Sonek Önek Ağaç Projesi

Ömer Emircan Ayvaz
200202009

Engin Tosun
200202028

I. Özet

Bu rapor Programlama Laboratuvarı I Dersinin 2. Projesini açıklamak ve sunumunu gerçekleştirmek amacıyla oluşturulmuştur. Bu proje C dilinde Code:Blocks ortamında geliştirilmiştir.Raporda projenin tanımı, özet, yöntem,karşılaşılan sorunlar ve çözümler,sözde kod, sonuç bölümünden oluşmaktadır. Proje aşamasında yararlanılan kaynaklar raporun son bölümünde bulunmaktadır.

II. Proje Tanimi

Bu projenin amacı sonek ağaçlarını ve sonek dizilerini kullanarak katarlar üzerinde bazı arama işlemleri yapmaktır.

Bir katarın (p) başka bir katar (s) içinde bulunması aslında p'nin s'in herhangi bir sonekinin öneki olmasını gerektirir. Örnek olarak ili katarı bilişim katarının içinde bulunup bulunmadığı bulmak için bilişim katarının tüm sonekleri oluşturulur ve ili katarının bu soneklerin herhangi birinin öneki olup olmadığına bakılır. Yukarıda listelenen soneklere bakıldığında ili katarı 2. sonekin önekidir ve ili katarı bilişim katarının içinde yer alır. Aynı arama ila katarı için yapıldığında ise, ila katarı herhangi bir sonekin öneki olmadığı için bilişim katarı içinde yer almaz. Bu yaklaşımla bir katar içinde (uzunluğu n karakter olsun) başka bir kararı (uzunluğu m olsun) bulmak karmaşıklığı O(n+m).

Katarlar ne kadar uzun olursa olsun sınırlı sayıda farklı karakterin birleşmesiyle oluşur. Örnek olarak çok fonksiyonlu bazı proteinlerin uzunlukları binlerce karakter olabilirken bir protein dizilimleri 20 farklı karakterden oluşur. Böyle çok uzun katarlar içinde çok kısa birçok katarı aramak yukarıda anlatılan temel yöntemi kullanılarak yapılması pahalıdır. Ancak soneklere dayalı bazı veri yapıları kullanılarak arama işlemi çok daha ucuza (algoritmik karmaşıklık olarak) yapılabilir. Bu amaçla sonek ağaçları ve sonek dizileri geliştirilmiştir.

n uzunluklu s kataın sonek ağacı aşağıdaki özelliklere sahiptir:
• Ağacın 1'den n'e kadar numaralandırılmış n adet yaprağı vardır.
• Kök dışında her düğümün en az iki çocuğu vardır.
• Her kenar s'in boş olmayan bir altkatarı ile etiketlenir.
• Aynı düğümden çıkan kenarların etiketleri farklı karakter ile başlamalıdır.
• Kökten başlayıp k. yaprağa giden yoldaki kenarların etiklerinin birleştirilmesi ile k. sonek elde edilir.

Bu proje kapsamında sonek ağaçları kullanılarak aşağıdaki problemler çözülecektir:
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?

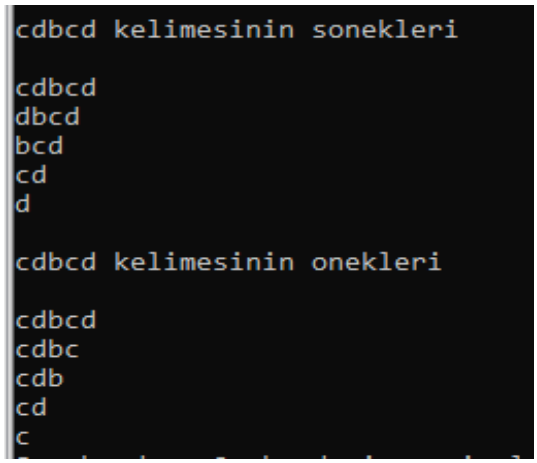
III. Yöntem

Algoritma tasarımı ve projenin kodlama kısmına geçmeden önce suffix, prefix, suffix tree, katar,altkatar gibi terimler hakkında daha fazla bilgi edinmek için konu başlıklarına farklı kaynaklardan çalışıldı.Proje ilerledikçe gerekli yerlerinde lazım olabilecek hazır fonksiyonlar araştırıldı ve kodda uygulandı (strcmp,strstr,strcpy gibi) . Girdi olarak alanan string verisinin ve işlemlerin doğruluğu konsol üzerinden çokca yerde printf ile kontrol edildi ve hatalı durumlarda düzenlemelere gidildi.

İlk olarak değeri elle konsola girerek daha hızlı bir şekilde test yapılabileceği için değeri scanf ile almak için kod oluşturuldu. İstenen problemleri switch case içine atarak her çözüme bir sayı verip çözülmesi istenen problemin kullancıdan girilmesini sağlandı.

1-) s katarı için sonek ağacı oluşturulabilir mi?

Sonek ağacı oluşturma şartı kelimenin kendisi haricinde kelimenin oluşan sonek ve öneklerinin herhangi birinin birbirine eşit olmaması ile sağlanır.örnek olması açısından cdbcd için oluşacak suffix ve prefixler bu şekildedir:



Görüldüğü üzere hem sonek hem de öneklerinde ”cd” ortak olduğundan dolayı sonuna kelimede kullanılmayan özel bir sembol eklemeden ağaç yapısı bu kelime için oluşturulamaz.Kodda bu problemi çözmek için girilen kelimenin hem soneklerini hem de öneklerini ayrı matrislere atandıktan sonra konsolda kullanıcının da sonekleri ve önekleri görmesi için konsola tüm sonek ve önekler yazdırıldı ardından kelimenin kendisi haricinde kalan sonek ve öneklerini strcmp ile döngü içinde karşılaştırıldı. Eşitlik durumunda ağaç oluşturulamayacağı yazdırılıp programdan return ile çıkıldı. Tüm değerler için eşitsizlik durumunda ise ağaç oluşturulabileceği yazdırılıp programdan return ile çıkıldı.

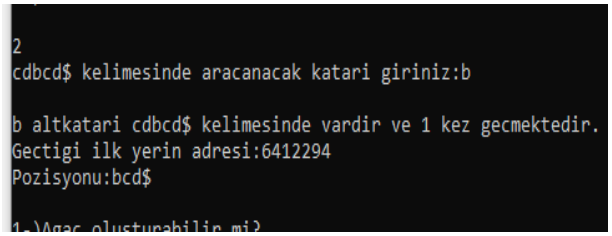
2-) Sonek ağacı oluşturulan bir s katarı içinde p katarı geçiyor mu, geçiyorsa ilk bulunduğu yerin başlangıç pozisyonu nedir, kaç kez tekrar etmektedir?

Bu problem için pdf de örnek olarak verilen şu yöntem kullanıldı:
Bir kataın (p) başka bir katar (s) içinde bulunması aslında p'nin s'in herhangi bir sonekinin öneki olmasını gerektirir. Örnek olarak ili katarı bilişim katarının içinde bulunup bulunmadığı bulmak için bilişim katarının tüm sonekleri oluşturulur ve ili katarının bu soneklerin herhangi birinin öneki olup olmadığına bakılır. Yukarıda listelenen soneklere bakıldığında ili katarı 2. sonekin önekidir ve ili katarı bilişim katarının içinde yer alır. Aynı arama ila katarı için yapıldığında ise, ila katarı herhangi bir sonekin öneki olmadığı için bilişim katarı içinde yer almaz. Bu yaklaşımla bir katar içinde (uzunluğu n karakter olsun) başka bir kararı (uzunluğu m olsun) bulma karmaşıklığı O(n+m).

Bundan yola çıkarak girilen string için sonek oluşturduk oluşan sonekler için de önekler oluşturuldu örnek olarak ”banana” kelimesinin soneklerinin önekleri şu şekildedir:

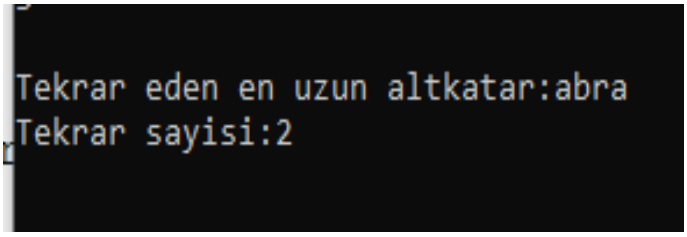
banana banan bana ban ba b
anana anan ana an a
nana nan na n
ana an a
na n
a

Soneklerin öneklerini oluşturmak için ayrı bir char değişkenine stringi atandı. sonra döngü içinde sonekleri oluşturduktan sonra if bloğu ile aranan katara eşit olup olmadığına bakıldı. Eşitlik durumunda başlangıç değeri 0 olan sayacı arttırıldı. Döngü bittikten sonra strstr kullanarak aranan katarı başka bir char pointerının içine başlangıç pozisyonu atandı. if bloğu ile sayacın sıfırdan büyük olduğu durumda stringde bulunduğunu printf ilk başladığı yerden stringin sonuna kadar stringi ve bulunduğu ilk bölgenin adresini yazdırılıp program bitirildi. Sayacın sıfırdan büyük olmama durumunda ”Katar Bulunamadı.” şeklinde bir hata kodu yazdırıldı.



3-) Sonek ağacı oluşturulan bir s katarı içinde tekrar eden en uzun altkatar nedir, kaç kez tekrar etmektedir?

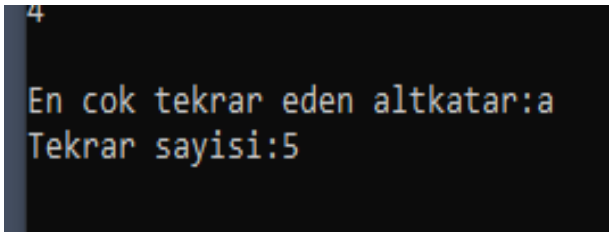
Bu problemde istenen şey en uzun altkatar olmasının yanısıra tekrar etmesi tekrar durumu 2 ve 2 den fazla durumlarda bu problemin istenileni sağlanılabileceğinden bunu bulurken öncelikle int max uzunluk,max,sayac ve kelimeyi tutmak için char arrayı tanımlandı. Tekrar stringin soneklerinin oneklerini bir matrise atandı. Başka bir döngü işleminde iç içe döngü oluşturup her sonekin önek değerini tüm sonek önek değerleri karşılaştırıldı eşitlik durumlarında sayacı arttırıldı. İçteki döngüden çıkınca if bloğu kullanarak altkatar uzunluğu max uzunluk'tan büyük olan ve sayacı 1 den büyük olan durumlarda yeni max değerini sayac sayısı ile char arrayi ise bu sonekin önek değeri olarak güncellendi ve döngü bitince çıkan sonucu konsoldan yazdırıldı.
”abracadabra” stringi için tekrar eden en uzun altkatar:



4-) Sonek ağacı oluşturulan bir s katarı içinde en çok tekrar eden altkatar nedir, kaç kez tekrar etmektedir?

Bu problemde istenen şey en çok tekrar eden alt katar.Eğer girilen stringin hiçbir elemanı kendini tekrar etmiyorsa her altkatar 1 kez tekrar edeceğinden en çok tekrar eden altkatar tüm altkatarlardır buna stringin kendisi de dahildir.Bu problem için uygulanan şey tekrardan stringin soneklerinin öneklerini bir matrise atılması oldu. Kontrol işlemleri tekrar eden en uzun altkatar problemindeki gibi yapıldı. Eşitlik durumunda sayac arttırıldı ve dış döngüde bulunan if bloğu ile bu sefer sadece sayacın önceki max değerinden büyük olduğu durumlarda max tekrar sayısı ve altkatar değiştirildi.döngü bittikten sonra kelime yazdırıldı.

"abracadabra" stringi için en çok tekrar eden altkatar:



IV. Karşılaşılan Sorunlar ve Çözümleri

Overlapping Problemi(Katar Arama-Tekrar Eden En Uzun Altkatar:

Hataya sebep olan durumlar ard arda aynı harfin çok tekrar ettiği stringlerde (örnek olarak "qzaaaaab" gibi) tekrar eden en uzun altkatar tekrar eden harfin harf sayısının bir eksiği olarak tekrar etme durumu iki kez olacak şekilde bulunuyordu. "qzaaaaab" için beklenen çıktı 2 kez "aa" altkatarının tekrar etmesi olurken "aaaa" altkatarı 2 kez tekrar etmesi şeklinde gösteriliyordu. Bunun sebebi tüm işlemlerin soneklerin öneklerinde yapıldığından dolayı aynı yerin üzerinden iki kez geçme (overlapping) denilen durum ortaya çıkıyordu. Bunu düzeltmek için soneklerin önek kontrolünden sonra string in kendisinde de kontrol amaçlı bir if bloğu oluşturuldu sorun böylelikle hallolmuş oldu.

V. Sözde Kod

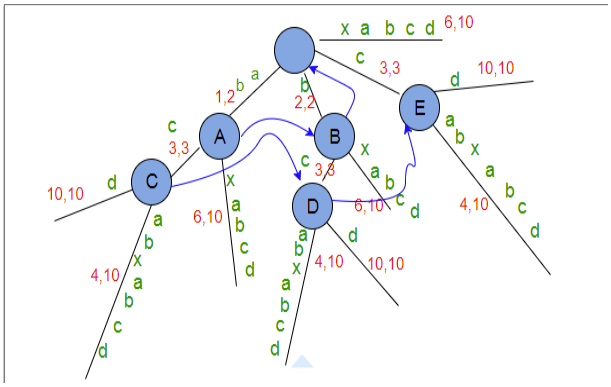
```
int main


char sonek[100][100];
char onek[100][100];
char kelime[100],katar[100];
dosyadan_al(kelime);
yaz(kelime)
char yedek[100];
strcpy(yedek,kelime);

for(int i=0; i<boyut(kelime); i++)
 kopyala(sonek[i],kelime+i);
 kopyala(onek[i],yedek);
 yedek[boyut(yedek)-1] = '\0';
end for

yaz(1- Ağaç oluşturabilir mi?)
yaz(2- Aranan katar kelimede geçiyor mu?)
yaz(3- Tekrar eden en uzun altkatar)
yaz(4- En çok tekrar eden altkatar)
yaz(5- Çıkış)

yaz(bir seçim yapınız)
oku(seçim)

_seçim 1 ise;

int agac_olusur_mu(int length,
char sonek[length][100],char onek[lengh][100])
 for(int i=1;i<length;i++)
  if(sonek i == onek i)
 yaz(agac olusamaz)
 return 0;

end if
 end for

 yaz(agac olusabilir)
 return 1;

_seçim 2 ise;

void katar_geciyor_mu(int length,
char sonek[length][100],char katar[100])


 char yedek[100];
 kopyala(yedek,sonek[0];
 int sayac=0,tut=0,q;

 for(int i=0;i<boyut(yedek);i++)
if(katar[0]!=yedek[q]
 q=i;
 for(int k=0;k<boyut(katar);k++,q++)
  if(katar[k]!=yedek[q]
  break;
  else if(k==(boyut(katar)-1))
    sayac++;
  end for
  i=q-1;
end if
 end for

 char *ptr;
```

```
 ptr=strstr(sonek[0],katar);
 if(sayac>0)
  yaz(katar vardır ve
  sayac kez geçmektedir);
 end if
 else
  yaz(aranan katar bulunamadı);
 end else


_seçim 3 ise;

void tekrar_eden_en_uzun_altkatar
(int length,char sonek[length][100])


 char yedek[100];
 kopyala(yedek,sonek[0];
 int j=0,sayac=0,tut=0,max_tekrar=0;
 int max_uzunluk=0;
 char sonek_onek[5050][100];

 while(boyut(yedek)>0)
  for(int i=tut;i<5050; i++)
   if(boyut(yedek)>0)
kopyala(sonek_onek[i],yedek);
yedek[boyut(yedek)-1]='\0';
   end if
   else
tut=i;
break;
   end else
  end for
  j++;
  kopyala(yedek,sonek[j]);
 end while

 j=0;
 kopyala(yedek,sonek[0]);
 int q;

 while(boyut(sonek_onek[j])>0)
  for(int i=0;i<boyut(yedek);i++)
if(sonek_onek[j][0]!=yedek[q]
 q=i;
 for(int k=0;k<boyut(sonek_onek[j]);k++,q++)
  if(sonek_onek[j][k]!=yedek[q]
  break;
  else if(k==(boyut(sonek_onek[j])-1))
    sayac++;
  end for
  i=q-1;
end if
   end for
  if(boyut(sonek_onek[j])>max_uzunluk &&
 sayac>1)
kopyala(temp,sonek_onek[j]);
max_tekrar=sayac;
max_uzunluk=boyut(sonek_onek[j]);
  end if
  j++,sayac=0;
 end while

 if(max_tekrar>1)
  yazdır(tekrar eden en uzun altkatar temp
tekrar sayısı max_tekrar);
 end if
 else
  yazdır(tekrar eden altkatar bulunamadı);

_seçim 4 ise;

void en_cok_tekrar_eden_altkatar
(int length,char sonek[length][100])
 char yedek[100];
 kopyala(yedek,sonek[0];
 int j=0,sayac=0,tut=0,max_tekrar=0;
 int max_uzunluk=0;
 char sonek_onek[5050][100];

 while(boyut(yedek)>0)
  for(int i=tut;i<5050; i++)
   if(boyut(yedek)>0)
kopyala(sonek_onek[i],yedek);
yedek[boyut(yedek)-1]='\0';
   end if
   else
tut=i;
break;
   end else
  end for
  j++;
  kopyala(yedek,sonek[j]);
 end while
 j=0;

 while(boyut(sonek_onek[j])>0)
  for(int i=0;i<tut; i++)
if(karşılaştır(sonek_onek[j],sonek[onek]))
  sayac++;
 end if
  end for
  if(sayac>max)
kopyala(temp,sonek_onek[j]);
max=sayac;
  end if
 end while

 yazdır(en cok tekrar eden temp sayısı max)

_seçim 5 ise;
 Çıkış: Programdan çıkış yap;
```

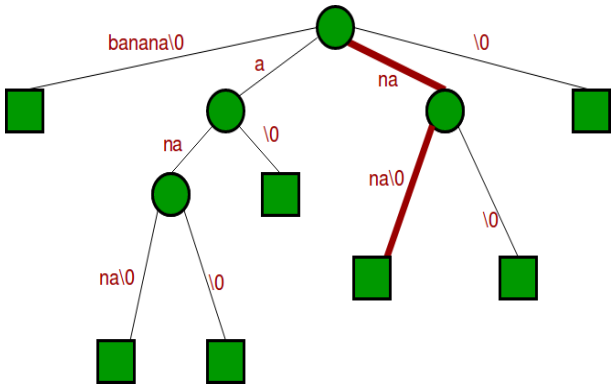
VI. Sonuç

Sonek ağaç yapısı gerçek hayat uygulaması olarak en çok Biyoinformatik ve Hesaplamalı Biyoloji gibi alanlarda kullanılır. Örnek olarak çok fonksiyonlu bazı proteinlerin uzunlukları binlerce karakter olabilirken bir protein dizilimleri 20 farklı karakterden oluşur. Böyle çok uzun katarlar içinde çok kısa birçok katarı aramak için temel yöntemler kullanılması pahalıdır. Ancak soneklere dayalı bazı veri yapıları kullanılarak arama işlemi çok daha ucuza (algoritmik karmaşıklık olarak) yapılabilir. Bu amaçla sonek ağaçları ve sonek dizileri geliştirilmiştir.

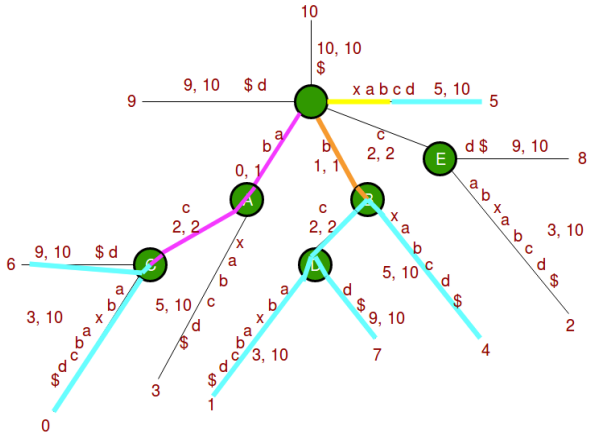
Ağaç yapısı oluşturmanın farklı yolları vardır bunun en bilineni Ukkonen'in algoritması kullanılarak koluşturulandır.Algoritma özetle katarın içindeki her bir altkatarın ilk ve son harfinin indislerinin sayısal olarak alınması ve tutulması şeklindedir. Örnek olarak "xabxac" katarı için "xac" altkatarının değeri [3,5] şeklinde ifade edilir.Ağaç yapısının bir örneği şu şekildedir:



Ağaç yapısında altkatar araması ise sayısal değerler ve bağlı liste kullanımı ile gerçekleşir önce köke bağlı olan dallardan başlangıç altkatarı sağlayan dal bulunur ve oradan aynı arama şekli ile altkatarın tamamı bulunana kadar devam edilir."Banana" kelimesinin sonek ağacında "nana" için yapılan arama şu şekildedir:



Bu örnekte ise "abcabxabcd" stringi için "abc","xab" ve "c" için yapılan aramalar gösterilmiştir."abc" için bulunan sonuç mor ile,"xab" için olan sonuç sarı ile "c"için olan da turuncu ile gösterilmiştir. Mavi renkler bulunduğu dalın indis değerini göstermek için yapılmıştır. Örnek olarak "xab" için sarı ile gösterilen çizgi altkatarın bulunduğu yerdir devamında mavi ile devam edip 5 numaralı yere çıkması da "abcabxabcd" içinde "xac" altkatarının başlangıç değeri olan "x"'in katar içinde ilk kez 5.indiste gözükmesidir.



VII. Kaynakça

Sonek, Önek, Katar, Altkatar kavramları için;
- www.koseburak.net/blog/
- en.wikipedia.org/wiki/

Ağaç yapısı oluşturma ve görselleştirme için; - brenden.github.io/ukkonen-animation/
- newbedev.com

Ukkonen Algoritması ve Uygulamaları için;
- geeksforgeeks.org
- codeforces.com

LaTeX Raporu hazırlamak için gerekli ekipman ve bilgiler;
- www.overleaf.com