PROGRAMLAMA LABORATUVARI 1, PROJE 3

Semanur Bağcı, Ece Peker

### [semanurbagci0@gmail.com](mailto:semanurbagci0@gmail.com) , [eceepeker@gmail.com](mailto:eceepeker@gmail.com)

KOCAELİ ÜNİVERSİTESİ/MÜHENDİSLİK FAKÜLTESİ/BİLGİSAYAR

MÜHENDİSLİĞİ BÖLÜMÜ

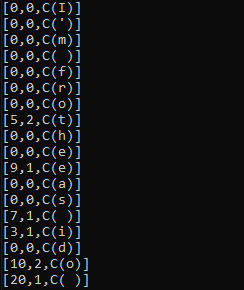
I. GİRİŞ

Projede bizden istenen metin sıkıştırma algoritmaları olan LZ77 ve DEFLATE hakkında araştırma yapıp yazdıktan sonra performanslarını birbiriyle karşılaştıran bir C programı oluşturmaktır. Projenin amacı ise biz öğrencilerin algoritma oluşturma ve çözümleme konularını anlayıp anlamadığını değerlendirmektir.

II. ÖZET

Programın çalışma mantığını özetleyecek olursak; algoritma bulunduğu klasördeki txt dosyasını fgetc() yardımıyla karakter karakter okuduktan sonra, sag ve sol adında iki temel kısım oluşturur. LZ77 algoritmasının çalışmasına göre, malloc ve realloc yardımıyla oluşturulan sağ tarafa, karakterleri atar. Sağ taraftan sol tarafa doğru metnin ilk indisini atarak sağ taraftaki metni sol tarafta arar.

Ekrandaki çıktı ise [uzaklik,benzerlik,C(karakter)] şeklindedir.Herhangi bir benzerlik bulunmazsa çıktı [0,0,C(karakter)] şeklinde olur.



DEFLATE algoritması için ise; kodun sıkıştırılmış hali

bir char dizisinde tutulur, kodun sıkıştırılmış halindeki

harflerin tekrar etme sayıları hesaplanır ve bu sayılarla

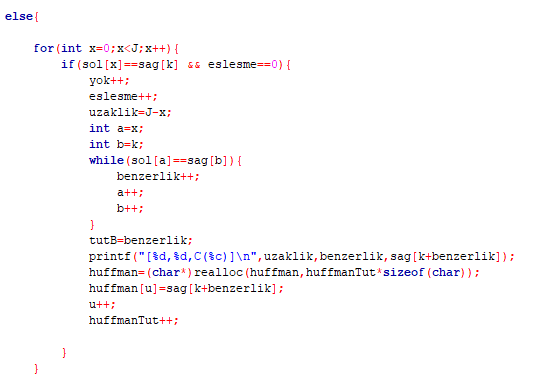
harfler kullanılarak fonksiyonlar yardımıyla sıralı bir

linked-list oluşturulur. Kodun devamında

huffmanKodlama() fonksiyonu ile DEFLATE algoritmasını

da yazmayı planladık ama teslim tarihine kadar bu kısmı

tamamlamayı başaramadık ne yazık ki.



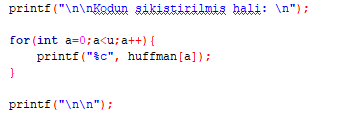
III. Yöntem

Projenin kodunu yazmaya başlamadan önce projenin içeriği olan LZ77 ve DEFLATE algoritmasının temeli olan Huffman kodlaması hakkında araştırma yapıp bilgiye ulaşmaya çalıştık.

Önce proje sırasında bize yardımcı olacak bazı structları ve fonksiyonları oluşturduk. Dosya okuma işlemiyle önce txt dosyasındaki bilgileri karakter karakter okuyup LZ77 algoritmasına uygun bir şekilde iki kısım oluşturduktan sonra daha kısa bir şekilde olmasını sağlayacak şekilde metni sıkıştırma işlemini yapmış olduk. Daha sonra ise LZ77 ile Huffman algoritmasının birleşimi olan DEFLATE algoritmasını yazmak için ise Huffman’ın temeli olan ikili arama ağacını oluşturmaya çalıştık. Özette de belittiğimiz şekilde, huffman kodlama algoritmasını tamamlayamadık ama yapabildiğimiz ve yapmaya çalıştığımız şeyleri sizlere de göstermek adına bu fonksiyonları silmeyip yorum satırına almak istedik.

IV. SONUÇ

Bu proje sayesinde dosya okuma, dinamik bellek kullanımı, bağlı liste, döngüler ve algoritma oluşturma mantığı hakkında pek çok bilgi edinerek C dili ve programlamada kendimizi geliştirdik. Ayrıca daha önce hiç bilmediğimiz tamamen yeni bir algoritma olan LZ77 ve DEFLATE algoritmaları hakkında tamamen bilgi sahibi olmuş olduk.



V. KULLANILAN KÜTÜPHANELER, STRUCT YAPILARI VE BAZI FONKSİYONLAR

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <malloc.h>

struct huffman{

char deger;

int tekrar;

int toplandiMi;

struct huffman\* next;

struct huffman\* left;

struct huffman\* right;

};

1. struct huffman\* dugumOlustur(char harf, int T)

{

…

}

1. void ekle(char harf, int T)

{

struct huffman\* eklenecek = dugumOlustur(harf, T);

if(start==NULL){

start=eklenecek;

else{

...

}

}

[3] void yazdir(){

temp=start;

while(temp!=NULL){

printf("%c-->%d\n",temp->deger,temp->tekrar);

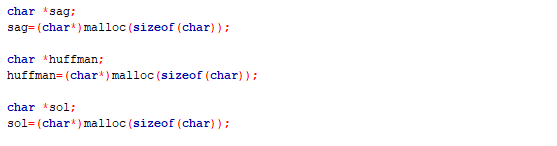
temp=temp->next;

}

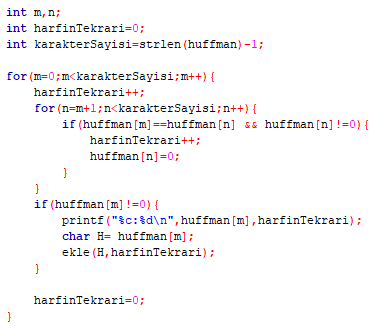
}

VI. PROJE YAPIM AŞAMASINDA KARŞILAŞILAN SIKINTILAR

Proje yapım aşamasında önce LZ77 hakkında araştırmalar yapıp çeşitli videolar izledik. Kodu yazmaya başlamadan önce ise yaptığımız araştırma sonucu bulduğumuz bir koddan yararlanmaya çalıştık. Önce kodu aşama aşama inceledik, ne yapılması gerektiğini anladık ve kendi kodumuza uyarlanmış bir şekilde yazmayı denedik. Fakat bizim araştırdığımız diğer LZ77 kodlarından farklı olduğunu gördük ve kopya ihtimaline karşı bu koddan tümüyle yararlanmaktan vazgeçtik. Bunun yerine izlediğimiz bir videoda algoritmanın mantığını anlayıp, geçmiş C bilgilerimizle kodu yazmaya başladık ve bu sıkıntıyı çözmüş olduk. Kodu sıfırdan yazdığımız için çeşitli hatalarla karşılaştık. Bunlardan birisi ise bellek problemiydi. Oluşturduğumuz char dizilerine boyut atamanın mantıksız olacağını düşündük, çünkü bize verilecek metnin boyutu belirsiz olacaktı. Bu yüzden dinamik bellek konusunda biraz daha araştırma yaparak projemizdeki yapıları malloc/realloc yardımıyla dinamik bir hale getirmiş olduk.



LZ77 algoritmasını tamamladıktan sonra DEFLATE temeli olan Huffman algoritmasında da aynı sıkıntıyı yaşadık. İnternetteki kodlar mantığımıza bir türlü uymadı ve kendimiz sıfırdan yazmaya karar verdik. Sıkıştırılmış metindeki her bir karakterin sıklığını bulup, herhangi bir sıralama algoritması kullanıp gruplandırdıktan sonra binary search tree kullanarak çıktı elde etmeyi amaçlamıştık. Başta çeşitli algoritmalar denesek de selection sort gibi sıralama algoritmaları kullanmak yerine bağlı liste yapısını kullanaraak ve başa/sona/araya ekle diyerek tekrar değerlerini sıraya sokmayı başardık. Fakat gruplandırma ve daha sonrasında BST oluşturma kısmını fonksiyonları yazmamıza rağmen maalesef yapamadık. Bu yüzden de iki algoritmayı karşılaştırma kısmı eksik kalmış oldu.



VII. KABA KOD

Adım 1 - Başla.

Adım 2 - fopen ile dosyayı aç.

Adım 3 - if(dosya==NULL) return 0;

Adım 4 - sag,sol,huffman pointerları için malloc ile yer aç.

Adım 5 - fgetc() ile bir karakter oku.

Adım 6 - realloc ile sağ tarafın yerini büyüt.

Adım 7 - fclose(dosya);

Adım 8 - if(ilk==0) printf("[0,0,C(%c)]\n",sag[k]);

Adım 9- else printf("[%d,%d,C(%c)]\n",uzaklik,benzerlik,sag[k+benzerlik]);

Adım 10 - if(yok==0) printf("[0,0,C(%c)]\n",sag[k]);

Adım 11 - free(sag), free(sol);

Adım 12 - karakterSayisi=strlen(huffman)-1;

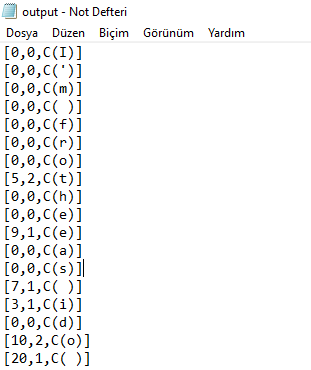
Adım 13 - İç içe 2 for yardımıyla harf tekrarlarını bul.

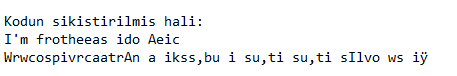
If(huffman[m]==huffman[n] && huffman[n]!=0)

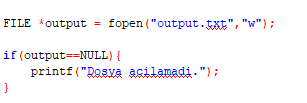
Adım 14 - if(huffman[m]!=0) ekle fonksiyonu ile bağlı listeye karakteri ve sıklığını gönderip ekle.

Adım 15 - yazdir();

Adım 16 - Bitir.







KAYNAKLAR

1. <https://ysar.net/algoritma/lz77.html>
2. <https://ysar.net/algoritma/huffman-kodlamasi.html>
3. <https://www.youtube.com/watch?v=0vMyw4Lv3Hw>
4. <https://www.youtube.com/watch?v=DGi_gSfYfKo&list=PLh9ECzBB8tJN9bckI6FbWB03HkmogKrFT&index=3&t=0s>
5. <https://drive.google.com/drive/folders/1RWlSaFEaXei4_7trdfUlwBnbv18S2zEs>
6. <https://www.bilgigunlugum.net/prog/cprog/c_dinamik>