Sıkıştırma Algoritmaları

Engin Yenice
Kocaeli Üniversitesi
Bilgisayar Mühendisliği
190201133
enginyenice2626@gmail.com

***Özet*—Bir metin LZ77 ve Deflate algoritmaları kullanılarak sıkıştırılması ve bu sıkıştırma işlemlerinin sonucunun karşılaştırılması.**

Keywords—lz77, lzss, deflate, sıkıştırma, c, encode, bit, byte

I. GIRIŞ

Bir metin dosyasının Lz77 ve Deflate algoritmaları ile sıkıştırılması ve sıkıştırılan dosyalar ile gerçek dosyaların boyutlarının karşılaştırılması işlemi yapılmaktadır. Bu algoritmaları kodlamak için seçilen yazılım dili C olarak belirlenmiştir.

II. ALGORITMALARI TANIYALIM

A. LZ77

Abraham Lembel ve Jacob Ziv tarafından 1977 yılında yazdıkları "A *Universal* Algorithm for Sequential Data Compression" *başlıklı* makale ile ortaya atılmıştır. Genellikle hazırlanmış metinlerde kendi kendini tekrar edilen kısımlar bulunmaktadır. LZ77 algoritmasının amacı, metin belgesindeki tekrar eden bölümleri ortadan kaldırmak ve buna yönelik metin belgesini sıkıştırmaktır.

LZ77 algoritmasında kullandığımız temel parametreler vardır. Bunlar offset, lenght ve characterdir.

Offset: Kaç karakter geride olduğunu belirtmek için kullanılır.

Lenght: Bulunan benzerliğin ne kadar uzun olduğunu belirtmek için kullanılır.

Character: O anki veya bir ilerisindeki karakterin ne olduğunu göstermek amacı ile kullanılır. Eğer benzerlik (arama tamponunda) hiç yoksa karakterin kendisi atanır. Fakat arama tamponunda bir veya birden çok benzerlik bulunması durumunda benzerlikten sonra gelen ilk karakter atanır.

Bu parametreler genel olarak şu şekilde gösterilmektedir. <Offset, Lenght, C(Character)>

LZ77 algoritması iki tampondan oluşmaktadır.

1) Arama Tamponu

LZ77 sıkıştırma kısmının en önemli tampon bölgesi burasıdır. Arama tamponu araması sondan başa doğru çalışmaktadır. Bu tamponun içerisinde bulunan karakterler ile ileri tampondaki karakterler karşılaştırılır.

2) İleri Tampon

Arama tamponun son karakterinden sonra başlayan ve belirli bir uzunlukta olan bir tampondur. Arama tamponundaki karakterler ile buradaki karakterler taranır ve benzerlik sonuçları bulunarak sıkıştırılır.

3) LZ77 Nasıl Çalışır

Arama tamponundaki karakterler ile ileri tampondaki karakter eşleşmemesi durumunda bu karaktere ilk defa karşılaşıldığı varsayılır ve offset değerine 0 lenght değerine 0 ve character değerine o harf atanır.

Fakat arama tamponundaki karakter ile ileri tamponun ilk karakteri eşleştiğinde benzerlik oranlarına bakılmaya başlar. En büyük benzerlik oranı bulunduktan sonra ileri tamponun ilk karakterinden benzerliğin ilk karakter arasındaki uzaklık bulunarak offset değerine atanır. Benzerli değeri lenght değerine ve benzerlikten bir sonraki harfi de Character parametresine atanır.

Örneğin: **ababcbababaa** kelimesini LZ77 algoritmasına tabi tutalım

Arama (7 Karakter)	İleri (4 Karakter)	Sıkıştırma
	abab~~cbababaa~~	
a	babc~~babababaa~~	<0,0,C(a)>
ab	abcb~~ababaa~~	<0,0,C(b)>
ababc	baba~~baa~~	<2,2,C(c)>
~~ab~~abcbaba	baa	<4,3,C(a)>
~~ababc~~bababaa		<2,2,C(a)>

4) LZ77 Dezavantajları

LZ77 yaklaşımının tıkandığı bir nokta vardır. Eğer periyodik bir dizimiz varsa ve periyodu arama tamponundan büyükse, hiç benzeşme bulunamaz ve her karakter için fazladan gönderdiğimiz veriler nedeniyle sıkıştırma yerine genişletme yapmış oluruz.

B. Deflate Algoritması

Phil Katz tarafından 90'lı yılların ortalarında geliştirilmiş kayıpsız veri sıkıştırma formatıdır. Huffman Kodlaması ve LZSS algoritmasının bir bileşimidir. LZW'nin tersine, Deflate herhangi bir patent koruması altında değildir. Bu nedenle geniş bir alanda kullanılmaktadır. WinZip (Windows), gzip (Unix), jar (Java) dosyaları (jar aslında bir zip dosyasıdır) ve PNG kayıpsız görüntü sıkıştırma standardı Deflate algoritmasını kullanmaktadır *[Deutsch, 1996].*

1) Huffman Kodlaması

Kayıpsız veri sıkıştırma algoritmasıdır. En büyük avantajlarından birisi verileri frekanslarına göre bir kodlama yapması bu sayede sık kullanılan karakterleri daha az, nadir kullanılan karakterleri ise daha fazla yer kaplamasını sağlar.

a) Huffman Kodlaması Nasıl Çalışır

Önce metindeki karakterlerin frekansları hesaplanır. Frekanslar küçükten büyüğe doğru sıralanır ve ağaç oluşturulur.

Huffman kodlaması ile **aaaabbbccdeeeeee** metnini kodlayalım.

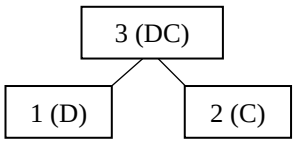
Kodlamamıza başlarken ilk olarak karakter ve frekanslarının bulunduğu tabloyu oluşturalım.

Harf	Frekans
A	4
B	3
C	2
D	1
E	6

Oluşturulan tabloyu küçükten büyüğe sıralayalım.

Harf	Frekans
D	1
C	2
B	3
A	4
E	6

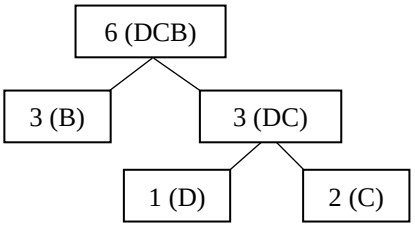
En küçük 2 harfi alalım.



Bu iki harfi bir bütün olarak kabul ettik tablomuzu güncelleyelim.

Harf	Frekans
B	3
DC	3
A	4
E	6

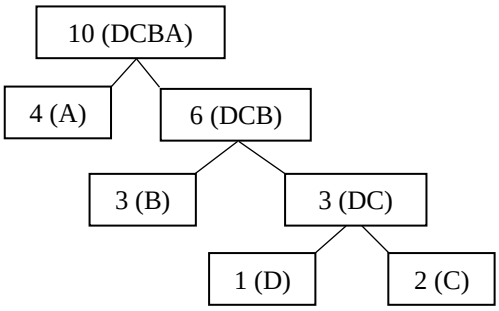
En küçük 2 harfi alalım.



Bu iki parçayı bir bütün olarak kabul ettik tablomuzu güncelleyelim.

Harf	Frekans
A	4
DCB	6
E	6

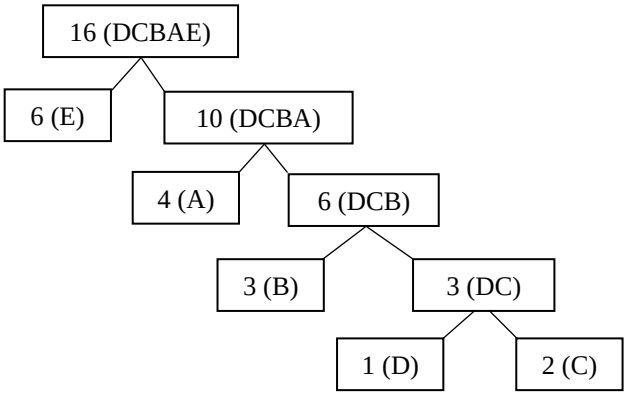
En küçük 2 harfi alalım.



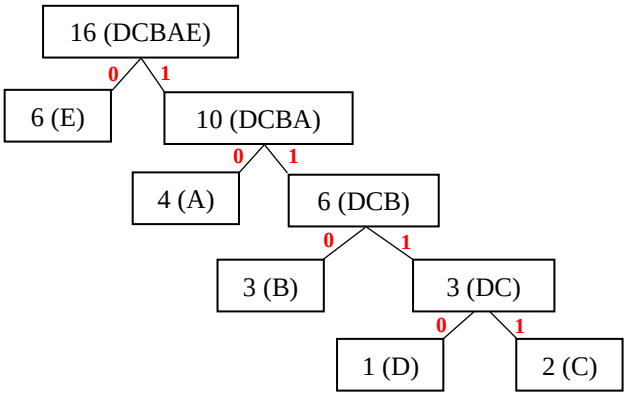
Bu iki parçayı bir bütün olarak kabul ettik tablomuzu güncelleyelim.

Harf	Frekans
E	6
DCBA	10

En küçük 2 harfi alalım.



Ağacımız tamamlandı şimdi sola doğru giden oklara 0 sağa doğu giden oklara 1 değerini atayalım.



Şimdi bulduğumuz değerler ile harflerin kodlamasını yapalım.

Harf	Kodlaması
E	0
A	10
B	110
D	1110
C	1111

Artık metnimizi şifreli olarak yazabiliriz.

10101010110110110111111111110000000

Aaaabbbccdeeeeee

Artık sıkıştırma oranını hesaplayabiliriz.

Normal halinde 16 karakter bulunuyordu. Her harf 3 bite tekabül etmektedir. Yani normal metin 16 * 3 = 48 bit yer kaplamaktadır.

Sıkıştırılmış metin 35 karakterde oluşmaktadır. Her karakter 1 bit olarak kabul edilirse eğer 35 * 1 = 35 bit yer kaplamaktadır.

Sıkıştırma oranını hesaplayacak olursak eğer:

Sıkıştırılmış değerin gerçek değere oranını buluyoruz.

48Bit — 100%

35Bit — ?%

(35 * 100) / 48 = 72,92 %

Tamamından sıkıştırılmış değeri çıkartıyoruz.

100 - 72,92 = 27,08 %

Huffman algoritması ile metnimiz 27,08% sıkıştırılmış olduğunu görebiliriz.

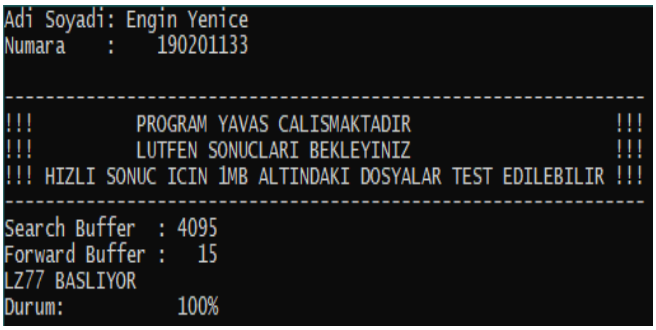
Deflate Sıkıştırma Algoritması kayıpsız veri sıkıştırma formatıdır. Huffman Kodlaması ve

LZ77 algoritmasının bir bileşiminden oluşmaktadır. Her blok, sıkıştırılmış verilerin bulunduğu kısım ve bu verileri temsil eden Huffman kod ağaçlarının saklandığı kısım olmak üzere iki kısımdan oluşur. Kullanılan Huffman ağaçları her bloğun başında sıkıştırılmış veriden önce yer alır ve bu ağaçlar da Huffman kodlaması kullanılarak sıkıştırılır.

III. Program Nasıl Çalışıyor

Programın .exe dosyasının yanına sıkıştırılacak metin dosyası "metin.txt" adı ile kopyalanır ve program çalıştırılır.

A. Ekran Görüntülerile İle Anlatım



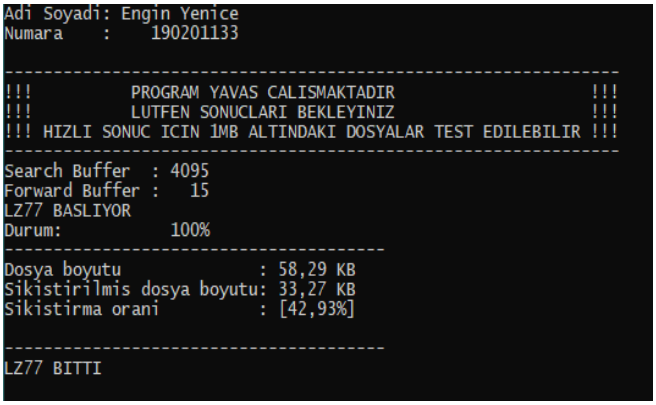
Program çalıştığında ilk olarak LZ77 algoritması ile sıkıştırma işlemini yapıyor.

- Search Buffer: LZ77' den de bildiğimiz arama tamponunu ifade etmektedir. Boyut olarak 4095 karakter olarak ayarlanmıştır.
- Forwad Buffer: LZ77' den de bildiğimiz ileri tamponunu ifade etmektedir. Boyut olarak 15 karakter olarak ayarlanmıştır.

Search buffer ve forward buffer aslında 16 bit (2 byte)lik alanı temsil etmektedir.
Kırmızı ile gösterilen kısım arama tamponunu mor ile gösterilen alan ileri tampon temsil etmektedir.
1111 1111 1111 1111

- Durum: Şuan LZ77 algoritması tamamlanması için yüzde kaç kaldığını göstermektedir.

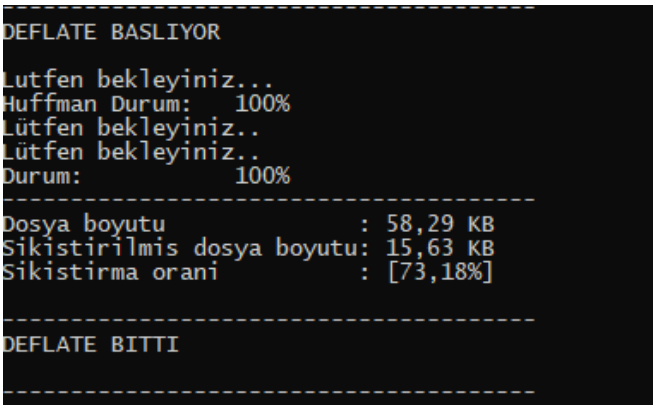
NOT: Durum değeri %5'er oranda ilerlemektedir.(0 – 5 – 10 …)



LZ77 Algoritması tamamlandığında sıkıştırma sonucu detaylı bilgiler verilmektedir.

- Dosya boyutu: Sıkıştırılması istenen dosyanın boyutu
- Sıkıştırılmış Dosya Boyutu: Sıkıştırılma işlemi tamamlandıktan sonra oluşturulan dosyanın boyutunu göstermektedir.
- Sıkıştırma Oranı: Orjinal dosya ile sıkıştırılmış dosya arasındaki sıkıştırma oranını göstermektedir. Bunu formül olarak göstermek gerekirse

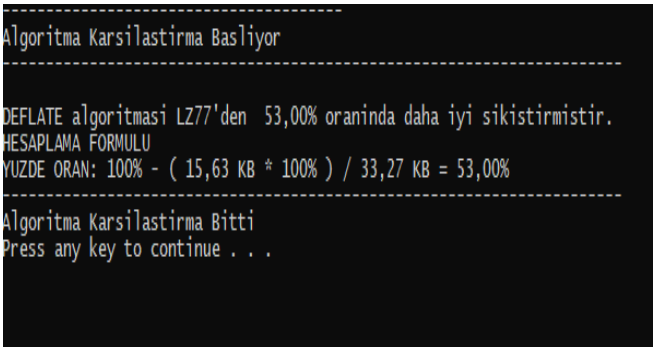
Sıkıştırma Oranı = 100 - ((Sıkıştırılmış Dosya Boyutu * 100) / Dosya boyutu)



Ardından DEFLATE algoritması için sıkıştırma işlemleri yapılmaktadır. İlk olarak Huffman kodları çıkartılır ve ardından LZ77 algoritması ile beraber çalışarak sıkıştırma işlemi yapılmaktadır.

- Dosya boyutu: Sıkıştırılması istenen dosyanın boyutu
- Sıkıştırılmış Dosya Boyutu: Sıkıştırılma işlemi tamamlandıktan sonra oluşturulan dosyanın boyutunu göstermektedir.
- Sıkıştırma Oranı: Orjinal dosya ile sıkıştırılmış dosya arasındaki sıkıştırma oranını göstermektedir. Bunu formül olarak göstermek gerekirse

Sıkıştırma Oranı = 100 - ((Sıkıştırılmış Dosya Boyutu * 100) / Dosya boyutu)



Ardından Algoritma karşılaştırması yapılmaktadır. Burada hangi algoritma daha iyi sıkıştırmışsa onu belirtmektedir. Belirlemenin ardından ne kadar iyi olduğu hesaplanmaktadır. Buradaki hesaplama formülü

YUZDE ORAN: 100% - (İyi sıkıştırma boyutu* 100%) / Kötü Sıkıştırma Boyutu

Şeklinde hesaplanmaktadır.

- İyi Sıkıştırma: Çıkan sonuca göre en az KB sahip olan sıkıştırma iyi sıkıştırma olarak adlandırıldı.
- Kötü Sıkıştırma: Çıkan sonuca göre en çok KB sahip olan sıkıştırma kötü sıkıştırma olarak adlandırıldı.

B. Çıktı Dosyaları Nerede ve Nelerdir

1) Çıktılar Nelerdir

5 adet çıktı dosyası oluşmaktadır.

1. DeflateNormal.txt: Deflate algoritmasının çıktılarının sıkıştırılmamış halinin çıktısı bulunmaktadır.
2. LZ77Normal.txt: LZ77 algoritmasının çıktılarının sıkıştırılmamış halinin çıktısı bulunmaktadır.
3. HuffmanAgaci.txt: Huffman algoritmasının sonucunda oluşan ağacın çıktısı bulunmaktadır.
4. metin.deflate: Deflate algoritması sonucu sıkıştırılmış dosyadır. İçerisinde Huffman ağacının verileri de bulunmaktadır.
5. metin.lz77: LZ77 algoritması sonucu sıkıştırılmış dosyadır.

2) Çıktılar Nerededir

Çıktılar .exe dosyasının yanında oluşturulan outputs klasörünün içerisindedir.

C. Dosyalar Sıkıştırılmış Çıktıya Nasıl Yazılıyor

LZ77 ile oluşturulan sıkıştırma dosyasın da:

Offset değeri 4 bit sola kaydırılıyor ve bu işlemden sonra oluşturulan sayıya length değeri ekleniyor. Bu işlem sayesinde maksimum 2 byte yer kaplayacak şekilde şifreleme oluşuyor ardından 1 byte karakteri ekliyoruz ve bu sayede metini 3 byte olarak sıkıştırmaya başlıyoruz.

Deflate ile oluşturulan sıkıştırma dosyasın da:

Önce metin dosyasını Huffman koduna çeviriyoruz. Ardından çevrilen metinde LZ77 algoritmasını yaparak sıkıştırmaya başlıyoruz.

IV. KABAKOD

LZ77

```
While(pointer > metindeki karakter sayısı)
{
  I = pointer – 1
    If(Eğer I = -1 ise )
    {
       Ilk karakteri kodla
    }
 Else
    {
     Arama tamponunda aranan harf varmı bul.
    }
 If(Arama tamponunda harf yoksa)
{
 Harfin length offset değerlerine 0 ver karakteri al ve kodla
} else
{
Ileri tampondaki benzerlik ile arama tamponundaki benzerlik değerlerini bul.
En büyük benzerlik değerini bularak offset length ve bir sonraki karakteri alarak kodla.
}
}
```

HUFFMAN

While(*pointer* > *metindeki karakter sayısıı*)
{
Karakterlerin frekans değerlerini oluştur.
}
 	
Karakterleri frekans değerlerine göre küçükten büyüğe sırala
 	
While(karakter array uzunlugu 1 olana kadar dön)
{
Ilk iki karakteri hafızaya al.
Küçük olanı ağacın soluna ekle
Büyük olanı ağacın sağına ekle
2 karakterin toplamını al ve bir karaktermiş gibi diziye kendisini ve yeni frekansı ekle
Karakter dizisinden küçük olan ilk değeri çıkar.
}
Oluşan Huffman metnini LZ77 algoritmasına aktar.

Sıkıştırma İşlemi (Offset , Length , Karakter)
Offset değerini al ve 4 bit sola kaydır. Oluşan yeni offset değerini Length değeri ile topla Karakter değerini al. Alınan değerleri bir struct'a aktar. Binary modda dosyaya oluşan verileri yaz.

V. KAYNAKLAR

[1] VERİ SIKIŞTIRMADA YENİ YÖNTEMLERAltan MESUT Doktora Tezi http://193.255.140.18/Tez/044455/METIN.pdf

[2] Lz77 Lz88 Sıkıştırma Algoritması, Ramazan Akbuz https://ramazanakbuz.com/lz77-lz88-sikistirma-algoritmasi/

[3] Rudra Singh LZ77 Basics, Encoding & Limitations | Data Compression https://www.youtube.com/watch?v=PqJR83O1DPE

[4] An Explanation of the Deflate Algorithm, Zlib https://zlib.net/feldspar.html

[5] Der LZ77 Algorithmus http://www.gm.fh-koeln.de/~hk/lehre/ala/ws0506/Praktikum/Projekt/D_rot/Ausarbeitung.pdf

[6] DCE LZ 77 Encoding and Decoding, Tuples Edu https://www.youtube.com/watch?v=cSyK2iCqr4w&t=333s

[7] Puff.C zLib github https://github.com/madler/zlib/blob/master/contrib/puff/puff.c

[8] Raul Fraile: How GZIP compression works | JSConf EU 2014 JSConf https://www.youtube.com/watch?v=wLx5OGxOYUc

[9] Understanding zlib, euccas https://www.euccas.me/zlib/#deflate

[10] HEXED.IT https://hexed.it/

[11] Deflate disassember to convert a deflate, zlib, or gzip stream into a readable form. madler/infgen https://github.com/madler/infgen

[12] Metin bulma konusunda en büyük yardımcım Lorem Ipsum https://www.lipsum.com/