içerisinde eğer dışarıdaki for'dan gelen kişi ile içerideki for'dan gelen kişinin adı aynı ise kişilerin adlarını ve yaşlarını yazdırır.
Ağacın derinliğini bulabilmek için döngüye girmeden önce yMax=0 şeklinde bir değer tanımlanır.for döngüsünde tüm kişilerin y'sine bakar eğer kişinin.y'si yMax değerinden büyük ise yMax'ın yeni değeri kişinin.y'si olur.Ağacın derinliği ise ; Ağacın Derinliği = (yMax + 1) / 2
şeklinde bulunmuş olur.
Son olarak bu metotta sırasıyla BFS(),DFS(),ArayuzOlustur() metodları çağrılır.Bu metodun karmaşıklığı n$\hat{2}$'dir.

- alfabetikSirala metodu

Graph sınıfından çağırılan alfabetikSirala metodu kişinin id'sinin bulunduğu arraylist ve Kisi tipinden nesneler tutan kisiler arraylistini parametre olarak almaktadır.n$\hat{2}$ karmaşıklığı içeren bubble sort sıralama algoritması kullanılarak ve gelen id listesinden j. kişinin id'si üzerinden kişinin adına erişilir ve j+1. kişinin de id'si üzerinden adına erişilir.Ve isimler charAt() metoduyla harf harf kıyaslanır.Eğer kişilerin adı aynı ise j. kişinin id'si üzerinden kişinin soyadına erişilir ve j+1. kişinin de id'si üzerinden soyadına erişilerek harf harf karşılaştırma yapılır ve bu metot sonunda Graph metodundan gelen id arraylistesi alfabetik olarak sıralanmış olur.Bu metodun karmaşıklığı n$\hat{2}$'dir

- dogumTarihiSirala metodu

Graph sınıfından çağırılan dogumTarihiSirala metodu kişinin id'sinin bulunduğu arraylist ve kisiler düğümünü parametre olarak almaktadır.n$\hat{2}$ karmaşıklığı içeren bubble sort sıralama algoritması kullanılarak ve gelen id listesinden j. kişinin id'si üzerinden kişinin doğum tarihine erişilir ve j+1. kişinin de id'si üzerinden doğum tarihine erişilir.Gelen doğum tarihleri GG/AA/YY şeklinde tutulduğu için ve bunların indexleri bilindiğinden dolayı ikişer tane gün,ay ve yıl değişkeni oluşturup değerlerini charAt() ile alınarak değişkenlere atanmıştır. Değişkenlere atama işlemi yapıldıktan sonra bu sayıları karşılaştırmamız için integer bir değere çevirmemiz gerekmektedir. Bu yüzden aşağıdaki örnek kod parçacığı kullanılarak string integer değere dönüştürülmüştür.
gun1 = Integer.parseInt(gunStr1);
Sıralama algoritmsında önce yıllar karşılaştırılmıştır.Eğer yıllar aynı ise aylar ,aylar aynı ise günler karşılaştırılmıştır.Ve bu metot sonunda Graph metodundan gelen id arraylistesi yaş sıralamasına göre olarak sıralanmış olur.Bu metodun karmaşıklığı n$\hat{2}$'dir

- altSoyGez metodu

altSoyGez metodu girilen kişinin (metot bu kişiyi kök olarak alır) soyunun ağaçta gösterilmesini gerçeklemek için yazılmıştır. Parametre olarak bir adet bir adet Integer ArrayList'i , aramaya başlayacağı kişinin düğümünü ve *dugumler* ArrayList'ini alır. İlk iş olarak metoda paramete olarak gelen *root'un id değerini ve var ise eşinin id değerini* ArrayList içerisine kayıt eder.Bu metodun karmaşıklığı n'dir

Daha sonra kişinin çocuk düğümünü kontrol eder. Eğer çocuk düğümü boş ise kardeşlerine bakar ve her kişinin kardeşlerin kayıt edildiği **siblingsID** listesi içerisinden her kardeş fonksiyona parametre olarak tekrar gönderilir.

(recursive çağrı)

Eğer çocuk düğümü dolu ise bu kişinin en az bir çocuğu olduğu anlamına gelmektedir. *Kişinin tüm çocukları her kişi için oluşturulan **kidsID** listesi içerisinde tutulmaktadır.* Bu liste sayesinde bir **for each** döngüsü ile kişinin bütün çocukları altSoyGez metoduna gönderilir.

Not: Ağaç oluşturulurken kişinin çocuk düğümüne bir adet çocuğu ve eğer varsa diğer çocukları ilk bağlanan çocuğuna kardeş olarak bağlanır.

Sonuç Olarak **altSoyGez** metodunun gezme sırası **Root , Root.Eş , Root.Çocuk** daha sonrasında da Root'un kardeşlerine giderek bu şekilde devam eden **Recursive** bir metot şeklinde gerçekleşmektedir. Metodun ağaçta gezme şekli *Depth First Algorithm* ile benzerlik göstermektedir.

- altSoyGez2 metodu

altSoyGez2 metodu altSoyGez metodu ile çalışma olarak benzerlik göstermektedir. *Recursive bir şekilde çalışmaktadır.* Farklı olan bu metotta gezinmeye kardeşlerden başlanmasıdır. Parametre olarak üstteki gezme metodu ile aynı parametreleri almaktadır. Gezinme Sırası **Root , Root.Kardeş(ler) , Root.Çocuk** şeklinde ilerlemektedir. Daha sonrasında da Root'un çocuklarına geçerek aynı gezme sırası *Recursive* bir şekilde ilerlemektedir. Fonksiyonun ağaçta gezme şekli *Breath First Algorithm* ile benzerlik göstermektedir.Bu metodun karmaşıklığı n'dir

- altSoyGez3 metotları

altSoyGez3 metotları ağacın alt tarafından başlayarak gezinmek için yazılmıştır. Bu bize kişinin üst soylarına giderek ilişkilerinin bulunmasında yardımcı olmaktadır. Öncelikle kişinin sol tarafındaki atalarına doğru gezmeye başlayan altSoyGez3Left çalışır ve kişinin solunda bulunan ataları kayıt eder. Eğer aranan kişinin ID ' si kayıt edilmediyse altSoyGez3Right metodu devreye girer ve aranan kişi var ise kişinin sağ tarafında bulunur.Bu metodun karmaşıklığı 1'dir

Not: Bu metotlar kişinin aile ilişkilerinin bulunmasında kullanılmaktadır. Babasının babasının annesi gibi çıktılar bu metottan gelen ArrayList üzerinden yapılmaktadır.

- altSoyGez4 metodu

altSoyGez4 metoduna kişinin tüm ataları ile olan ilişkisinin yazılabilmesi için ihtiyaç duyulmuştur. Kişinin (metoda ilk parametre olarak gelen kök) öncelikle anne tarafından giderek en üstteki atasına kadar *(anne ve baba düğümü null olan kişiye kadar)* konsol ekranında yazılır. Ardından kök kişisinin baba tarafına da gidilerek aynı işlemler *recursive* bir şekilde tekrarlanır.Bu metodun karmaşıklığı 1'dir

- iliskiBul metodu

Kullanıcdan alınan 2 ismin birbirine olan yakınlığını bulur.Parametre olarak kullanıcdan alınan iki isim,dugumler arraylisti ve kişiler arraylistini alır.Bir döngü içinde dönen isimler ile gelen isimler aynı mı diye kontrol eder aynı ise id1 veya id2 değişkenine o kişinin/kişilerin id'sini atar.Kişilerin id'leri bulunduktan sonra hangisinin büyük olduğu tespit edilir.Ve büyük olan kişinin parametre olarak eklendiği altSoyGez3Left() metodu çağrılır.Eğer altSoyGez3Left() metodunda bulunamadıysa altSoyGez3Right() metodu çağrılır.Kişilerin yakınlık dereceleri bulunur.Bu metodun karmaşıklığı n'dir.

- BFS metodu

Kisi tipinden nesneler tutan kisiler arraylistini parametre olarak almıştır.Graph sınıfından bir nesne oluşturulmuştur.Düğüm yapısındaki her bir kişi için :Kişinin üvey kardeşi varsa üvey kardeşlerinin tamamını Graph sınıfındaki add.Edge() metodunu çağırarak döngü içinde linkliste eklemiştir,Kişinin babası varsa Graph sınıfındaki add.Edge() metodunu çağırarak linkliste eklemiştir ve Kişinin annesi varsa Graph sınıfındaki add.Edge() metodunu çağırarak linkliste eklemiştir.Düğümdeki tüm kişiler bittikten sonra Graph yapısındaki BFS metodunu çağırmıştır.Bu metodun karmaşıklığı $n\hat{2}$'dir.

- DFS metodu

Kisi tipinden nesneler tutan kisiler arraylistini parametre olarak almıştır.Graph sınıfından bir nesne oluşturulmuştur.Düğüm yapısındaki her bir kişi için :Kişinin çocuğu varsa çocuklarının tamamını Graph sınıfındaki add.Edge() metodunu çağırarak döngü içinde linkliste eklemiştir,Kişinin eşi varsa Graph sınıfındaki add.Edge() metodunu çağırarak linkliste eklemiştir.Düğümdeki tüm kişiler bittikten sonra Graph yapısındaki DFS metodunu çağırmıştır.Bu metodun karmaşıklığı $n\hat{2}$'dir.

- assignXY metodu

Arayüzde kişilerin konumlarını doğru gösterebilmek için yazılmıştır.Node tipinden nesneler tutan dugumler arraylistini parametre olarak almıştır.Arayüzde bir anne ve babanın çocukları ve bu çocukların kardeşleri vardır.Bu kardeşlerden bazıları evlidir.Bu aile ağacını arayüzde gösterirken bireylerin üst üste gelmemesi için bazı durumlar göz önünde bulundurularak x ve y atamaları yapılmıştır.Eğer kişinin çocuğu varsa çocuğun koordinatı kişinin.x+1,kişininy+2 şeklinde olmaktadır.Eğer kişinin eşi var ise eşinin koordinatı, kişinin.x+2 şeklinde olmaktadır.Kardeşlerin koordinatları ise diğer kardeşlerinin evli olmasına göre

değişmektedir.Kişinin çocuğu yoksa kardeşinin koordinatı, kişinin.kardeşinin.x+2 şeklinde,kişinin eşi varsa kardeşinin koordinatı kişinin.kardeşinin.x+4, vb. şeklinde bir çok parametre göz önüne alınarak hesaplanmıştır.Bu metodun karmaşıklığı n$\hat{2}$'dir

- EsBul metodu

Düğüm yapısındaki eşleri birbirlerine bağlamak için yapılmıştır.Parametre olarak aramaya başlayacağı kişinin düğümünü,kişiler arraylistini,düğümler arraylistini almıştır.Bir döngü ile tüm düğümleri gezerek kişinin adı ile diğer kişinin eşinin adı eşitlenirse bu kişiler birbirlerine eş olarak bağlanmış olur.Bu metodun karmaşıklığı n'dir.

- KardesBul metodu

Düğüm yapısındaki kardeşleri birbirlerine bağlamak için yapılmıştır.Parametre olarak aramaya başlayacağı kişinin düğümünü,kişiler arraylistini,düğümler arraylistini almıştır.Bir döngü ile tüm düğümleri gezerek kişinin anne adı ve baba adı ile diğer kişinin anne adı ve baba adı eşitlenirse bu kişiler birbirlerine kardeş olarak bağlanmış olur.Bu metodun karmaşıklığı n'dir.

- CocukBul metodu

Düğüm yapısındaki anne ve babayı çocuklara bağlamak için yapılmıştır.Parametre olarak aramaya başlayacağı kişinin düğümünü,kişiler arraylistini,düğümler arraylistini almıştır.Bir döngü ile tüm düğümleri gezerek kişinin eşi varsa ve diğer kişinin annesinin adı ile kişinin eşinin adı ve babasının adı kişinin adı ile eşleşiyorsa kişinin eşinin çocuğu olarak birbirine bağlanır.Eğer döngüde diğer kişinin annesinin adı ile kişinin adı ve babasının adı kişinin eş adı ile eşleşiyorsa kişinin eşinin çocuğu olarak birbirine bağlanır.Ve bu sayede baba-çocuk, anne-çocuk bağlantısı yapılmış olur.Bu metodun karmaşıklığı n'dir.

- UveyKardesBul metodu

Düğüm yapısındaki üvey kardeşleri birbirlerine bağlamak için yapılmıştır.Parametre olarak aramaya başlayacağı kişinin düğümünü,kişiler arraylistini,düğümler arraylistini almıştır.Bir döngü ile tüm düğümleri gezerek kişinin anne adı ve baba adı ile diğer kişinin anne adı veya baba adından bir tanesi eşitlenirse bu kişiler birbirlerine üvey kardeş olarak bağlanmış olur.Bu metodun karmaşıklığı n'dir.

- ArayuzOlustur metodu

ArayuzOlustur metodunda temel olarak 3 adet ağaç oluşturma işlemi yapılmaktadır. Birinci ağaç genel ağacı göstermektedir. Her düğümün X ve Y değerleri daha önce çalışan *assignXY* fonksiyonunda atanmıştı. Ağaç oluşturulurken her bir düğümü tutacak bir adet ArrayList oluşturulur.

✓ `ArrayList <JLabel> agacDugumler = ...`

Her bir düğüm için bir adet Etiket (**new JLabel()**) oluşturulur ve ardından koordinatları ve cinsiyetine göre etiket rengi ayarlanır. Yeri ve rengi belli olan etiket liste içerisine kayıt edilir ve bir sonraki düğüme geçilir. Bu şekilde tüm düğümlerin etiketi oluşturulur ve liste içerisine kayıt işlemi gerçekleştirilir. Temel olarak 3 ağaç için de bu işlem ortaktır.

İkinci oluşturulan ağaç aynı kan grubuna sahip kişileri göstermek için kullanılmaktadır. Sıra bu ağaca geldiğinde kullanıcıdan hangi kan grubuna sahip kişileri listelemek istediği sorulur ve gelen *input* değerine göre o kan grubuna sahip kişiler 2. ağaç üzerinde aynı renkte işaretlenerek belirlenir. Karşılaştırma yapılırken kisiler listesindeki kişilerin kan gruplarının ' (' karakterine kadar olan kısımlarına bakılır.

✓ `kisiler.get(i).kanGrubu.indexOf("(")`

Son ağacımız olan üçüncü ağaç belirli bir kişinin soyunda o kişi ile aynı mesleği yapan kişileri göstermek için kullanılmaktadır. Öncelikle kullanıcıya *"Kimin alt soyunda aynı mesleği yapanları istersiniz ?"* sorusu yöneltilir ve girilen kişinin isminden o kişinin id bilgisi alınır ardından kişinin düğümü, gerekli parametreler ile beraber altSoyGez(...) fonksiyonuna gönderilir ve girilen kişinin alt soyu elde edilmiş olur.

Daha sonra bir **for each** döngüsü ile kişinin alt soyundaki herkesin mesleği ile root kişisinin meslek bilgisi karşılaştırılır ve aynı mesleğe sahip olmayan kişilerin id bilgisi yerine ' -1 ' yazılır. Sonuç olarak içerisinde -1 ler ve gerçek id bilgilerinden oluşan liste elde edilmiş olur. Buradan sonra liste içerisi kontrol edilir ve id bilgisi liste içerisinde bulunan kişilerin **Label**'ı aynı renk ile ayarlanarak belirlenmiş olur.

Artık ekranda 3 adet ağaç görülebilmektedir. Sıra bu ağaçlar üzerinde gerçekleşen animasyonların ayarlanmasına geldi. Bunun için öncelikler iki tane **ArrayList** oluşturularak ağaç üzerinde hangi sıra ile gezeceğim kayıt edilir. Animasyonlar birinci ve üçüncü ağaç üzerinde oynamaktadır. Tüm kişiler taranır ve y değeri 1 olanlar **(kökler)** gerekli altSoyGez fonksiyonlarını gönderilerek ağaçlar üzerinde çalışan animasyonların çalışma sırası belirlenmiş olur.
Programın en sonunda bir adet **sonsuz while döngüsü** bulunmaktadır. Bu animasyonların sürekli bir şekilde devam etmesini sağlamaktadır. Öncelikle ilk ağaç için sıranın kayıtlı olduğu liste bir döngü sayesinde okunur ve düğümler sırası ile renklenir. Üstteki ağacın tümü gezildikten sonra üçüncü ağaç için gerekli liste okunarak düğümler sırası ile renklenir. Birinci ağaç üzerinde çalışan animasyon *Depth First Algorithm*, ikinci ağaç üzerinde çalışan animasyon *Breath First Algorithm* ile benzerlik göstermektedir. Program **JFrame** kapanana kadar devam etmektedir, bu sayede ekranda çalışan animasyonlar sürekli bir şekilde görülebilmektedir.Bu

metodun karmaşıklığı n$\hat{3}$'tür

Graph Sınıfı:

Bu metot graf yapısını kullanılabilmesi için oluşturuldu.private integer V değeri ve private integer değer tutan LinkedList oluşturuldu.

- Graph metodu

Parametre olarak integer bir v değeri almaktadır.Gelen değer grafın boyutunu belirlemek için kullanılmaktadır.Gelen değer V değerine atanır. Ve bir for döngüsünde her LinkedList'in her bir indexi için LinkedList oluşturulur.Bu metodun karmaşıklığı n'dir

- addEdge metodu

Parametre olarak sırasıyla integer v ve integer w değişkeni alır.Ve linklist'in v. index'ine w değeri eklenir.Bu metodun karmaşıklığı 1'dir

- BFS metodu

Parametre olarak integer s ve Kisi tipinden nesneler tutan kisiler arraylistini alır.queue adında linkedlist formatında bir kuyruk yapısı, boolean değeri alan visited adında bir dizi oluşturulur.Gelen s değeri visited[s]=true şelinde ziyaret edildi olarak eklenir.Ve gelen s değerini kuyruk yapısına ekler.leafnode adında bir değişken oluşturulup 1 değeri atandı.Eğer leafnode değişkeni 0 olursa bu kişi üvey kardeş anlamına gelmektedir.0 olduğunda s değeri, CreateTree sınıfındaki alfabetikSirala metodu çağrılır.Kuyruk boş olmadığı sürece kuyruğun başındaki değeri kaldırır ve s değerine atar.Yineleyici i değeri ,linklist[s]. lisnklistini tutar ve i'nin devamı var ise bir while döngüsüne girmektedir ve bu döngüde n değeri i'nin bir sonraki değerini tutmaktadır.Eğer visited[n] değeri false ise visited[n] true olur,kuyruğa n eklenir ve leafnode 0 olur yani üvey kardeş bulunmuş olur .BFS metodu n$\hat{2}$ karmaşıklığındadır.

- DFSUtil metodu

İnteger v, boolean visited dizisi ve Kisi tipinden nesneler tutan kisiler arraylistini almaktadır.Gelen v değeri visited dizisinin v. indexinde true olarak işaretlenir.leafnode değişkeni oluşturulup 1 değeri atanır. Yineleyici i değeri ,linklist[s]. lisnklistini tutar ve i'nin devamı var ise bir while döngüsüne girmektedir ve bu döngüde n değeri i'nin bir sonraki değerini tutmaktadır.Eğer visited[n] değeri false ise leafnode'a 0 değeri atanır ve tekrar DFSUtil metodu çağırılır.Bu döngü sonucunda leafnode hala 1 değerindeyse çocuğu olmayan düğüm bulunur.Bu metodun karmaşıklığı n'dir

- DFS metodu

Parametre olarak integer v değeri ve Kisi tipindeki kisiler arraylistini alır.Ve burada DFSUtil() metodu çağırılmaktadır.Bu metodun karmaşıklığı logn'dir

B. UML DİYAGRAMI

Kısım ektedir.[fig.11]

VI. İSTATISTIK

Kullanılan class'lar ve kullanım amaçları:

java.util
ArrayList oluşturmak,kullanıcıdan veri almak için kullanılmıştır.

java.awt
Yazı tipi, rengi gibi olayları düzenlemek için kullanılmıştır.

javax.swing
Buton, görüntü , çerçeve,.. gibi arayüzü oluşturmak için gerekli nesneleri oluşturmak için kullanılmıştır.

java.net
Görüntüleri , jar dosyasında görüntüleyebilmek için görüntülerin URL'lerini kaynak olarak ayarlar.(get.Resource("görüntü dizini"))

java.io
Dosyalara erişmek için kullanılmıştır.

VII. SONUÇLAR

Sonuç olarak kullanıcıdan alınan iki isim ile birlikte büyük olan kişinin küçük olan kişi arasındaki ilişki bulunup konsol ekranına yazdırılır, isimleri aynı olan kişiler bulunup yaşları ile birlikte ekrana yazdırılır, oluşturulan ağacın derinliği bulunup konsol ekranına yazdırılır,breath first algoritması kullanılarak üvey kardeşler bulunup sıralama adımlarını gösterecek şekilde harf sıralamasına göre sıralanıp gösterilir,depth first algoritması kullanılarak çocuğu olmayan düğümler bulunup sıralama adımlarını gösterecek şekilde yaş sıralamasına göre sıralanıp gösterilir,Kullanıcıdan alınan kan grubu bilgisi ile arayüz ekranında o kan grubuna sahip kişiler belirtilir,kullanıcıdan alınacak kişi bilgisinin alt soyunda aynı mesleği yapan kişiler bulunup hem konsol ekranına yazdırılır hem de arayüzde belirtilir,kullanıcdan alınacak olan kişi bilgisi ile o kişiye ait soy ağacı arayüz üzerinde gösterilir ve ekrana kişiler ile olan ilişkileri yazılır.Arayüz ekranında breath first algoritması ve depth first algoritmasının animasyonuda bulunmaktadır.

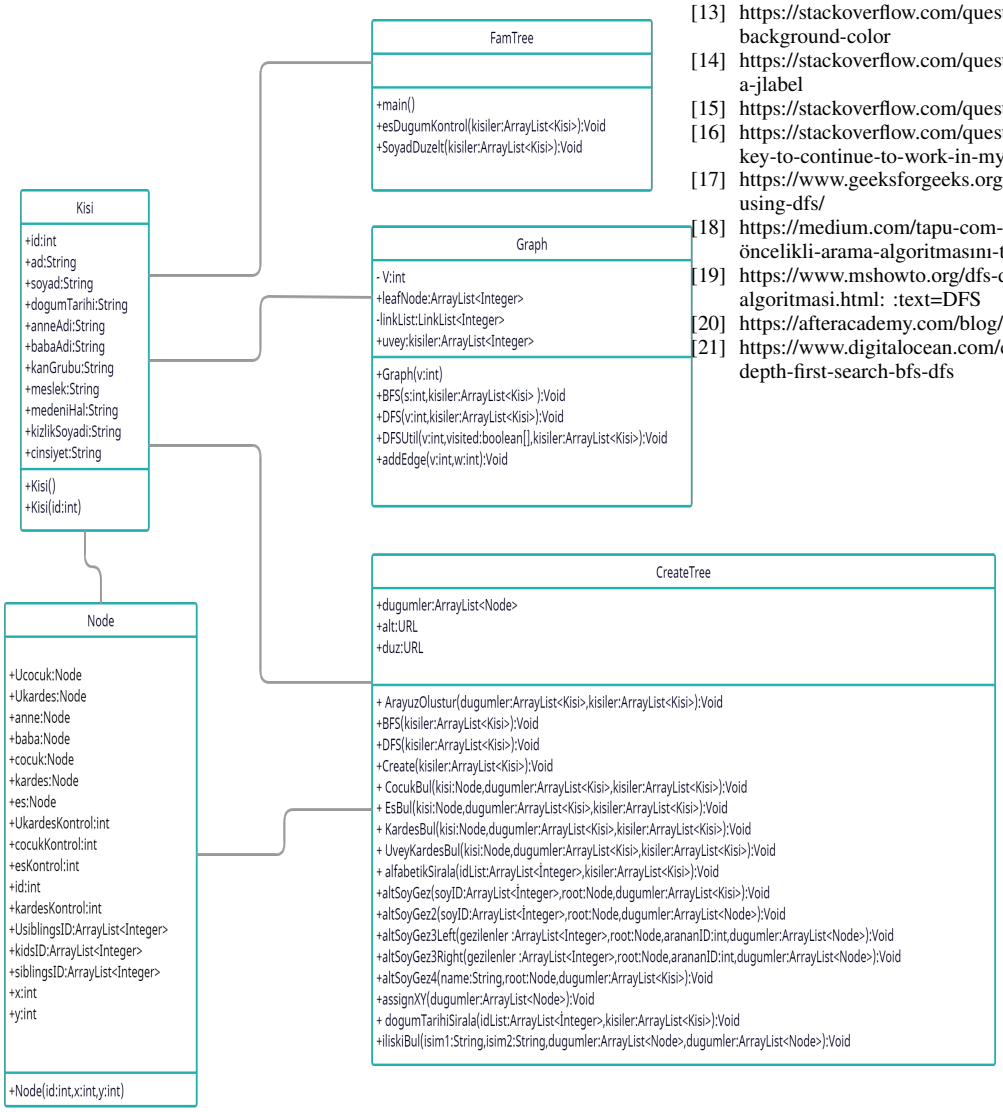

Fig. 1.

REFERENCES

[1] https://www.youtube.com/watch?v=zKDmzKaAQro (Read CSV Files)
[2] https://www.youtube.com/watch?v=p3Ct3hELjSs (Hierarchy Tree)
[3] https://www.youtube.com/watch?v=0dR-YAFFg6Ilist=PLZPZq0r-RZOMhCAyywfnYLlrjiVOkdAI1index=75 (Java HashMap)
[4] https://www.youtube.com/watch?v=eDhhSwpling (Map and HashMap)
[5] https://stackoverflow.com/questions/10126695/how-to-draw-a-tree-representing-a-graph-of-connected-nodes
[6] https://stackoverflow.com/questions/15544549/how-does-paintcomponent-work
[7] https://www.youtube.com/watch?v=1-l-UOFi1Xw (Introduction to Trees)
[8] https://www.youtube.com/watch?v=gQ3iqBh69fU (Traversing the Family Tree)
[9] https://stackoverflow.com/questions/72779630/parent-child-tree-in-java
[10] https://www.youtube.com/watch?v=wL7JOLxbMI4 (Implement Binary Tree in Java)
[11] https://stackoverflow.com/questions/35460895/adding-arraylist-of-jlabels-into-jpanel
[12] https://stackoverflow.com/questions/12119327/placing-a-jlabel-at-a-specific-x-y-coordinate-on-a-jpanel
[13] https://stackoverflow.com/questions/2380314/how-do-i-set-a-jlabels-background-color
[14] https://stackoverflow.com/questions/6810581/how-to-center-the-text-in-a-jlabel
[15] https://stackoverflow.com/questions/8586137/java-delay-wait
[16] https://stackoverflow.com/questions/19870467/how-do-i-get-press-any-key-to-continue-to-work-in-my-java-code
[17] https://www.geeksforgeeks.org/print-all-leaf-nodes-of-an-n-ary-tree-using-dfs/
[18] https://medium.com/tapu-com-bakış-açısı/bfs-breath-first-search-geniş-öncelikli-arama-algoritmasını-tanıyalım-ec7050a41af
[19] https://www.mshowto.org/dfs-depth-first-search-derin-oncelikli-arama-algoritmasi.html: :text=DFS
[20] https://afteracademy.com/blog/graph-traversal-breadth-first-search/
[21] https://www.digitalocean.com/community/tutorials/breadth-first-search-depth-first-search-bfs-dfs