Programlama Laboratuvarı I
Proje 3

Furkan Demirsoy, Önder Alp Civcir. Kocaeli Üniversitesi Bilgisayar Mühendisliği

I. ÖNSÖZ

Bu proje Kocaeli Universitesi Bilgisayar Mühendisliği Programlama Laboratuvarı I dersi 3. projesidir. Bu raporda bu projeyi nasıl yaptığımız, yaparken nelerden faydalandığımız ve karşılaştığımız hataları nasıl çözdüğümüz gibi birçok konuda bilgilendirme yapılıcaktır.

II. GİRİŞ

Bu projenin amacı, verilecek olan excel dosyalarında bulunan kişi bilgilerini kullanarak soy ağacı oluşturmak ve bu soy ağacı üzerinden çeşitli işlemler yapmaktır.

III. ÖZET

A. *Proje açıklaması*

Proje kapsamında; birçok aileye mensup kişilere ait, çeşitli bilgiler içeren 4 excel dosyası verilecektir. Verilen bu dosyalardaki ailelerin birbiriyle çeşitli şekillerde kesişimi olacaktır. Dosyaların program tarafından okunup içindeki bilgilerin kullanılarak kişiye ait soy ağacının oluşturulması projenin en temel hedefidir. Kişi için sınıf oluşturulacaktır. Kişi sınıfı tc no (id), ad, soyadı, doğum tarihi, anne adı, baba adı, kan grubu ve meslek özellikleri, kızlık soyadı ve cinsiyet ile oluşturulacaktır. Excel içerisindeki her satır ağaçta bulunacak bir düğümü oluşturacaktır.

- 1. Çocuğu olmayan düğümlerin listesinin yaş sıralamasına göre kaydedilmesi istenmektedir. Kaydetme adımında yapılan sıralama (sort) işleminin adım adım gösterilmesi beklenmektedir. (Depth First Algoritması kullanılacaktır.)
- 2. Üvey kardeşler bulunarak harf sıralamasına göre kaydedilecektir. Kaydetme adımında yapılan sıralama (sort) işleminin adım adım gösterilmesi beklenmektedir. (Breath First Algoritması kullanılacaktır.)
- 3. Kan grubu A olanların listesi kaydedilerek gösterilecektir. (Sunum esnasında kan grubu değiştirilebilir.)
- 4. Soyunda aynı mesleği yapan çocuklar veya torunlar gösterilecektir. (Baba mesleğini, dede mesleğini devam ettiren kişiler ve mesleğini devam ettirdiği ataları)
- 5. Soy ağacında aynı isme sahip kişilerin ismi ve yaşları gösterilecektir.
- 6. Kullanıcıdan alınacak 2 tane isim girdisinden sonra büyük olan kişinin küçük olan kişiye yakınlığı gösterilecektir. (Örnek: Kişinin annesinin annesinin babası gibi.)
- 7. Kullanıcıdan alınan kişi bilgisi ile o kişiye ait soy ağacının gösterilmesi istenmektedir. Verilmiş olan excel dosyalarına ek olarak

kişinin soy ağacında bulunanların kişi ile yakınlık derecelerinin yazılması beklenmektedir. (Örnek: Kişinin annesinin annesinin babası gibi.)

- 8. Soy ağacının kaç nesilden oluştuğu bulunacaktır. (Ağacın maksimum derinliği bulunacaktır.)

- 9. Kullanıcıdan alınan isim girdisinden sonra o isimden sonra kaç nesil geldiği bulunacaktır.

- 10. Yukarıdaki tüm isterler için en iyi ve en kötü senaryo karmaşıklık hesabı yapılacaktır. Rapor içinde detaylı bir şekilde anlatılması beklenmektedir.

İstenilen isterler farklı bir renkle anlaşılır şekilde gösterilecektir.Ayrıca kaydedilen veri yapısı da gösterilecektir. Kişilerin cinsiyetine göre farklı renk veya şekillerin kullanılması istenmektedir. Grafik gösteriminin nasıl olacağı konusunda herhangi bir kısıt bulunmamaktadır. Projede grafikler önemli bir yer tutmaktadır

IV. Yöntem

A. Arayüz

Arayüzün oluşturulması için java dilinde bir arayüz kütüphanesi olan swing kütüphanesi kullanıldı ve bunun için oluşturduğumuz tree veri yapısından node lerin bilgileri çekildi ve rekürsive bir fonksiyon sayesinde her seferinde root düğüm çocuklarıyla güncellenmiştir ve güncellenen her node JLabellar içerisine gelen düğümün gender bilgisi ”Kadın” ise folder içerisine verilen Ekran görüntüsü(36).png isimli dosya JLabel lar içerisine klendi .Erkekler için ise gelen nodenin gender bilgisi ”Erkek” ise mavi22.jpg isimli resim JLabellar içerisine eklendi.Ve bu ekleme jLabel13.setIcon() fonksiyonu sayesinde yapıldı Resim eklemeleri

tamamlandıktan sonra JLabellar içerisine gelen kişilerin isim bilgilerinin yazılmaları için JlABEL.setText() fonksiyonu kullanıldı.

B. EXCEL Dosyalarından Verilerin Alınması

Verilerin excel dosyalarından çekilmesi için Apache poi uygulamasından faydalanıldı bunun için bu kütüphaneye özel fonksiyonlar indirildi ve projenin library bölümüne eklendi bu eklemenin ardından sayfaları temsil eden fonksiyon getSheetat() metodu kullanılarak hangi dosyaların alınacağı bulundu ve daha sonra yine bu kütüphaneden gelen row nesnesiyle sütunların okunması,cell nesnesi ile her satırdaki cell(hücrelerin) okunması sağlandı okunan bilgiler value adında String tipinde bir çok boyutlu diziye eklendi

C. Btree Sınıfı ve Ağaç Veri Yapısının Oluşumu

Ağaç veri yapısının oluşturulması için öncelikle yapılan araştırma sonucunda Binary tree veri yapısının bir kişinin 3-4 tane çocuğu olabileceği için kullanılmasının uygun olmadığına karar verildi ve bunun için yaptığımız araştırmaların sonucunda General Tree veri yapısının kullanılmasına karar verildi bunun için Btree sınıfında human isimli bir nesne oluşturuldu ve Btree sınıfının içerisine Traverse adında bir fonksiyon oluşturuldu bu rekürsive fonksiyon içerisine parametre olarak human nesnesi sayesinde kişi sınıfından bir nesne (human.root yani kök düğümümüz) ve excel dosyasının hücrelerinden gelen ve value çok boyutlu dizisine atılan bilgiler traverse fonksiyonuna parametre olarak gönderildi bu sayede root düğümüne root düğümüne ait bilgiler gönderilmiş oldu Btree sınıfının içerisindeki

Traverse fonksiyonu içerisinde gönderilen human root düğümünün null olup olmadığına bakılır eğer null ise else içerisine girilerek gelen bilgiler kişiekle fonksiyonu içerisine gönderilir ve gönderilen bilgiler kişiekle fonksiyonu içerisinde root=new Kişi() denierek Kişi sınıfında yeni bir düğüm oluşur ve bilgiler de paramtere olarak gönderilerek Düğüm bilgileri atanır eğer gönderilen human.root null değilse (yani daha önceden oluşturulmuşsa) düğümün hangi çocuğu olduğunu tespit etmek için root.child1!=null denielrek o düğüm daha öncede

oluşturulup oluşturulmadığı kontrol edilir eğer daha önceden oluşmamışsa akrabalığın tespiti için anne veya baba adının kişinin momname,dadname bilgileriyle aynı olup olmadığı kontrol edilir eğer aynıysa rekürsive olarak yeni root.child1 düğümü traverse fonksiyonun tekrar gönderilir ve aynı işlemler bu node içinde tekrarlanır bu sayede bu sayede tüm düğümler rekürsive olarak ağaca eklenir

D. İsterlerin Gerçekleştirilmesi

- 1)Çocuğu olmayan düğümlerin bulunması: Çocuğu olmayan düğümlerin bulunması için depth first algoritması kullanıldı ve hazır veri yapısı kütüphaneleriin kullanılması yasak olduğu için Stack sınıfı oluşturuldu bu algoritma ile düğümler teker teker bulundu ve eğer root.child1 yoksa(yani 1. çocuk yoksa) o düğümlerin çocuğu yoktur denildi ve println fonksiyonu ile bastırıldı

- 2)Üvey kardeşlerin bulunması isteri: Üvey kardeşlerin bulunması için Breath First algoritması kullanıldı ve bu algoritma ile tek tek düğümler soldan sağa soy şeklinde bulundu eğer bir düğümün soyadı annesinin soyadı ile aynı değilse o kişinin üvey olduğuna karar verildi(düğümün kız olması ve evli olabilme durumu da göz önünde bulunduruldu)

- 3) Kan grubu aynı olanların listelenmesi: Kan grubu seçimi scanner sınıfının fonksiyonlarıyla alındı ve breath first algoritması ile düğümler tek tek bulundu ve kan grubu grilen kişinin kindofblood bilgisi ile aynı olup olmama durumuna göre println fonksiyonu ile bastırıldı

- 4)Soyunda Aynı mesleği Yapanlar : Soyunda Aynı mesleği yapanların bulunması için Algoritma sınıfının içerisinde fonksiyon adında bir metot oluşturuldu ve bu meto içerisinde occupation isimli değerin aynı olması durumunda println fonksiyonu ile ekranda gösterilmesi sağlanmıştır

- 6) Kullanıcıdan Alınacak isim bilgilerinin birbiriyle ilişkisini bulma: Breath first algoritmasının içerisinde düğümler tespit edilirken girilen isimlerin isim ve soyisminin bilgilerinin taranan düğümler içerisinde aynı olması durumunda bir arraylist içerisine bu düğümler atandı daha sonra atanan düğümler maxdepth fonksiyonuna gönderildi ve bu sayede o düğümden sonra kaç nesil geldiği bulundu bu nesil bilgisi büyük olanın daha yaşlı olduğu tespit edildi ve ve aradaki nesil farkına göre kişinin annesi,büyük annesi gibi soy ilişkisi tespit edildi.

- 8) Soyağacının Kaç nesilden oluştuğunun tespiti: Algoritma sınıfının içerisinde maxdepth isimli bir fonksiyon oluşturuldu ve bu fonksiyon içerisine gönderilen nesnenin bağlantılı olduğu düğümlere gider ve null olanı gördüğünde durur bu sayede kaç nesil gittiği tespit edilir

- 9)Kullanıcıdan Alınan isim girdisinden sonra o isimden sonra gelen nesil: istenilen isim breath first algoritmasının içerisinde tespit edildikten sonra maxdepth fonksiyonuna gönderilerek bu düğümden sonra kaç nesil geldiği tespit edilir

E. Deneysel Sonuçlar

her düğümün 2 den fazla düğümü(çocuğu) olabileceği için farklı bir tree veri yapısında tutulması gerektiğine karar verildi ve binary tree yerine rekürsive bir şekilde düğüm eklemesi yapan bir general tree veri yapısı oluşturuldu

Aynı soy içerisinde aynı mesleği yapan kişilerin bulunmasında sadece alt-üst soyun bulunması için preorder,inorder gibi traverse işlemlerinin işe yaramayacağı tespit edildi ve bunun için düğümler breath first algoritmasıyla bulundu

F. Sonuç

Bu proje sayesinde veri yapılarından tree, stack,queue gibi veri yapıları öğrenildi ayrıca farklı tree veri yapısı türlerinin bulunduğu ve hepsinin farklı durumlarda kullanılabilecek farklı özelliklere sahip olduğu örneğin 2 den fazla node bilgisi tutmak için n-ary tree veri yapısı veya general tree veri yapısı gibi ayrıca breath first,depth first gibi farklı açıdan traverse yapan algoritma türlerinin varlığı ve farklı algoritmaların farklı durumlarda kullanılması gerektiği öğrenildi

G. Kaynakça

https://www.geeksforgeeks.org/difference-between-general-tree-and-binary-tree/
https://www.mshowto.org/bfs-breath-first-search-genis-oncelikli-arama-algoritmasi.html
https://www.mshowto.org/dfs-depth-first-search-derin-oncelikli-arama-algoritmasi.html
https://www.javatpoint.com/java-swing

https://medium.com/@tolgahan.cepel/do

H. Karmaşıklık Analizi

- 1.İster yani çocuğu olmayan düğümlerin Karmaşıklık Analizi: Depth first algoritması içerisinde düğüm tespit edilmiştir. karmaşıklık analizi O(N) dir çünkü en iyi durumdada en kötü durumda da düğüm tree veri yapısındaki tüm düğümleri okumaktadır

- 2.İster yani üvey kardeşlerin bulunması için Karmaşıklık Analizi: Breath first algoritması içerisinde tespit edilmiştir. karmaşıklık analizi O(N) dir çünkü en iyi durumdada en kötü durumda da düğüm tree veri yapısındaki tüm düğümleri okumaktadır

- 3.İster yani Kan grubu aynı olanların listelenmesi: Kan gruplarının tespiti için breath first algoritması içerisinde tüm düğümlerin kindofblood değerine bakılmıştır bu sebepten karmaşıklık analizi:O(N) dir
- 4.İster Soyacağında aynı mesleği yapan çocuklar veya torunlar: Soyunda aynı mesleği yapan çocuklar için breath first algoritması ile okunan tüm düğümler Algoritma sınıfında fonksiyon isimli rekürsive bir fonksiyona gönderilir ve burada tespit edilir.Bu sebepten zaman karmaşıkloğı:O(N'2) dir

- 6.İster Kullanıcıdan alınacak 2 isim bilgisinin birbirleriyle olan akrabalığı:Akraba isimlerin isim bilgisinden yola çıkılarak breath first algoritması içerisinde hangi düğüme ait oldukları bulunmuş bulunan düğümlerin ağaç içerisindeki derinliği Algoritma sınıfındaki rekürsive olan maxdepth fonksiyonu içerisinde bulunmuştur bu sebepten karmaşıklık O(N'2) dir

- 8.İster Soyağacının kaç nesilden oluştuğu: Soyağacının kaç nesilden oluştuğunu bulmak için root düğümünü Algoritma sınıfındaki rekürsive olan maxdepth fonksiyonuna gönderilmiştir bu sebepten karmaşıklığı O(N) dir:

- 9.İster: Kullanıcıdan alınan isim bilgisinin hangi düğüme ait olduğunun bulunması için breath first algoritması içerisinde düğümün name bilgisine bakılarak hangi düğüme ait olduğu bulunmuştur ve düğüm bulunduktan sonra rekürsive olan maxdepth fonksiyonuna gönderilerek derinlik tespit edilmiştir bu sebepten karmaşıklığı:O(N'2) dir.