Soyağacı

Taha Furkan Genç
Öğrenci No: 210201077
Mehmet Ali Kır
Öğrenci No: 210201033

Bu proje liste tabanlı veri yapılarının (bağlı liste, kuyruk ve ağaç gibi) kullanımının daha iyi öğrenilmesi amcıyla verilmiştir. Projede istenilen verilen csv tipindeki dosyadan okunacak olan kişi ve akrabalık bilgileri ile ağaç veri yapısı kullanılarak bir soyağacı oluşturulması istenmektedir. Bu soyağacı gösteriminin grafik arayüzü ile gösterilmesi istenmektedir. Ekstra istenen işlevler ise girilen kan grubuna sahip kişilerin listelenmesi, çocuğu olmayan düğümlerin kaydedilmesi(bu yapılırken depth first algoritmasının kullanılması ve sıralama işleminin gösterilmesi), üvey kardeşler bulunarak harf sıralamasına göre kaydedilmesi(yapılırken sıralama işlemi gösterilecek ve breath first algoritması kullanılacaktır), soyunda aynı mesleği yapanların listelenmesi, aynı isme sahip kişilerin yaş ve isimlerinin listelenmesi, isim girdisine göre soy derinliği bulunması, isme göre soy ağacı oluşturulması, iki isim alınması ve yakınlıklarının gösterilmesi ve son olarak yapılacak bu isterlerin karmaşıklık hesaplarının raporda en iyi ve en kötü senaryo ile gösterilmesi istenmektedir.

I. Giriş

Öncelikle proje planlamasında önceliklerimizi belirledik. Algoritma ağırlıklı bir proje olduğundan ana sınıf içerisinde işimize yarayacak sınıfları yazdıktan sonra ana sınıfa geçip burada grafik arayüzü, isterleri gerçekleştirecek fonksiyonları ve csv dosyasını okuma işlemini yaptık. Sonraki bölümde bunları nasıl yaptığımız anlatılmakradır.

II. Kodun yazilmasi ve Yöntemler

A. Ana sınıf dışındaki sınıfların yazımı

Öncelikle zaten projenin de temeli olan ağaç yapısını oluşturmak için gereli olan düğüm yapısını oluşturmamız gerekiyor. Bunun için ana sınıf dışına ağaçdüğümü sınıfını ekliyoruz.

- Csv dosyasından alacağımız kişilerin tüm özeliklerini sınıf içerisinde tanımlıyoruz. Ondan sonra da constructer içerisinde atamaları yapıyoruz.

- Treenode 'un kenndi tipinde tanımlamalarını kendi içerisinde yer vererek iteratiflik sağlayarak ağaç veri yapısının oluşumu sağlanmıştır. Düğümler tanımlanırken kadın eşlerin ve çocukların bağlanabilmesi için TreeNode tipinde eşte değişken, çocukta (birden fazla olabileceği için ArrayList oluşturulmuştur. Bu düğümlerin tanımlama atamları constructerlarda yapılmıştır.

- Ardından önce hazır kütüphanesi ile denediğimiz kuyruk yapısını sonradan projede istenildiği üzere kendimiz bir sınıf oluşturarak kullandık. Kuyruk yapısını kullanırken bize gerekli olduğu değişken tipi TreeNode olduğundan kuyruk tipini ona göre yazdık. Bunlar ana sınıf dışında yapılan son şeylerdi.

B. Ana sınıfın yazılması

Genel olarak düğüm ve kuyruk yapısını oluşturmak dışınnda here şeyi ana sınıf içerisinde yaptık. Bu işlemler grafik tabanının oluşturulması, ister ve normalağaç işlevlerinin çalışması ve bu ikisinin birbirine entegre edilmesi olarak basitçe ifade edilebilir.

- Önce csv dosyasını okuyup verileri bölerek başladık. Veriler bölünürken TreeNode tipinde bir ArrayList e sırayla ekleniyorlar. Sonra bu listeden başka bir TreeNode tipinde listeye bu verileri sırayla çekiyoruz. Lİsteye giren verilerin id lerine göre en büyük id yi öğrenip veri sayısını alıyoruz. Böylece kisilerNode adlı asıl Node listemizin kapasitesini belirliyoruz. kisilerNode listemize de nodelist ten verileri

sırası ile çekiyoruz. Böylece artık ağaç yapımızı oluşturmaya başlayabileceğimiz veri yapılarını elde ediyoruz.

- Ardından Ekle fonksiyonu ile fonksiyonu ile düğümleri listeye eklenir.

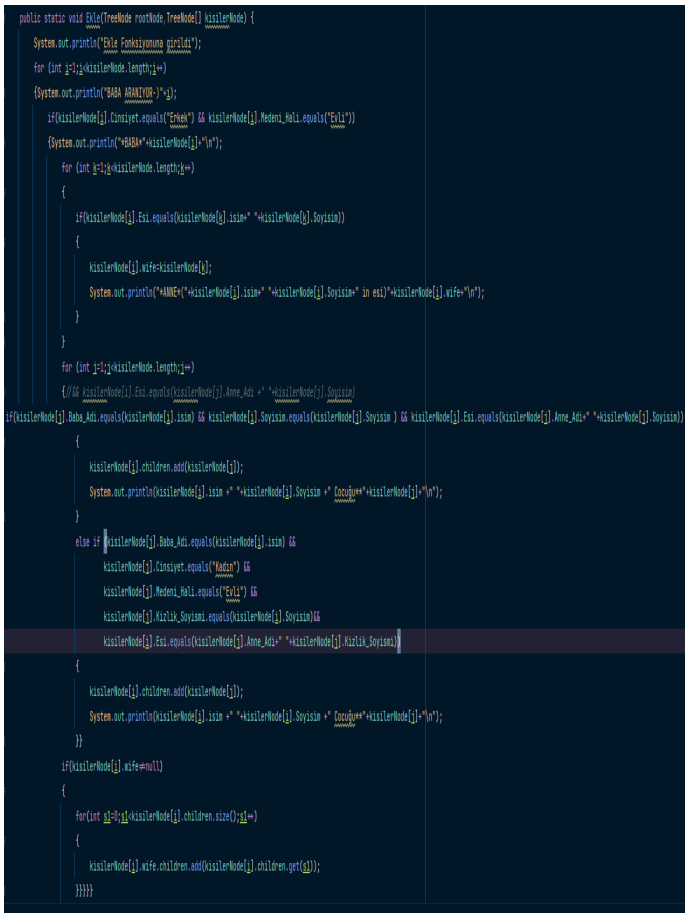


Fig. 1. Resim genişlik mech inden dolayı bozulduğundan görülmeyebilir, ekle fonksiyonunun yazımı

1) Ekle fonksiyonu:

- Fonksiyonun temeli filtrelendirme yaparak Node tiplerinin bulunmasını ve bulunan tipe göre ağaca eklene yapmasını sağlıyor. Örneğin bir çocuğu bulmaya çalıştığı zaman onun babasının adına, soyismine ve annesinibn adına bakıyor. Böylece kişiler filtrelenip yerleştirilmiş oluyor.Fig 1 de gösterilmiştir.

Eklenin ardından programın çalışmasının görünürlüğü adına ağacın basit bir modelinin konsola bastırılması için printNAryTree fonksiyonu kullanılıyor. İşleve katkısı olmasa da programın geliştirilmesinde yardımcı oluyor.

```
private static void printNAryTree(TreeNode root){
    if(root == null) return;
    Queue<TreeNode> queue = new LinkedList<>();
    queue.offer(root);
    int i=1;
    while(!queue.isEmpty()) {
        int len = queue.size();

        System.out.println("*"+"SATIR"+i+"*");
        i++;
        for(int i=0;i<len;i++) {
            TreeNode node = queue.poll();
            //assert node != null;
        System.out.print("sutun"+i+"*"+node + "\n");
            for (TreeNode item : node.children) {
            queue.offer(item);
            }
        }
        System.out.println();
    }
}
```

Fig. 2. Kuyruk veri yapısı ile temel ağacın basit bir gösterimini yapan NAryTree fonksiyonu

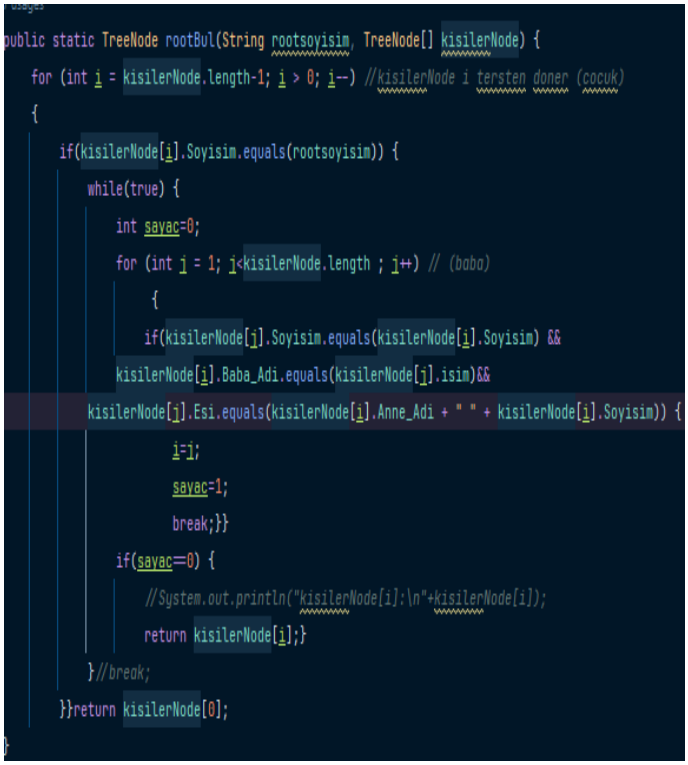
2) printNAryTree Fonksiyonu:

- Bu fonksiyonnun temel amacı ağacın ilk safhada görünürlüğünü kuyruk veri yapısı aracılığıyla yapmaktır. Kök atandıktan sonra kuyruk boş olmamak şartı ile çocuklar sırayla kuyruğa eklenmekte ve yazdırılmaktadır. Fig 2 de görülebilir.

Ardından bir döngü ile küçük ağaç yapısı konsolda gösterilmektedir. Bu ilişki çocuk ve torunu kapsamaktadır. Örnek olması amacaı ile Çelik soyadındı için bir rootBUl fonksiyonu kullanılmışıtr.

3) rootBul Fonksiyonu:

- Bu fonksiyonun işlevi ise özyinelemeli yapısıyla bize en alttan itibaren düğümlerde gezerek baba ismi ve soyisim şartıyla filtreleme yapıp root a ulaşma konusunda yardımcı olur.



```
public static TreeNode rootBul(String rootsoyisim, TreeNode[] kisilerNode) {
    for (int i = kisilerNode.length-1; i > 0; i--) //kisilerNode i tersten doner (cocuk)
    {
        if(kisilerNode[i].Soyisim.equals(rootsoyisim)) {
            while(true) {
                int savac=0;
                for (int j = 1; j<kisilerNode.length ; j++) // (baba)
                    {
                    if(kisilerNode[j].Soyisim.equals(kisilerNode[i].Soyisim) &&
                    kisilerNode[i].Baba_Adi.equals(kisilerNode[j].isim)&&
                kisilerNode[j].Esi.equals(kisilerNode[i].Anne_Adi + " " + kisilerNode[i].Soyisim)) {
                        i=j;
                        savac=1;
                        break;}}
                if(savac=0) {
                    //System.out.println("kisilerNode[i]:\n"+kisilerNode[i]);
                    return kisilerNode[i];}
            }//break;
        }}return kisilerNode[0];
}
```

Fig. 3. rootBul fonksiyonunun bulunduğu kod parçacığıdır.

Döngü veri yapısının uzunluğunun tamamını tersine dönerek bulur ve bulunduğunda sayaç değişir, bulunmazsa özyineleme devreye girer.

Bu konsol ve backend işlerinden sonra arayüz yazılmaya başlar. İşlevleri ayrı ayrı soyağacı oluşturma, kan grubu bulma, aynı isimdekileri bulma, aynı meslektekileri bulma ve çocuğu olmayanları bulma olan 5 tuş tanımlanır. Sırasıyla işlevlerin anltımına geçelim.

4) Soyağacı Oluşturma:

- Bu işlev soyagacibas fonksiyonu ile ifade edilir. Bu fonksiyon çalıştığında ise sizden soyağacını oluşturmak istediğiniz kişinin adı ve soyadını girmenizi ister. Girilen bu kişi referans alınarak kök bulunur ve ağaç gösterilir. Ayrıca derinlikbul fonksiyonuyla da ağacın derinliği belirtilir.

- derinlikbul fonksiyonunun çalışma prensibi ise bir kuyruk yapısı oluşturularak içine kökün verilmesi, sonra da köke göre sonraki nesillerin eklenmesi son olarak da o çocukların kuyruktan çıkarılması ile derinliğin hesaplanmasından oluşur.

5) Kan Grubu Aynı Olanları Bulma:

- BU işlevin çalışmasından KanGrubuBul fonksiyonu sorumludur. Çalışma şekli ise şöyledir : Sizden aramak istediğiniz kan grubu bilgisi alınır. Buna göre dosyada bulunan kan gruplarını içeren sütun gezilir ve alınan veri ile uyumlu olanlar eklenir.

6) Aynı İsimdekileri Bulma:

- Bu işlevi gerçekleştiren fonksiyon ayniismiBul fonkssyonudur. Çalışma şekli şöyledir : Önce elimizdeki verinin depolandığı kisilerNode listesinden bütün isimler alınır. Sonra string karşılaştırmalarıyla ismi aynı olanaların ismi, soyismi ve yaşı gösterilir.

7) Aynı Meslektekileri Bul:

- Bu ister aynimeslekleriBul fonksiyonu ile yapılmıştır. Şöyle çalışmaktadır : Öncelikle kisilerNode değişkeninden tüm soyisimler ve mesleklerin üzerinde işlem yapabilmek için birer ArrayList oluşturulur, meslekleri boş olanlar elenerek kalanlar meslek ArrayList ine dahil edilir, ardından alınan mesleklere göre TreeNode tipinde bir ArrayList oluşturularak buna listedeki mesleklerdekiler atanıyor, meslekler için oluşturulan diziye TreeNode tipindeki diziden meslekler atanıyor, sonra meslek kökleri için bir dizi oluşturulup her meslek için rootBul fonksiyonuyla root alınıyor, sonra tek tek her root için çocuklarının meslekleri ile eşitlikleri karşılştırılıyor ve aynı olanlar gösteriliyor.

8) Çocukları olmayanları Bul:

- Bu işlevi yerine getiren fonksiyon cocukolmayandugumsirala. Bu fonksiyon şöyle çalışıyor : Öncelikle cocuguolmayanlar isimli TreeNode tipinde bir ArrayList tanımlıyoruz. Ardından bekar olanlar direkt listeye eklenip evli olanlar koşullara girmeye başlıyor. Erkeklerden isimleri baba adı olarak geçenler ve eşinin ismi anne olarak geçenler eleniyor. Kadınlarda ise isimleri anne adı olarak geçenler ve eşinin simi baba olarak geçenler eleniyor. Bu elemelerden sonra ise siralilar adlı bir TreeNode listesi oluşturuluyor ve

cocuguolmayanlar listesindekiler buraya ekleniyor. Sonra yasBul fonksiyonu ile (sonraki madde bu fonksiyonu açıklamaktadır) listedekilerin yaşı bulunarak sıralanmaktadır.

- yasBul fonksiyonu şöyle çalışmaktadır : Doğum tarihi bilgileri listeden alınır, Local Date tipinde alınan bu veri ile bugünün tarifi arasındaki period bulunur ve bu da bize yaş bilgisini verir.

III. Deneysel sonuçlar

Fig. 4. Programın aynı isimdeki kişileri grafik arayüzünde sıraladığı örnek

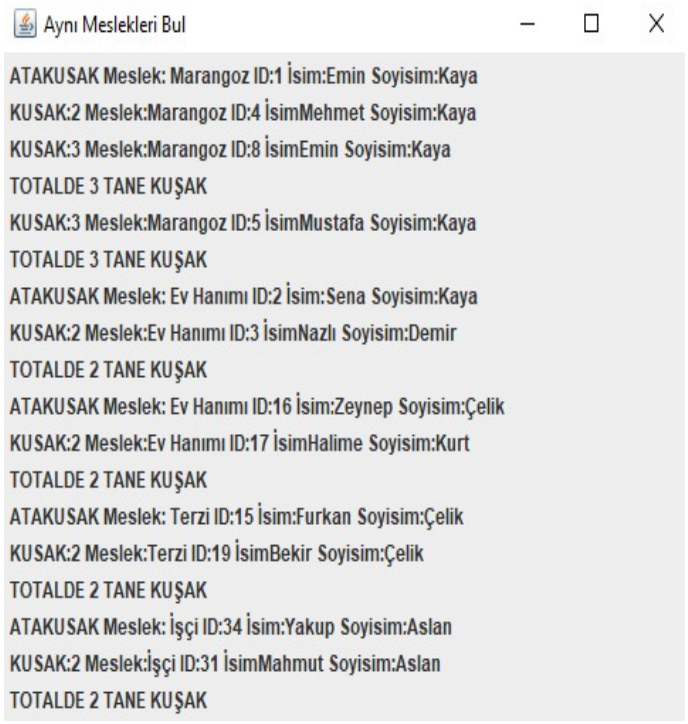


Fig. 5. Programın grafik arayüzünde soyağacında mesleği devam ettirenleri gösterdiği örnek

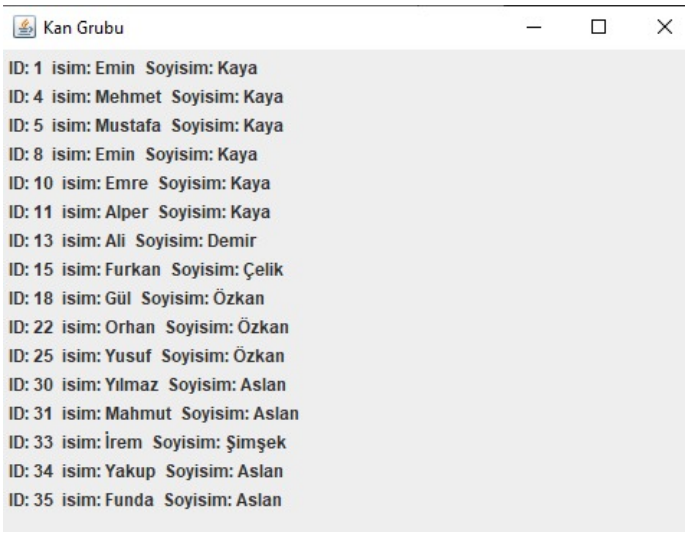


Fig. 6. Programın kan grubu 0 olanları grafik arayüzünde yazdırdığı örnek

Fig. 7. Programın kan grubu A olanları grafik arayüzünde yazdırdığı örnek

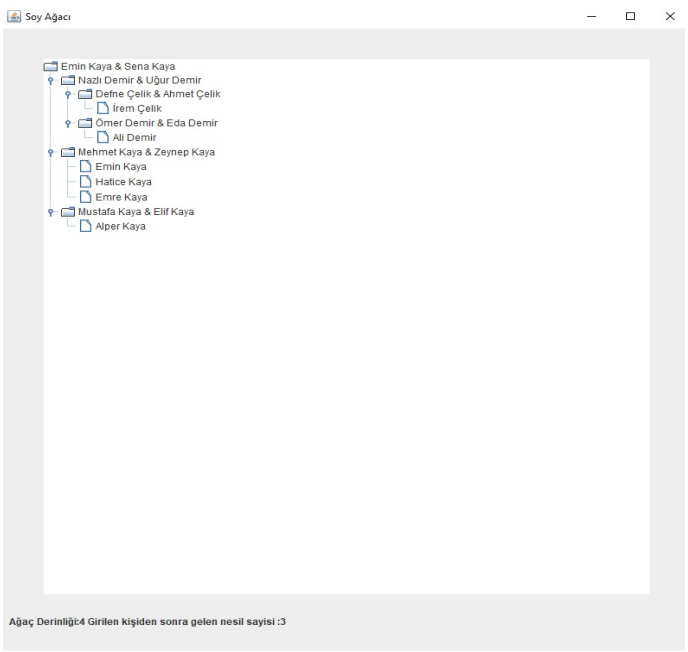


Fig. 8. Girilen isim ve soyisme göre programın soyağacı yazdırdığı örnek

Fig. 9. Çocuğu olmayan düğümlerin ID ve yaş çıktılarını grafik arayüzünde yazan örnek

A. İsterlerin Yöntemlerinin Karmaşıklık Hesapları

Bu bölümde raporda istenen

1) Çocuğu olmayan düğümlerin kaydedilmesi : bir for döngüsü ile tüm düğümleri dönüyoruz. o(n) karmaşıklı bir for döngüsü şuan elimizde. ardından içindeki koşul aramasından ”medeni hali:evli” sonucunu alırsak evli node'un çocuğunun olup olmadığını bir for döngüsü ile daha arıyoruz. Böylelikle iç içe Linear aramamızın karmaşıklık sonucunu şimdilik ”o(n*n)” buluyoruz. Daha sonrasında bu düğümleri sıralamak için iç içe 2 for döngüsü tanımlamamız gerekirken bu işlemi her düğümü tek tek vermesi ve animasyonik bir görüntü oluşturmak için 2 for döngümüzü içine alan bir for tanımlıyoruz ve algoritma karmaşıklığımız ”o(n*n*n)” oluyor.

2) Üvey Kardeşlerin Kaydedilmesi: Bu işlevi yerine getiremediğimizden karmaşıklık hesabı konusunda elimizde yeterli bilgi bulunmamaktadır.

3) Kan Grubuna Göre Kişi Listesi: Kan grubu bilgileri kisiNode değişkeninden alındığından dolayı önce oluşturulan yapının karmaşıklığı hesaplanır. kisilerNode iki kez for a girmiş (O(n2)) ve sonra Ekle fonksiyonu kullanılmıştır. Ekle fonksiyonunda ise sırayala for if for if yapıalrına girmiştir(for lineer artış yapmadığında ayrıca belirtilir, buradakiler linneer artış yapmaktadır). Onlar da O(n*n) karmaşıklığı yaratır. Son olarak KanGrubuBul fonksiyonu içerisinde ise bir for ve if durumuna girerek karmaşıklığı tamamlamış olur.Karmaşıklığı döndüğü normal forların (iç içe olmayanlar karmaşıklığı değiştirmez) miktarına göre hesaplanır(O(n*n)).

4) Soyda Aynı meseleği yapanların listelenmesi: kisilerNode yapısının genel karmaşıklığı dışında fonksiyonun kendi içerisinde karmaşıklık sağlayan içiçe or sayısı maksimum 2 olduğundan kisilerNode un karmaşıklığıyla birlikte O(n*n*n*n) olarak hesaplanabilir.

5) Aynı İsme Sahip Kişilerin Listelenmesi: Burada ise kisilerNode un karmaşıklığı (O(n*n)) dışında fonksiyonun kendi içeriisindeki krmaşıklığı içiçe bulunan 3 for dan dolayı O(n*n*n) oluyor. Dış if ve for larda da içiçelik olmadığından etki etmiyor. Sonuç olarak karmaşıklık O(n*n*n) olabilir.

6) Girişe göre yakınlık belirleme: Bu işlevi yerine getiremediğimizden karmaşıklık hesabı konusunda elimizde yeterli bilgi bulunmamaktadır.

7) Girişe göre Soyağacı oluşturulması:

8) Soyağacı Nesil Sayısı:

9) Girişe göre Sonraki Nesil Sayısını Bulma: Bu üç durumun karmaşıklık yönünden benzerliğinden doalayı genel anlamda karmaşıklığa neden olan derinlikBul un algoritması anlatılacaktır. derinlikBul da ise özyinelemeli yapıda ağaç gezilmesi yapıldığından dolayı karmaşıklığı O(logn) dir ve soyagaci fonksiyonu içierisindeki for dan dlayı da O(n) karmaşıklık eklenecektir. Genel karmaşıklık O(nlogn)

IV. Sonuç

Genel anlamda algoritma odaklı olan bu proje zaman ve planlamanın özellikle de backend in frontend e yedirilmesi konusunda önemli bir örneği olup frontend konusunda bizim projemniz çok şey vaad etmese de programın genel yapısı anlamında (algoritmanın karmaşıklığının parçalanıp modülerleştirilerek çözümlenmesi) temel gereklilikleri yüksek miktarda yerine getirmektedir.

Kaynakça

https://www.javatpoint.com/how-to-read-csv-file-in-java
https://stackoverflow.com/questions/64488594/reading-from-csv-file-and-create-object
https://github.com/Taaqif/Family-Tree-Application/blob/master/TreeGUI.java
https://www.studytonight.com/advanced-data-structures/nary-tree
https://stackoverflow.com/questions/44994171/java-tree-data-structure-with-multiple-nodes-how-to-search-efficiently
https://chat.openai.com/chat

https://www.geeksforgeeks.org/generic-tree-level-order-traversal/
https://stackoverflow.com/questions/67603756/how-to-implement-a-tree-data-structure-with-multiple-roots
http://www.java2s.com/ref/java/java-jtree-extend-to-create-family-tree.html
https://stackoverflow.com/questions/9090202/multiple-cursors-for-netbeans
https://www.geeksforgeeks.org/how-to-create-array-of-objects-in-java/
https://mameko.gitbook.io/pan-s-algorithm-notes/my-notes/25-trie/1506-find-root-of-n-ary-tree
https://stackoverflow.com/questions/63587809/the-array-is-only-written-to-never-read-from-error-showing-in-java-code
https://bizneyapiyoruzki.com/2018/20/yazilim/dizileri-kopyalama-yontemi-2-yontem-java/
https://www.reitix.com/merak/java-da-iki-islem-arasinda-bekleme-eklemek/facc28aa

Bazı fonksiyonların resimleri pdf içerisine sıkıştırmada olan bozulmadan dolayı eklenmemiştir.