PROGRAMLAMA LABORATUVARI 1
PROJE 2

TARIK GÖREN
Bilgisayar Mühendisliği
200202022@kocaeli.edu.tr

ISHAK ERDOĞAN
Bilgisayar Mühendisliği
200202035@kocaeli.edu.tr

ÖZET

Bu doküman Programlama Laboratuvarı 1 dersi 2. Projesi için hazırlanmıştır. Dokümanda projenin tanımı, çözüm yöntemi, kullanılan kütüphaneler gibi programın oluşumunu açıklayan başlıklara yer verilmiştir. Doküman sonunda projemizi hazırlarken yararlandığımız kaynaklar bulunmaktadır.

I. GIRIŞ

Bu projede bizden C dilini kullanarak sonek ağaçları(suffix tree) ve sonek dizileri (suffix array) kullanarak katarlar üzerinde bazı aramalar yapmamız istenmiştir.

Bir katarın (p) başka bir katar (s) içinde bulunması aslında p'nin s'in herhangi bir sonekinin öneki olmasını gerektirir. Bunun için s katarının tüm sonekleri dizilerde toparlanıp 'p katarı bu soneklerin öneki mi?' diye sorgulamamız gerekir. Fakat bu yöntem çok uzun karakter sayılı stringler için biraz pahalı(algoritmik karışıklık olarak) geldiği için sonek ağaçları geliştirilmiştir.

Bu proje kapsamında metin dosyasından almış olduğumuz string dizisine (s katarı) birtakım işlemler yaptırıp aşağıdaki problemlere yanıt aramamız istenmiştir.

1) s katarı için sonek ağacı oluşturulabilir mi?
2) Sonek ağacı oluşturulan bir s katarı içinde p katarı geçiyor mu, geçiyorsa ilk bulunduğu yerin başlangıç pozisyonu nedir, kaç kez tekrar etmektedir?
3) Sonek ağacı oluşturulan bir s katarı içinde tekrar eden en uzun altkatar nedir, kaç kez tekrar etmektedir?
4) Sonek ağacı oluşturulan bir s katarı içinde en çok tekrar eden altkatar nedir, kaç kez tekrar etmektedir?

II. YÖNTEM

A. Algoritma

Öncelikle oluşturacağımız sonek ağacının nasıl bir yapıda olacağından bahsedelim. Soneklerimizi bir bağlı liste (linked listed) üzerinden her düğüm bir harfi karşılayacak şekilde bir yapı oluşturduk. Şekil 1'de de görüldüğü gibi her düğümden 53 adet yaprak çıkabilmekte. Bunun sebebi 53 farklı karakter bulunması.

```
struct Agac
{
    struct Agac *yaprak[53];
    char harf[1];
    int dallanmaSayisi;
};
```

Şekil 1. Struct

İlk olarak okunan string'in sonek ağacı çizilebilirmiyi sorgulamamız gerekiyor (agacOlusturulabilirMi fonks. ile). Eğer oluşturulabiliyorsa sonek dizimizdeki (suffixArr[][]) soneklerimizi agacOlustur() fonksiyonu vasıtasıyla ağacımıza ekliyoruz. Her sonek teker teker agacOlustur() fonksiyonuna gönderilmektedir. İlk gönderilen sonek liste boş olduğu için direkt root düğümünden bir yaprak çıkar. Daha önce de belirttiğimiz gibi her yaprak bir harfi temsil etmekte olduğundan ilk soneğimizin ilk harfinden başlayarak listeye ekleme yapar. Structta tanımlanan Agac tipindeki yaprak[53] dizisinin her indisini unique olarak belirledik. (kucuHarf_INDEX , buyukHarf_INDEX ve dolar$INDEX komutlarını define ederek her karakterin ASCII karşılığını buldurduk.) Örneğin yaprak[0]= 'a' , yaprak[26]= 'A' ve yaprak[52]= '$' karakterlerini temsil etmekte. Bu şekilde her soneğin her karakteri teker teker listeye ekleme yapılır, ekleme yapılırken şöyle bir yol izlenir.

agacOlustur() fonksiyonuna gönderilen soneğin ilk harfinden başlayarak ve aynı zamanda root düğümünden başlayarak düğümün yaprağında soneğin ilk harfi mevcut mu diye sorgulama yapılır. Daha sonra mevcut ise dallanmaSayisi 1 (düğümün kaç kere tekrar ettiğini tutmak için) arttırılır ve diğer harfe geçilir. Diğer harflerde de böyle devam eder. Farklı olduğu tespit edildiğinde farklı olan düğümden yeni bir düğüm çıkarılır. Bu şekilde döngü sonlanana kadar ağaç oluşturulmaya devam edilir.

Ağaç oluşturulduktan sonra diğer isterler gerçekleştirilebilir hale gelir. Ağaçta kullanıcıdan alınan katar bulunuyor mu, bulunuyorsa kaç kere tekrar ediyor sorusuna katarBul() fonksiyonunda yanıt verdik. Öncelikle kullanıcıdan almış olduğumuz katar ve başlangıç düğümümüz (root) katarBul() fonksiyonuna gönderilir. Örnek olarak "XABXAC$" ağacında "AB" katarını aradığımızı ele alalım. "AB" kelimesini karakter karakter gezerek sırasıyla şu işlemleri yaparız. "A" karakteri büyük harf mi yoksa küçük harf mi diye sorgulama yaparız büyük olduğu için buyukHarfINDEX komutu çalışır ve [26+ "A" - "A"] işlemi gerçekleştirilip index(int) değişkenine "A" nın alabileceği indis değerini (26) eşitleriz. Sonra bulunduğumuz düğümdeki yaprak[26] indisi NULL değerden farklı çıkacağından diğer harfe geçilir. "B" karakterine de [26+"B" - "A"] işlemi gerçekleştirilir ve index olarak 27 döner. Bu sefer yaprak[27] sorgulanır tekrar NULL'dan farklı çıkar ve döngü durmadan sonlanır. Bu aradığımız katarın ağaçta bulunduğuna işaret olur. Çünkü aradığımız katardaki herhangi bir harf indeksi yapraklarda bulunmasaydı fonksiyon sonlanacaktı ve bulunamadı yazısı çıkacaktı. Bulunduğu için iterator (bağlı listedeki gezici değişkeni) pozisyonundaki düğümün dallanmaSayisi ekrana yazdırılır ve ne kadar tekrar ettiği bulunmuş olur.

Oluşturduğumuz sonek ağacında tekrarlayan en uzun katar ve kaç kere tekrarladığını buldurma problemine tekrarEdenEnUzunKatar() fonksiyonunda yanıt verdik. Bu fonksiyonda önceden oluşturduğumuz sonek dizilerini katarBul() fonksiyonundaki indeks mantığı ile başlangıç düğümünden başlayarak arama yaptırırız. Arama yaptırırken geçilen düğümlerde dallanmaSayisi>1 kontrolü yaparız. Eğer büyükse diğer düğüme geçer büyük değilse dallanma yapmıyor demektir ve bir önceki düğüme kadar olan kısmı temp(char) değişkenine eşitler. Eğer karakter sayısı olarak temp, enUzun(char)'dan büyük ise enUzun temp'e eşitlenir ve yaprakSayisi değişkeni de tempin son harfinin bulunduğu düğümdeki dallanmaSayisi'na eşitlenerek o katarın ne kadar tekrar ettiğini saklamış oluruz. Sonek dizisindeki tüm kelimelere bu işlemler uygulanır. Ağacımızdaki tarama bittikten sonra enUzun değişkeninde saklı olan katar ve yaprakSayisi değişkeninde saklı olan tekrar sayımız ekrana bastırılır ve ister gerçekleşmiş olur.

Oluşturduğumuz sonek ağacında en çok tekrarlayan altkatar buldurma problemine enCokTekrarEdenKatar() fonksiyonunda yanıt verdik. Bu fonksiyon da önceden oluşturduğumuz tekrarEdenEnUzunKatar() fonksiyonundaki gibi sonek dizimizdeki soneklerimiz tarama yaptırılır. Tarama esnasında sorumuzda da istendiği gibi dallanmaSayisi>1 (dallanma yapıyor mu) ve iter(gezici)'nin gösterdiği dallanmaSayisi>=yaprakSayisi kontrolü gerçekleştirilir. Eğer kontrolden geçiyorsa temp(char) değişkenimize bulunduğumuz düğümün olduğu konuma kadarki karakterler eklenir ve yaprakSayisi iter'in dallanmaSayisi'na eşitlenir. Bu şekilde ağacın tamamı sorgulanarak en büyük dallanmaSayisi'na sahip katarımız enCokTekrar(char) ve bu katarın dallanmaSayisi da yaprakSayisi(int) değişkenine eşitlenerek bu değişkenler ekrana bastırılır. Böylelikle bu isterimiz de gerçekleşmiş olur.

B. Geliştirme Ortamı

Proje standart C dilinde geliştirilmiştir. Windows sitemde geliştirilip IDE olarak Dev C++ kullanılmıştır. Dev C++ uygulamasında 'TDM-GCC 4.9.2 64-Bit Release' modda derlenip çalıştırılmıştır. Projeyi çalıştırmadan önce projenin bulunduğu klasörde 'strFile.txt' adında tek satırlık bir metin dosyası oluşturulması gerekmektedir. Aksi takdirde dosya bulunamayacaktır ve program istenilen şekilde çalışmayacaktır.

C. Kütüphaneler ve Tanımlamalar

Kullandığımız kütüphaneler ve ne için kullandığımız aşağıdaki gibidir:

<stdio.h> : input ve output almak için
<stdlib.h> : malloc gibi fonksiyonları kullanmak için
<string.h> : strcmp,strcpy gibi string üzerinden işlem yapan fonksiyonları kullanmak için
<ctype.h> : islower fonksiyonunu kullanmak için (küçük,büyük harf kontrolü)

Gerçekleştirdiğimiz tanımlamalar ve işlevleri aşağıdaki gibidir:

kucukHarf_INDEX(c) ((int)c - (int)'a') : c bir karakter tutar. c ve 'a' karakterinin int tipindeki dönüşümlerinin (type casting) farkını bulan önişlemci komutu.

buyukHarf_INDEX(c) (26 + (int)c - (int)'A') : c bir karakter tutar. c ve 'A' karakterinin int tipindeki dönüşümlerinin (type casting) farkını bulup 26 ile toplayan önişlemci komutu.

D. Yalancı Kod

- Dosya kontrolü yap eğer dosya yoksa "String dosyası oluşturulmamış" eğer varsa da okunan stringi yaz.

- Ana menüden seçim yap [1-4 arası olsun.çıkış için (-1)]

- Eğer seçim 1 ise s katari icin sonek agaci olusturulabilir mi . karar 1 ise "Bu string ile agac olusturulamaz.Lutfen stringin sonunda '$' ekleyiniz."

- Eğer kontrol sıfır değilse " Bu dizgi ile agac olusturulabilir."

- Eğer seçim 2 ise Klavyeden girilen p katari s katari icinde geciyor mu? Geciyor ise kac kez tekrar ediyor "String giriniz" yaprak eğer null ise "Girilen katar bulunamadi" aksi halde "Girilen katar bulundu! "Tekrar Sayisi..." diye yazar.

- Eğer seçim 3 ise Sonek agaci olusturulan bir s katari icinde tekrar eden en uzun altkatar nedir, kac kez tekrar etmektedir? temp ve enUzun adında iki tane string değişkeni oluştur.eğer tempBoyut>enUzunBoyut ise enUzun'a temp değişkenini kopyala, değilse bir şey yapma

- En Uzun Tekrar Eden Katar ve Tekrar Sayisını yaz

- Eğer seçim 4 ise Sonek agaci olusturulan bir s katari icinde en cok tekrar eden altkatar nedir,kac kez tekrar etmektedir? temp stringi oluştur yaprağın dallanma satısı>1 ve yaprağın dallanma sayısı>yaprak sayısı ise teker teker harfleri tempe kopyala değilse döngüden çık

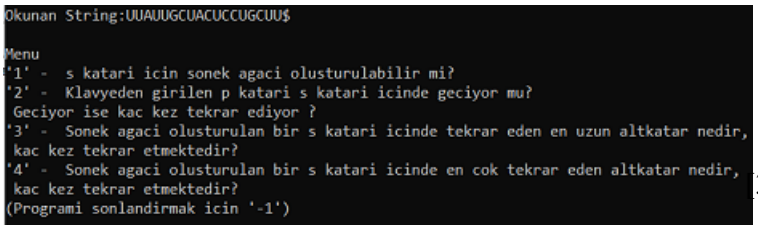
- En Cok Tekrar Eden Katar ve Tekrar Sayisını yaz

- Eğer seçim -1 ise algoritmayı sonlandır. Eğer seçim 1-5 arası veya -1 değilse kullanıcıya "Gecersiz giris" diye yaz.

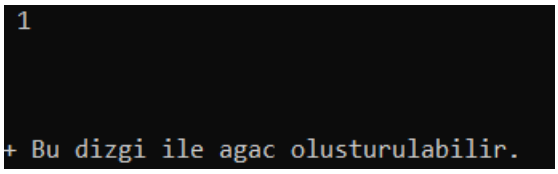
III. Sonuçlar

Bizden istenen tüm problemlere doğru cevaplar verilmiş fakat grafik çizdirme tarafında karşılaştığımız sorunlar dolayısıyla isterlerin grafiği eklenememiştir. Grafik dışında herhangi bir eksik yoktur.
Proje sonunda Suffix Tree algoritmasının çalışma mantığı, ağaç yapıları hakkında detaylı bilgiler ve bağlı listeler üzerinde işlem yapabilme (arama,silme,bulma,gezinti yapma gibi) öğrenildi.Gerekli testler yapılarak programın runtime hataları vermesi önlendi.

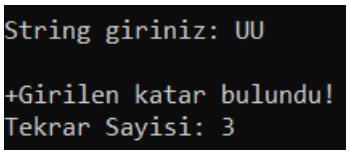
Program Çıktıları



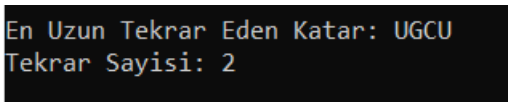
Şekil 2. Ana Menü



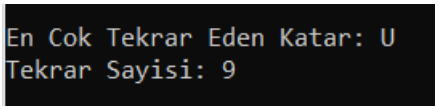
Şekil 3. Seçim 1



Şekil 4. Seçim 2



Şekil 5. Seçim 3



Şekil 6. Seçim 4

Kaynaklar

[1] Suffix Tree Algoritması
https://www.koseburak.net/blog/suffix-tree/
https://www.cs.cmu.edu/~ckingsf/
bioinfo-lectures/suffixtrees.pdf

[2] Bağlı Listeler (Linked Listed)
https://www.youtube.com/watch?v=
r3uOBb3BM-0
https://bilgisayarkavramlari.com/2007/05/03/
linked-list-linkli-liste-veya-bagli-liste/

[3] Ağaçlar (Tree)
https://www.youtube.com/watch?v=j9egu_
VB1j4

[4] Bazı unutulmus kavramlar için;
Onur Gök ve Suhap Sahin Programlama Dersi Slaytları

[5] Veri Yapıları için yazılı kaynak;
Onur Gök ve Suhap Sahin Veri Yapıları ve Algoritmalar Dersi Slaytları