SIKIŞTIRMA UYGULAMALARI

Şevki Karagöl
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
1700201009

Mustafa Yiğit
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
180201108

ÖZET- Bu projede, projeyi yapan kişiler için sıkıştırma mantığını ve sıkıştırma algoritmalarını anlamak ,bu mantık ve sıkıştırma algoritmalarını kullanarak bir probleme çözüm sağlayabilmesi amaçlanmaktadır. Bu amaç doğrultusunda öğrenciden, kullanıcıdan alınan bir kaynak metni,"Lz77" ve "Deflate" algoritmalarıyla sıkıştıran ve boyutunu küçülten bir program tasarlaması,ardından performans karşılaştırması yapması beklenmektedir.

Bu amaçlar ve isterler doğrultusunda bir dosya sıkıştırma uygulaması tasarlanmıştır.Proje için gerekli dosya okunduktan sonra veri modeline yerleştirilmiş ve proje isterleri yerine getirilmiştir.Deflate algoritmasının çalışma mantığı tam olarak anlaşılamadığı için Deflate algoritmasında kullanılan, Lzss algoritması ve Huffman kodlaması ayrı ayrı tasarlanmış ve performans karşılaştırmasına dahil edilmiştir.

Anahtar kelimeler-

proje,boyut,sıkıştırma,algoritma ,veri ve program

I.GİRİŞ

Proje klasörün içinde bulunan "source.txt" dosyasındaki kaynak metin okunarak bir pointer dizisine yerleştirilir. Proje kapsamında

öğrenciden istenen işlemlerin gerçekleştirilmesinin ardından metnin asıl boyutu ve sıkıştırılmış boyutu hakkında kullanıcıya bilgi verilir.Bu işlemden sonra proje sonlandırılır.

Bu projede veri yapıları,dosya işlemleri ve pointerların bir arada kullanımına yönelik bir çalışma gerçekleştirilmiştir.Aynı zamanda öğrencilerin, proje isterlerlerinin çözümününe yönelik oluşturduğu algoritmaları ide aracılığıyla bilgisayar ortamına aktarması amaçlanmıştır.Proje için tercih edilen programlama dili "C" olmuştur.

II.TEMEL BİLGİLER

Bu proje C programlama dilinde geliştirilmiş olup, geliştirme ortamı olarak "Code Blocs ver. 13.12" kullanılmıştır.İlk etapta proje tanıtım belgesi, bir grup konuşmasında, ortak bir şekilde okunup gerekli notlar alınmıştır.Ardından projede karşılaşılabilecek sorunlar hakkında tartışılmıştır.Bu tartışmaların ışığında bir yol haritası çıkarılmış, ön hazırlık sürecine girilmiştir. Bu süreçte araştırma konuları grup üyeleri arasında paylaştırılarak projenin isterlerine ve kullanılacak algoritmalara yönelik araştırmalar yapılmıştır.Elde edilen veriler doğrultusunda projenin ana hatları ortaya çıkarılmıştır.Daha önce sıkıştırma algoritmaları üzerine herhangi bir çalışma yapmamız ve bilgi sahibi olmamamız, projeyi uygulamamızdaki ilk zor adım olarak göze çarpmaktadır.Gerekli çalışmalar yapıldıktan sonra bu problem çözüme kavuşturulmuştur ve derleyici ortamında projenin ilk adımları atılmıştır.Proje yaklaşık 15 gün gibi süre içerisinde tamamlanmıştır.

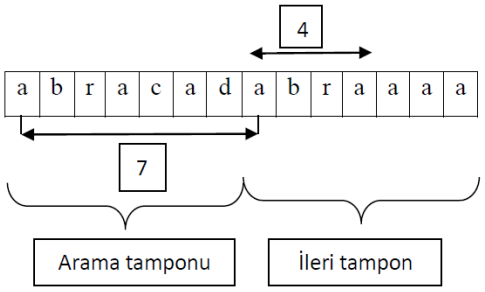
III.YÖNTEM

Bu projede “Lz77”, “Lzss” ve “Huffman Kodlaması ” algoritmaları kullanılmıştır.Lz77 algoritması için , “lz77.txt ” metin belgesi, Lzss algoritması için , “lzss.txt” metin belgesi oluşturulmaktadır.Huffman Kodlaması için ise sadece konsol üzerinde kullanıcıya bilgi verilmekte ve herhangi bir metin belgesi çıktısı oluşturulmamaktadır.Bu başlık altında,projede kullanılan algoritmaların yöntemleri detaylı bir şekilde anlatılmıştır.

1)LZ77 Algoritması

Öncelikle ,içerisinde metin üzerinde ne kadar geri geldiğimizi,tekrar sayısının kaç olduğunu ve harf bilgisini tutan bir struct yapısı tanımladık.Bu yapı metni sıkıştırmamızı sağlayan temel yapıdır.Ardından proje dosyasında bulunan “kaynak metin” dosyamızı okumamızı sağlayan fonksiyonumuzu tasarladık.Bu fonksiyon,metin belgesinde kullanılan imleci baştan sona gezdirmekte ve metnin boyutunu tespit etmektedir.Bu tespitten sonra bu boyutta bir tampon belleğe ,metin tamamen aktarılmaktadır.Bu tampon bellek char dizisi şeklindedir.

Bu işlemden sonra metnin sıkıştırılmasını sağlayan tokenleri doldurma işlemine geçilmektedir ve kaynak metin,sıkıştırma işlemini yapan fonksiyona gönderilmektedir.Bu fonksiyonda arama tamponu ve ileri tampon tanımlanmaktadır.



Bu tamponlar yukarıdaki görselde gösterilmiştir.

Bu tamponların tanımlanma amacı metinde tekrar eden karakteri veya karakter gruplarını bulmak ve tokenlerin içerisine doldurmaktır.Görselini eklediğimiz “abracadabra” örneği üzerinden algoritmayı incelemeye devam edelim. İleri tamponun ilk karakteri olan a, arama tamponunda baştan sona doğru aranır. Bu arama işleminin gerçekleştirilmesi için arama tamponun başlangıç noktası,ileri tamponun 255 karakter gerisine alınır.Burada arama tamponunu 255 karakter geriye almamızın nedeni 8 bitlik bir değişken kullanmamızdır.Geriye alma işleminin ardından, arama tamponumuzun başlangıç noktası metnin dışına çıkarsa,arama tamponunun başlangıç noktasını 0’a eşitleriz ve metnin içinden başlamasını sağlarız.Benzerlik arama işlemi bu adımdan sonra başlar.İleri tamponumuz ve arama tamponumuzun ilk elemanlarının indislerini,benzerlik bulma fonksiyonumuza göndeririz.İlk göndermemizde benzerliği buluruz ve 4 karakterlik bir grubun tekrar ettiğini bulmuş oluruz.Bulduğumuz bu değeri ve benzerliğin başladığı bu değeri birer değişkende saklarız.Böylelikle benzerliği bulmak için ne kadar geri gittiğimizi hesaplayabiliriz.İkinci karşılaştırmada da benzerlik buluruz, fakat bulunan benzerlik ilk bulduğumuz benzerlikten küçük olduğu için maksimum benzerlik değerini değiştirmez. Arama tamponunun ilk elemanı ,ileri tamponun ilk elemanına eşit olana kadar benzerlik arama işlemi devam ettirilir.Arama tamponunun başında, yani ileri tamponda aranan karakterden 7 karakter uzaklıkta bulduğumuz benzerlik,en uzun benzerliğimiz olduğu için maksimum benzerlik uzunluğu 4 olmuştur.

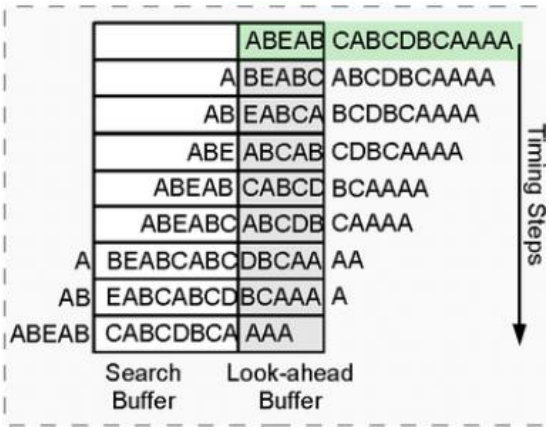
Bulduğumuz bu iki değerle tokenimizin ilk iki elemanını bulmuş olduk.Son elemanımız ise bulduğumuz benzerlikten sonra gelen ilk karakter olacaktır.Yani “abra” dan sonra yer alan a karakteri.Bulduğumuz üçüncü değerden sonra tokenimiz [7,4,a] şeklinde üçlü olarak kodlanır. Bir başka durum ise hiç eşleşme bulunamaması durumudur.Bu durumda maksimum benzerliğimiz 0 olarak kalır ve tokenimiz

[0,0,karakter] şeklinde kodlanır.Burada bahsedilen “karakter”, ileri tamponda ilk karakter olan, eşleşme bulamadığımız karakterdir. Örneğin: ileri tamponumuzun ilk karakteri “a” olsun ve hiç eşleşme bulunamasın.O zaman kodlamamız şu şekilde olur: [0,0,karakter].

İleri tamponumuzun sonuna geldiğimizde,algoritmamızın benzerlik bulma adımı bitmiş olur. Hazır hale gelen tokenlerimiz metin belgesine bastırılmaya gönderilir ve Lz77 algoritmamız sonlanır.

*Tasarlanan programın ikinci aşamasında Deflate algoritması yazılması istenmiştir.Fakat Deflate algoritması günlerce süren araştırmalara rağmen anlaşılamamıştır,Bu problem projenin yapım aşamasında karşılaşılan ikinci ve en zor problemdir.Bu problemi çözememiz nedeniyle projenin ikinci aşamasında Deflate algoritmasının bileşenlerinden olan Lzss algoritması ve Huffman Kodlaması ayrı ayrı kodlanmıştır.

1)LZSS Algoritması



Bu algoritma, Lz77 algoritmasına bir kaç ekleme yaparak oluşturulmuştur.Bu nedenle sadece yapılan eklemelerden ve farklı olan kısımlardan bahsedilecektir.

Lz77 algoritmasında,maksimum eşleşmeyi bulduğumuz fonksiyona,kodlama etkinliği açısından bulunan tekrar sayısının 3 ve üzeri olması için bir “if” bloğu eklenmiştir.Eşleşmenin üçten az bulunması durumunda,hiç eşleşme bulunmamış gibi devam edilir.

Lzss algoritmasını,Lz77 algoritmasından ayıran bir diğer özelliği ise tokenlerinin yapısıdır. Bu farkı yaratan özellik,tokenlerin üçlü değil de ikili kodlanmasıdır.Eşleşme bulunup bulunmamasına göre iki farklı şekilde token oluşturulur:

-<u>eşleşme varsa:</u>

<mesafe,tekrar uzunluğu> şeklinde,

-<u>eşleşme yoksa</u>

<0,ascii kodu> şeklinde oluşturulur.

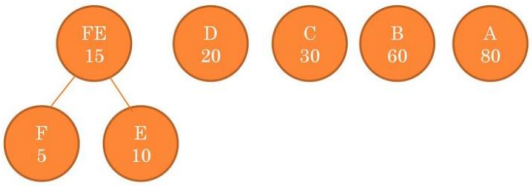
Buradaki “ascii kodu” , ileri tamponda eşleşme aranan karakterin ascii tablosundaki karşılığıdır.

3)Huffman Kodlaması

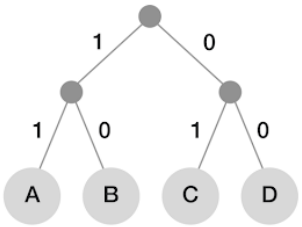
Bu algoritmada “source.txt” ‘den alınan metni en az bit kullanarak kodlamak amaçlanmıştır.

Öncelikle bu algoritmanın temelleri olan düğüm ve ağaç yapıları,structlar ile tanımlanmıştır ve kaynak dosyayı okuma işlemi gerçekleştirilmiştir.Ardından, okunan dosya baştan sona kadar gezilerek metin belgesinde bulunan karakterlerin tekrar etme sıklıkları(frekansları) bulunmuştur.Frekansları sıfırdan farklı olan karakterler teker teker dizi yapısına eklenmiştir.Frekans bulunma işleminde Ascii kodlarından faydalanılmıştır.Düğümleri diziye ek-

leme aşamasından sonra her karakter frekanslarına göre küçükten büyüğe doğru sıralanmıştır ve tüm karakterler bir ağacın yaprak düğümlerine eklenmiştir.Sıradan en az frekansa sahip iki düğüm alınmıştır ve yeni bir düğüm oluşturulmuştur.Bu yeni düğüme,sıradan alınan iki düğümün toplamı atanmıştır. Oluşturulan bu düğüm,sıralamada uygun yere yerleştirilmiştir.



Sırada karakter kalmayıncaya kadar bu işlem devam etmiştir ve ağaç oluşturululmuştur.Ağacın son hali oluştuğunda her bir sembolün yeni kodunu oluşturmaya başlarız.Kodlar oluşturulurken en tepedeki kök düğümden başlanmıştır. Kök düğümün sağ ve sol düğümlerine giden dallara "1" ve "0 " kodları verilir.



*(Görsel temsilidir.Bu projede tasarlanan algoritmada sağ dallara "1", sol dallara "0 " kodları verilmiştir.)

Örneğin : "D" düğümünün kodu 00 şeklinde bulunmuştur ve 2 bittir.Ayrıca bu kodların bulunması esnasında,karakterlerin metinde geçme sıklığı ve kod uzunlukları çarpılarak ,metnin yeni bulunan kodlarla yazılması durumunda, boyutunun ne kadar olacağı bilgisi bulunmuştur.Bu bilgi programın sonunda kullanıcıya bildirilmiştir.

IV.SÖZDE(PSEUDO) KOD

Verilen proje için hazırladığımız programın pseudo kodları şu şekildedir:

1-Program çalıştı.

2-"source.txt" dosyası okundu.

3-Huffman ağacı oluşturuldu ve kodlar bulundu.

5-Yeni bulunan kodlar ile metnin yeni boyutu hesaplandı.

6-Kullanıcıya Huffman kodlamasının sonucu ile ilgili bilgi verildi. *ek1

5-"source.txt" dosyası okundu.

6-Lzss sıkıştırma algoritması çalıştırıldı.

7-Lzss sıkıştırma algoritması sonucu oluşan tokenler bir dizide depolandı.

8-Depolanan tokenler bir metin belgesine bastırıldı. *ek2

9-Kullanıcıya Lzss sıkıştırma algoritmasının sonucuyla ilgili bilgi verildi. *ek3

10-"source.txt" dosyası okundu.

11-Lz77 sıkıştırma algoritması çalıştırıldı.

12-Lz77 sıkıştırma algoritması sonucu oluşan tokenler bir dizide depolandı.

13-Depolanan tokenler bir metin belgesine bastırıldı. *ek2

14-Kullanıcıya Lzs77 sıkıştırma algoritmasının sonucuyla ilgili bilgi verildi. *ek4

15-Program sonlandırıldı.

**Programın son halinde konsol ekranında sadece dosya boyutları hakkında bilgi verilmiştir.Token içeriklerinin nasıl olduğunun gösterilmesi amacıyla, rapora konsol ekranında tokenlerin bastırıldığı ekran görüntüleri eklenmiştir. *ek5*

V.PERFORMANS KARŞILAŞTIRMASI

Algoritma adı	Kaynak dosya boyutu	Sıkıştırılmış dosya boyutu
LZ77	3458 bayt	2997 bayt
LZSS	3458 bayt	3428 bayt
HUFFMAN KODLAMASI	27664 bit	14744 bit

**Huffman kodlamasının çıktı dosyası oluşturulmadığı için,sıkıştırma oranı bit hesabı ile yapılmıştır.Bütün algoritmalarda aynı metin kullanılmıştır.*

Sıkıştırma yüzdeleri:

Lz77- %13.33

Lzss- %0.86

Huffman Kodlaması- %46.7

Tabloyu incelediğimizde Huffman Kodlamasının dosyanın boyutunu %46.7 oranında küçülttüğünü görüyoruz.Bu durumda en başarılı sıkıştırma algoritması Huffman Kodlaması olarak görünüyor.Lz77 ve Lzss algoritmalarının sıkıştırma oranlarına baktığımızda ise Lz77 algoritmasının,Lzss algoritmasına göre daha başarılı olduğunu görüyoruz.Bu durumda bu üç algoritma arasında başarı sıralamasını şu şekilde yapabiliriz:

1)Huffman Kodlaması

2)Lz77 Sıkıştırma Algoritması

3)Lzss Sıkıştırma Algoritması

Lz77 ve Lzss algoritmalarında sıkıştırma oranı düşük görünse de , kaynak dosya olarak kullanılan metnin büyütülmesi durumunda bu algoritmalardan daha yüksek yüzdelerde sıkıştırma oranları alabiliriz.

VI.EKLER VE AKIŞ ŞEMASI

*ek1

```
->z karakteri: Kod: 110100001110 Frekans: 1
->P karakteri: Kod: 110100001111 Frekans: 1
 karakteri: Kod: 11010001 Frekans: 15
->v karakteri: Kod: 1101001 Frekans: 32
->u karakteri: Kod: 110101 Frekans: 69
->l karakteri: Kod: 11011 Frekans: 138
->  karakteri: Kod: 111 Frekans: 576

 *Huffman Agaci:
   ->Txt'nin normal boyutu 27664 bit
   ->Huffman kodlamasi sonrasi boyutu 14744 bit
   ->Metin %46.7 oraninda kuculmustur.
```

*ek2

*ek3

```
 -1710.Dugumumuz:
!!imlecin geri gitme miktari: 0
 !!Dugumde tutulan karakter: 4
----------------------------
 -1711.Dugumumuz:
!!imlecin geri gitme miktari: 0
 !!Dugumde tutulan karakter:
----------------------------
 -1712.Dugumumuz:
!!imlecin geri gitme miktari: 0
 !!Dugumde tutulan karakter: A
----------------------------
 -1713.Dugumumuz:
!!imlecin geri gitme miktari: 0
 !!Dugumde tutulan karakter: d
----------------------------
 -1714.Dugumumuz:
!!imlecin geri gitme miktari: 0
 !!Dugumde tutulan karakter: v
----------------------------
 -->dugum adedi 1714
LZSS sonucu :
Orijinal Boyut: 3458 kb, Kucultulmus Boyutu: 3428 kb
```

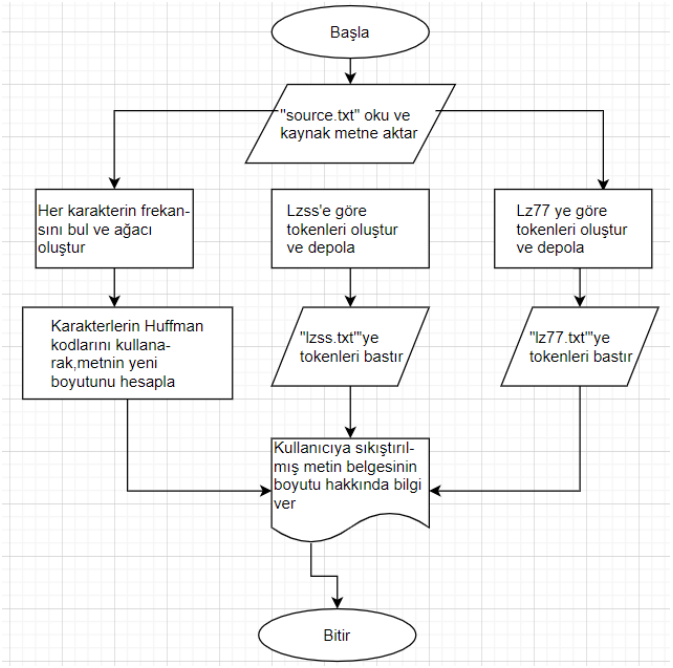
*ek4

```
***** 994.Tokenimiz:**********************
 jump->154 tekrar->2 harf->i
**********************************
***** 995.Tokenimiz:**********************
 jump->240 tekrar->3 harf->2
**********************************
***** 996.Tokenimiz:**********************
 jump->0 tekrar->0 harf->0
**********************************
***** 997.Tokenimiz:**********************
 jump->1 tekrar->1 harf->4
**********************************
***** 998.Tokenimiz:**********************
 jump->145 tekrar->2 harf->d
**********************************
***** 999.Tokenimiz:**********************
 jump->213 tekrar->1 harf->
**********************************
 -->token adedi 999

 LZ77 Sonuclari:
Orijinal Boyut: 3458 kb, Kucultulmus Boyutu: 2997 kb
```

*ek5

```
-----------------------------------------------

 HUFFMAN Inceleme:

 *Huffman Agaci:
   ->Txt'nin normal boyutu 27664 bit
   ->Huffman kodlamasi sonrasi boyutu 14744 bit
   ->Metin %46.7 oraninda kuculmustur.
-----------------------------------------------

 LZSS inceleme
 -->token adedi 1714
LZSS sonucu :
Orijinal Boyut: 3458 kb, Kucultulmus Boyutu: 3428 kb
-----------------------------------------------
 LZ77 Inceleme
 -->token adedi 999

 LZ77 Sonuclari:
Orijinal Boyut: 3458 kb, Kucultulmus Boyutu: 2997 kb
-----------------------------------------------
```

AKIŞ ŞEMASI



VII.KAYNAKÇA

[1]stackoverflow.com

[2] Youtube.com

[3]geekforgeeks.com

[4]embedded.kocaeli.edu.tr

[5]edestek2.kocaeli.edu.tr

[6]Veri Sıkıştırmada Yeni Yöntemler - Altan Mesut -Doktora Tezi

[7] faqs.org

[8] ysar.net

[9] slideplayer.biz.tr

[10] wikipedia.org

[11]zlib.net

[12]slideshare.net