**DOSYA SIKIŞTIRMA ALGORİTMALARI PROJESİ**

Derya ÖZGÜ, Mehmet KIYAK

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

[ozguderya@gmail](mailto:ozguderya@gmail).com, [ben@](mailto:ben@)mehmetkiyak.com.tr

Özet

*Projede LZ77 ve Deflate algoritmalarını kullanarak belirli bir metin dosyasının sıkıştırılma işlemi yapılmaktadır. Program metin.txt dosyasını içini okuyup daha sonra da iki algoritma ile de ayrı ayrı sıkıştırma işlemi yapmaktadır.*

*Programın amacı bu işlem sırasında LZ77 ve Deflate arasındaki sıkıştırma oranları ve sıkıştırma sürelerinin hesaplaması yapılmıştır. Daha sonra sıkıştırılan dosyalar output klasörünün içerisine kaydedilmiştir. Son olarak yapılan işlemler ekrana yazılarak işlem tamamlandı.*

# Giriş

Program ilk çalışma esnasında input klasörü içerisindeki metin.txt karakter karakter dosyası okuyup içindeki verileri ekrana yazmaktadır. Daha sonra sırayla önce LZ77 algoritması olmak üzere sıkıştırma işlemi için çalışmaktadır. LZ77 algoritması çalışması bittikten sonra girdi ve çıktı dosyalarının boyutlarını ekrana yazmaktadır.

LZ77 algoritması sonuçlandıktan sonra sıra Deflate algoritmasına geliyor. Aynı metin.txt dosyasının içindeki veriler okunarak önce Huffman algoritması ile sıkıştırma işlemi yapılıyor ardından çıkan sıkıştırılmış dosya LZSS algoritması ile sıkıştırılıyor ve Deflate algoritması sıkıştırma işlemi tamamlanmış oluyor. Girdi ve çıktı dosyalarının boyutlarını ekrana yazmaktadır.

# Temel Bilgiler

Program C programlama dilinde geliştirilmiş olup, tümleşik geliştirme ortamı olarak “CodeBlocks” kullanılmıştır. Sıkıştırma algoritmaları olarak proje gereği LZ77 ve Deflate (Huffman + LZSS) algoritmaları kullanılmıştır. Akış diyagramı için [draw.io](http://draw.io) online diyagram yapma programı kullanılmıştır

# Tasarım

Grafik olarak herhangi bir işlem bulunmamakta olup sadece konsol tarafında çalışmaktadır. Sıkıştırma algoritmaları programının programlanma aşamaları altta belirtilen başlıklar altında açıklanmıştır.

# Algoritma

Program öncelikle metin.txt dosyası içerisindeki verilerin karakter karakter okuma işlemlerini yapmaktadır ardından ekrana okuduğu verileri ekrana yazmaktadır. Ardından metin.txt dosyası LZ77 algoritmasının içine gönderilmektedir. LZ77 algoritması metin.txt dosyasının önce var olan boyutunu tutmakta ardından meitn.txt dosyasındaki verileri okuyup her bir karaktere uygun sıkıştırma işlemi yapmaktadır. Son olarak girdi ve çıktı dosyalarının boyutunu ekrana yazdıktan sonra algoritmanın dosyayı sıkıştırma süresi de ekrana yazılmaktadır.Sıra Deflate algoritmasına gelindiğinde metin.txt dosyası önce Huffman algoritması ile sıkıştırma işlemi yapılmaktadır. Ardından sıkıştırılan dosya LZSS(LZ77) algoritmasına gönderiliyor. LZSS algoritması sıkıştırma işlemi sonucunda girdi ve çıktı dosyalarının boyutlarını ekrana yazdıktan bu sıkıştırma algoritmasının yani deflate’in ne kadar sürede işlem yaptığı ekrana yazılmaktadır.

Problemin açıklamaları şu şekilde ifade edilmektedir;

1. **Giren Metin Dosyasının Okunması:** metin.txt dosyasındaki verilerin karakter karakter okunup ekrana yazılma işlemi yapılmaktadır.
2. **Sıkıtırılma İşleminin Yapılması:** Belirtilen LZ77 ve DEFLATE sıkıştırılma algoritmalarının işlemleri yapılmaktadır.
3. **Çıkış :** Yapılan sıkıştırma işleminin sonucunda iki algoritmanın çıktı dosyalarını yani sıkıştırılmış dosyaları output klasörüne atmaktadır. Ardından yapılan sıkıştırma işlemleri sonucunda sıkıştırma oranları ve süreleri ekrana yazarak kullanıcı bilgilendirilmiştir.

# Kullanılan Fonksiyonlar ve Tanımları

* + 1. **main() :** Ana fonksiyon olup gerekli bütün işlemlerin yapıldığı main fonksiyonunda öncelikle metin dosyasındaki veri okuma işlemi yapılmakta ve ekrana yazdırılmaktadır. Ardından metin.txt dosyası LZ77 ve Deflate fonksiyonlarına gönderilerek sıkıştırma işlemleri başlamaktadır.
    2. **FileReadChar(char\* inputPath):** İçerisine gönderdiğimiz metin.txt dosyasının karakter karakter ekrana yazmak için yazmış olduğumuz fonksiyondur.
    3. **LZ77(char\* inputPath) :** LZ77 algoritmasının ilk ana fonksiyonudur. LZ77 algoritması için gerekli fonksiyonların yönlendirildiği sıkıştırma süresinin başlayıp son bulduğu fonksiyondur.
    4. **LZ77\_DosyaSikistir(char\* inputPath, int m):**  LZ77(inputPath) fonksiyonu içerisine gelen metin.txt dosyası bu fonksiyonu gönderilir. Bu fonksiyonda metin dosyasının girdi boyutu alınır ardından içerisindeki veriler okunur ve sıkıştırma işlemi karakter karakter işlenerek yapılır işlem sonucunda çıktı dosyasının boyutuda ekrana yazılır. m değerinin 1 gelmesi durumunda LZ77 algoritması için m değerinin 2 gelmesi durumunda Deflate algoritması için çalışmaktadır.
    5. **DEFLATE(char\* inputPath) :** Deflate algoritmasının ilk ana fonksiyonudur. Deflate algoritması için gerekli fonksiyonların yönlendirildiği sıkıştırma süresinin başlayıp son bulduğu fonksiyondur.
    6. **DEFLATE\_DosyaSikistir(char\* inputPath) :** DEFLATE(inputPath) fonksiyonu içerisine gelen metin.txt dosyası bu fonksiyonu gönderilir. Karakterler\_ve\_Frekanslar fonksiyonu çalışmaktadır. Ardından onu içinde yapılan işlemler sonucunda içerisinde bulduğumuz karakterler ve frekansları Huffman algoritması çalışmaktadır
    7. **Karakterler\_ve\_Frekanslar() :** Daha önce **FileReadChar** fonksiyonda okuma işlemi sırasında verileri char str nesnesi içinde tutmuştuk bu fonksiyon içerisinde bu str nesnesi içindeki metini karakter olarak okuyup karakterlerini ve bu karakterlerin frekanslarını tutup karakterler ve frekanslar dizilerine yazılmaktadır.
    8. **frenkas\_bul(char ch, char str[]):** İçerisine gönderdiğimiz karakterin str nesnesi içerisindeki frekans değerine bakmaktadır.
    9. **Huffman(char veri[], int frekans[], int boyut):** Huffman sıkıştırma algoritmasının ilk başladığı yer olan bu fonksiyonda ilk önce içerisine gelen verilerin Hufmman\_Agaci\_Olustur fonksiyonuna gönderilerek huffman ağacının oluşturulması ve sıkıştırma işlemi yapılmaktadır. Ardından fonksiyona geri dönerek bu LZ77(LZSS) algoritmasının çalışması için LZ77\_DosyaSikistir fonksiyonu çağırılmıştır.
    10. **Hufmman\_Agaci\_Olustur(char veri[], int frekans[], int boyut) :** İçerisine gelen veriler ile huffman ağacı oluşturulmuştur. Yigin ve YiginDugumu struct’ları ağacımızın gövdesi ve yapraklarını tutuyor ve girilen veriler buralarda tutulmaktadır. İçerisinde birden fazla fonksiyonu çalıştırarak ağacımızın verilerinin struct’lara yazılaması işlemi yapmaktadır. Bu fonksiyonları alt başlıklarla açıklamak gerekirse;
  1. **Yigin\_Dugumu\_Olustur(char veri[], int frekans[], int boyut) :** İçerisine gönderdiğimiz verilerin ağaçlar üzerindeki yerlerini oluşturmakta ve verileri yerleştirmektedir.
  2. **Dugume\_Deger\_Ata(struct Yigin\* yigin) :** İçerisine gönderdiğimiz düğümün boyut değerine bakmakta ve bize true veya false değeri döndürmektedir.
  3. **Dugumden\_Cikar(struct Yigin\* yigin):** İçerisine gönderidiğimizi düğümü Yigin ağacından çıkarmaktadır.
  4. **Yeni\_Dugum(char veri, unsigned frekans):** İçerisine gönderilen veri ve frekans değerilerini kullanarak ağacımız da yeni bir düğümün oluşturmasını sağlamaktadır.
  5. **Dugum\_Ekle (struct Yigin\* yigin,struct YiginDugumu\* yigin\_dugumu):** İçerisine gönderilen verileri kullanarak Yigin düğümündeki boyutunu bir arttırarak YiginDugumu’ndeki yerini de düzenlemektedir.
  6. **Dugumden\_Cikar(struct Yigin\* yigin):**  İçerisine gelen ağacın Yigin düğümü üzerindeki verilerinin silinmesini ve ağacının dengelenmesi için minHeapify fonksiyonu çalıştırmaktadır.
  7. **minHeapify(struct Yigin\* yigin, int idx):** İçerisine gelen ağacın sağ ve sol yapraklarını kontrol ederek ağacın dengelenmesini sağlamaktadır. Ayrıca veri değişikliği içinde Dugumleri\_Takas\_Et fonksiyonunu kullanmaktadır.
  8. **Dugumleri\_Takas\_Et(struct YiginDugumu\*\* a, struct YiginDugumu\*\* b) :** İçerisine gelen iki yaprağın yerlerinin değiştirme işlemi yapmaktadır.
  9. **buildYigin(struct Yigin\* yigin):** Ağacın dengelenmesi için kullanılan bir fonksiyondur.
  10. **Yaprak\_Mi(struct YiginDugumu\* root) :** İçerisine gelen düğümün sağ ve sol düğümleri kontrol edilerek ağacın yaprağı olup olmadığı kontrol edilir.
  11. **printCodes(struct YiginDugumu\* root, int arr[], int top):** Ağacın bütün verilerinin removefile.txt dosyasına yazılması için bu fonksiyon ile ağaç üzerinde dolaşıyor ve yazması için printArr fonksiyonu çalışmaktadır.
  12. **printArr(int arr[], int n):** Gelen verileri removefile.txt dosyasına yazılma işlemini üstlenmektedir.

**3.2.11 findMatch(unsigned char window[], unsigned char str[], int strLen):** Okunan ve işlenen veriler bu fonksiyonda Huffman algoritmasının gereği 0 ile 1 lerden oluşmasını sağlayan fonksiyondur.

# Sonuçlar

Programımız dosya okuma ve yazma işlemi esnasında doğru olarak çalışmaktadır. Kullanıcı isterlerine ve gidilecek şehirlerin belirlenmesi konusunda test edilmiş olup doğru yanıtlar vermiştir. Fakat iki şehir arasındaki en kısa mesafeyi bulmaya çalışırken programı sonlandırmaktadır yani testler başarısız olmuştur.

# Kaynakça

1. https://zlib.net/feldspar.html/ kaynağından deflate algoritması algoritması hakkında bilgi edinildi.

# 6. Denklemler

Program içerisinde matematiksel olarak bir hesaplama bulunmadığı üzere hiçbir denklem kullanılmamıştır.

