Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Programlama Laboratuvarı-1

Mahmut Emre Terzi-180201087
Mahmut Bilgi-200201037

Özet

Bu doküman Programlama Laboratuvarı 1 dersi 2. Projesi için çözümümü açıklamaya yönelik oluşturulmuştur. Dokümanda projenin tanımı, çözüme yönelik yapılan araştırmalar, kullanılan yöntemler, proje hazırlanırken kullanılan geliştirme ortamı ve kod bilgisi gibi programın oluşumunu açıklayan başlıklara yer verilmiştir. Doküman sonunda projemizi hazırlarken kullandığımız kaynaklar ve proje derlenirken dikkat edilmesi gereken hususlar bulunmaktadır.

1-Proje Tanıtımı

Bu projenin amacı sonek ağaçlarını ve sonek dizililerini kullanarak katarlar üzerinde bazı arama işlemleri yapmaktır.
Bir katar için sonek, katarın herhangi bir karakterinden başlayarak sonuna kadar olan kısımdır. Örnek olarak bilişim kelimesinin tüm sonekleri aşağıdaki gibidir.
1. bilişim(birinci karakterden başlayan sonek) 2. ilişim (ikinci karakterden başlayan sonek)
3. lişim (üçüncü karakterden başlayan sonek)
4. işim (dördüncü karakterden başlayan sonek) 5. şim (beşinci karakterden başlayan sonek)
6. im (altıncı karakterden başlayan sonek)
7. m(yedinci karakterden başlayan sonek)

Bir kelimenin öneki ise kelimenin ilk karakterinden başlayarak herhangi bir karakterine kadar olan kısımdır. Örnek olarak bilişim kelimesinin tüm önekleri aşağıdaki gibidir.
1. b (birinci karakterde sonlanan önek)
2. bi (ikinci karakterde sonlanan önek)
3. bil (üçüncü karakterde sonlanan önek)
4. bili (dördüncü karakterde sonlanan önek)
5. biliş(beşinci karakterde sonlanan önek)
6. bilişi(altıncı karakterde sonlanan önek)
7. bilişim(yedinci karakterde sonlanan önek)

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 katarın sonek ağacı aşağıdaki özelliklere sahiptir:

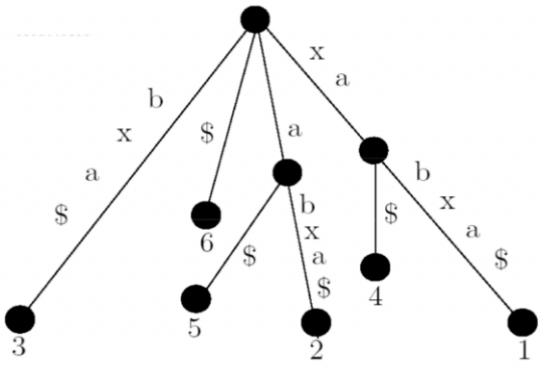
• Ağacın 1'den n'e kadar numaralandırılmış n adet yaprağı vardır.

• 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. Örnek olarak xabxa$ katarının sonek ağacı Figür 1'de verilmiştir

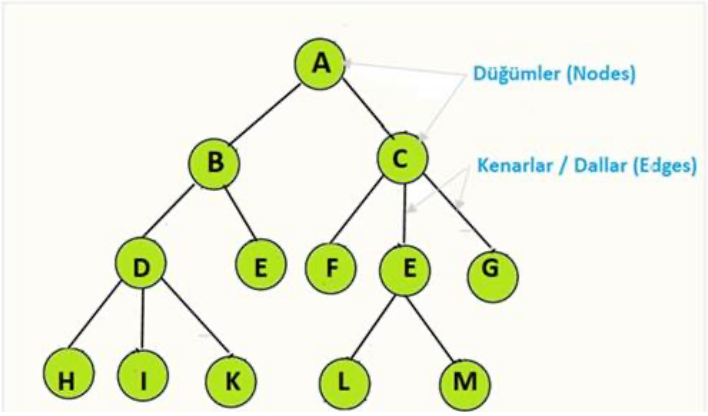


Figür 1

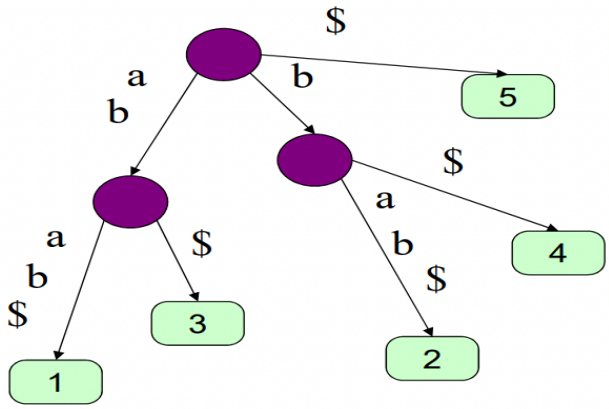
2-Proje yöntemi Deney ve Araştırma

Bu proje kapsamında sonek ağaçları kullanılarak aşağıdaki problemler çözülmesi istenmektedir:

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? Yukarıdaki isterler grafiksel olarak gösterilmek istenmektedir.Öncelikli olarak projede veri yapıları ve ağaç yapısını öğrenmemiz gerekliydi.Ağaç yapısı kısaca şöyle anlatılmaktadır. Ağaç (Tree) veri yapısı çok yaygın olarak kullanılan çok güçlü bir veri yapısıdır. Ağaçlar (Trees) doğrusal (lineer) veri yapıları olan diziler, bağlantılı listeler, yığınlar ve kuyruklardan farklı olarak, doğrusal olmayan hiyerarşik bir veri yapısıdır. Gündelik hayattan bildiğimiz soy ağacı ağaç veri yapısı hakkında bize fikir verebilir. Ağaç, verilerin birbirine sanki bir ağaç yapısı oluşturuyormuş gibi sanal olarak bağlanmasıyla elde edilir. Bu yapıda veri, düğümlerde (node) tutulur. Düğümlere ağcın elemanı denir. Ağaç veri yapısında düğümler arası ilişki kenarlar /dallar (edges) kullanılarak oluşturulur. Başka bir ifade ile ağaç veri yapısında düğümler birbirine kenarlar/dallar kullanılarak bağlanır. Aşağıda ağaç yapısı gösterilmiştir.



Ağaç yapısından sonra projedeki asıl ister olan suffix tree modellemesini öğrenmemiz gerekti.
Suffix Tree'nin Tanımı:
m uzunluğundaki bir S string için T suffix tree aşağıdaki özelliklere sahiptir:
• Köklü bir ağaçtır ve yönlüdür
•1 ile m arasında etiketlenmiş m yaprağı vardır
• Ağaçtaki her bir dal S string nin bir alt stringini oluşturur
• Kökten, i. yaprağa kadar etiketlenmiş bir yol üzerindeki kenarlar birleştirilebilir
• Kök olmayan her ara düğümün en az 2 yaprağı vardır
• Bir düğümden çıkan kenarlar farklı karakterler ile başlar.
S=abab
S string'inin suffix tree'si, S'nin bütün suffix'lerini sıkıştırılmış bir trie de tutsun. $ sembolü ilgili suffix'in sonunu göstersin.
{ $, b$, ab$, bab$, abab$ }



Suffix ağacımız yukarıdaki şekilde oluşturulmuştur.Suffix ağaçlarının işe yaradığı alanları şöyle anlatabiliriz.
Suffix Tree (Sonek Ağacı) kelime işleme algoritmalarındandır.
DNA dosyaları gigabyte seviyesinde yer kapladıklarından DNA analizinin elle yapılması mümkün değildir. Hatta, DNA dosyalarının bilgisayar yardımıyla işlenmesi de çok uzun sürmektedir.
Biyolojik veriler, arama motorları, derleyici tasarımı, işletim sistemi, veri tabanı, vs.. kullanılır.

3-Yalancı Kod

include
Struct n oluştur;
Struct n altında Struct n*çocuk[n] oluştur;
Struct n altında int basla,int son,int index oluştur;
Char katar[N]boyutlu dizisini tanımla;
İnt dugumMu fonksiyonunu tanımla;
İnt dugumMu fonksiyonuna x,y,z parametrelerini ata;
İnt dugumMu fonksiyonunda x==z ye ise return x-1 döndür;
İnt dugumMu fonskiyonunda katar[x]==katar[y] ise Return dugumMu fonksiyonun içinden x=x+1,y=+1,z=z döndür;
İnt dugumMu fonskiyonunda istenenler sağlanmıyor(else) ise return x-1 döndür;
void dalYap fonksiyonunu tanımla;
İnt dalYap fonksiyonunda parametre olarak *n kullan;
void dalYap fonksiyonunda katar karakter dizisi boyutuna eş int M tanımla;
void dalYap fonksiyonunda M kadar döngü tanımla;
void dalYap fonksiyonunda çocuk,son ve indexi kullanarak işlemleri yap;
void icDugum tanımla,parametre olarak node*root,int i kullan;
void icDugum içinde int t=0,int m=katar karakter dizisi boyutu,int bitiş [10],int başlangıç[10],int ilk[10] tanımla;
void icDugum içinde 3 farklı döngü oluştur;
void icDugum içinde ikisi j ve k ya kadar olucak şekilde biri ise eşit değildir i ye ve k M den küçükse şeklinde döngüler oluşturarak icDugum sağlanmış olucaktır;
void icDugum2 tanımla,parametre olarak node*root kullan;
void icDugum2 içinde int M=katar karakter dizisi boyutu,int bitiş[10],int başlangıç[10],int ilk[10] tanımla;
void icDugum2 içinde 3 farklı döngü oluştur;
void icDugum2 içinde biri M ye kadar biri t ye kadar diğeri k i ye eşit olmayacak ve k M den küçük olana kadar olucak şeklinde döngüler oluşturularak icDugum2 sağlanmış olucaktır;
void Dugumici tanımla,parametre olarak node*root kullan;
void Dugumiçi içinde katar karakter dizisi boyutunda int M tanımla;
void Dugumiçi içinde dalyap fonkisyonu içine rootu gönder ;
void Dugumiçi içinde M ye kadar olucak döngü şeklinde Dugumiçini oluştur;

void Dugumici2 tanımla,parametre olarak node*root, node*n tanımla;
void Dugumici2 içinde katar karakter dizisi boyutunda int M değişkeni tanımla;
void Dugumici2 içinde b tanımla;
void Dugumici2 içinde dalyap fonksiyonu içinde rootu gönder;
void Dugumici2 içinde M ye kadar olucak şekilde Dugümiçi2 yi oluştur;
void grafik fonksiyonunu tanımla,parametre olarak node *n kullan;
void grafik fonkisyonu içinde ağaç gösterimi katar gösterimi graphic h kütüphanesindeki işlevleri kullanarak gösterme ve çizdirme döngülerini ayarlamalarını yap;
void sorgula fonskiyonunu tanımla,parametre kullanma;
void sorgula fonksiyonunda N boyutunda string oluştur,system(cls) komutunu tanımla;
void sorgula fonksiyonunda string girişi ve bu string girişini int A ya atama yap;
void sorgula fonksiyonunda A için onek ve sonek char formatında tanımlamalar yap;
void sorgula fonksiyonunda sonek onek oluşumlarına bakarak katar için sonek ağacı oluşturulabilir değilse oluşturulumaz çıktılarını oluştur;
void olustur fonskiyonunu tanımla ve parametre kullanma;
void olustur fonkisyonunun içinde str [N] boyutunda char tanımla,string girişi için tanımlama yap;
void olustur içinde dalyap fonksiyonuna root gönder;
void oluştur içinde çocuk oluştur döngü ile;
void oluştur içinde icDugumu kullan ve içine root ve i gönder;
void oluştur içinde icDugumuden sonra dugumici2 ve icdugum2 fonskiyonlarını döngü içinde M ye kadar olucak şekilde oluştur;
void oluştur içinde tekrar eden katarı,en uzun alt katarı ve tekrar eden alt katarı bulmak için, ;
void oluştur içinde int tut,durum,basla,son,durum,tekrar=0 tanımlamalarını yap;
void oluştur içinde katar arama işlevlerini yap;
void oluştur içinde suffix tree bulunma bulunmama durumlarını göster;
main fonksiyonu tanımla,while döngüsü ile menü oluştur;
main içinde 2 seçim tanımla,il seçim sorgula fonksiyonu,ikinci seçim oluştur fonskiyonunu çağır,remove ile katar texti yaz ve kapat;
return0;

4-Kaynakça

https://www.koseburak.net/blog/suffix-tree/

https://www.it-swarm-tr.com/tr/suffix-tree/

https://cdn-acikogretim.istanbul.edu.tr/auzefcontent/20_21_Guz/veri_yapilari/9/index.html

https://avesis.yildiz.edu.tr/resume/downloadfile/diri?key=7aa1aefa-a342-4dca-ba27-27e38b5359ec

https://qastack.info.tr/programming/9452701/ukkonens-suffix-tree-algorithm-in-plain-english

https://nerdbook.wordpress.com/2018/03/28/agac-veri-yapisi/

https://cdn-acikogretim.istanbul.edu.tr/auzefcontent/20_21_Guz/veri_yapilari/9/index.html

https://www.youtube.com/watch?v=VEkAj-xVTKQ&t=187s