

**VERİ YAPILARI**

**VE**

**ALGORİTMALAR**

**ÖDEV-2**

**HUFFMAN AĞACI OLUŞTURULMASI**

**17011701 Onur KOÇKAN**

**PROGRAMIN ADIM ADIM AÇIKLANMASI VE YÜRÜTÜLMESİ:**

Programın temelinde 4 ana işlev yer alır bunlar;

1-Kullanıcıdan kelime alınır ve bu kelime karakter karakter frekanslarıyla birlikte ayrılarak ayrı ayrı linkli listeye eklenir:

Kullanıcıdan klavyeden bir kelime girilmesi istenir ve kullanıcı kelimeyi girdikten sonra 0dan 256a kadar olan ascii tablosuna göre mevcut bulunan kelimelerin indislerinin frekansları arttırılır daha sonra linkli listeye eklenirkenki döngü 0dan 256a kadar döner ve frekansı olan indisleri karakter olarak frekanslarıyla birlikte linkli listeye baştan ekleme yapan fonksiyona gönderilir.

2-Linkli listeye eklenen elemanlar Insertıon Sort algoritması ile sıralanır:

Insertion Sort algoritmasi ile linkli liste sıralanır başlangıçta global olarak tanımlanan heade her fonksiyonda olduğu gibi bu fonksiyonda da erişilir ve sıralama yapan bu fonksiyon herhangi bir parametre almaz veya herhangi bir dönüş değerine sahip olmaz.

3-Huffman ağacı oluşturulur:

Sıralanan linkli listeden belirtilen şekilde huffman ağacı oluşturulur.2 şart vardır elde tutulan değer ve bir sonraki değerlerin frekansının toplamı ya herhangi iki nodeun arasındadır veya hepsinden büyük yani en sondadır.Arasında olduğu taktirde ilgili fonksiyondaki ilk if bloğunda araya ekleme işlemi yapılır.Örnek: 1a 1b 2c 3e ise,bir sonraki adım 1a+1b=2 bu node left(1a) ve right(1b) nodeları atanarak 2c ve 3e nodelarının arasına root olarak yeni bir node olarak eklenir.Bir diğer şart ise toplanan ilk iki nodeun atanması gerekilen yerin en sonda olmasıdır(2. İf bloğu)bu şarta uyan yeni node listenin en sonuna toplamlarından elde edilen nodeları lefti ve righti olmak suretiyle ayarlanarak eklenir.

4-Oluşturulan huffman ağacı ekrana istenilen şekilde yazdırılır:

Huffman ağacını yazdırırken 2 adet recursive ve 1 adet iterative fonksiyon kullanılır bunlar;ağacın levelini öğrenme,mevcut leveldeki tüm nodeları yazdırma(recursive) ve ağacın leveli kadar dönen bir for döngüsü içerisinde spesifik olarak gönderilen leveldeki nodelar yazdırılması(iterative) şeklindedir.

Programda main fonksiyonu hariç total olarak 7 fonksiyon bulunmaktadır bunlar;

1- **void insert(char harf,int freq):**parametreleri verilen değişkenler ile linkli listeye eleman eklenilmesini gerçekleştirir.

2- **void insertionSort():**Linkli listedeki bulunan elemanların küçükten büyüğe Insertıon Sort algoritması ile sıralanmasını yapan fonksiyondur.

3- **void createHuffman():**Küçükten büyüğe sıralanmış olan linkli listemizin huffman ağacına uyarlanmasını yapan fonksiyondur.

4- **void printLinkedList():**Linkli listemizi ekrana yazdıran fonksiyon.

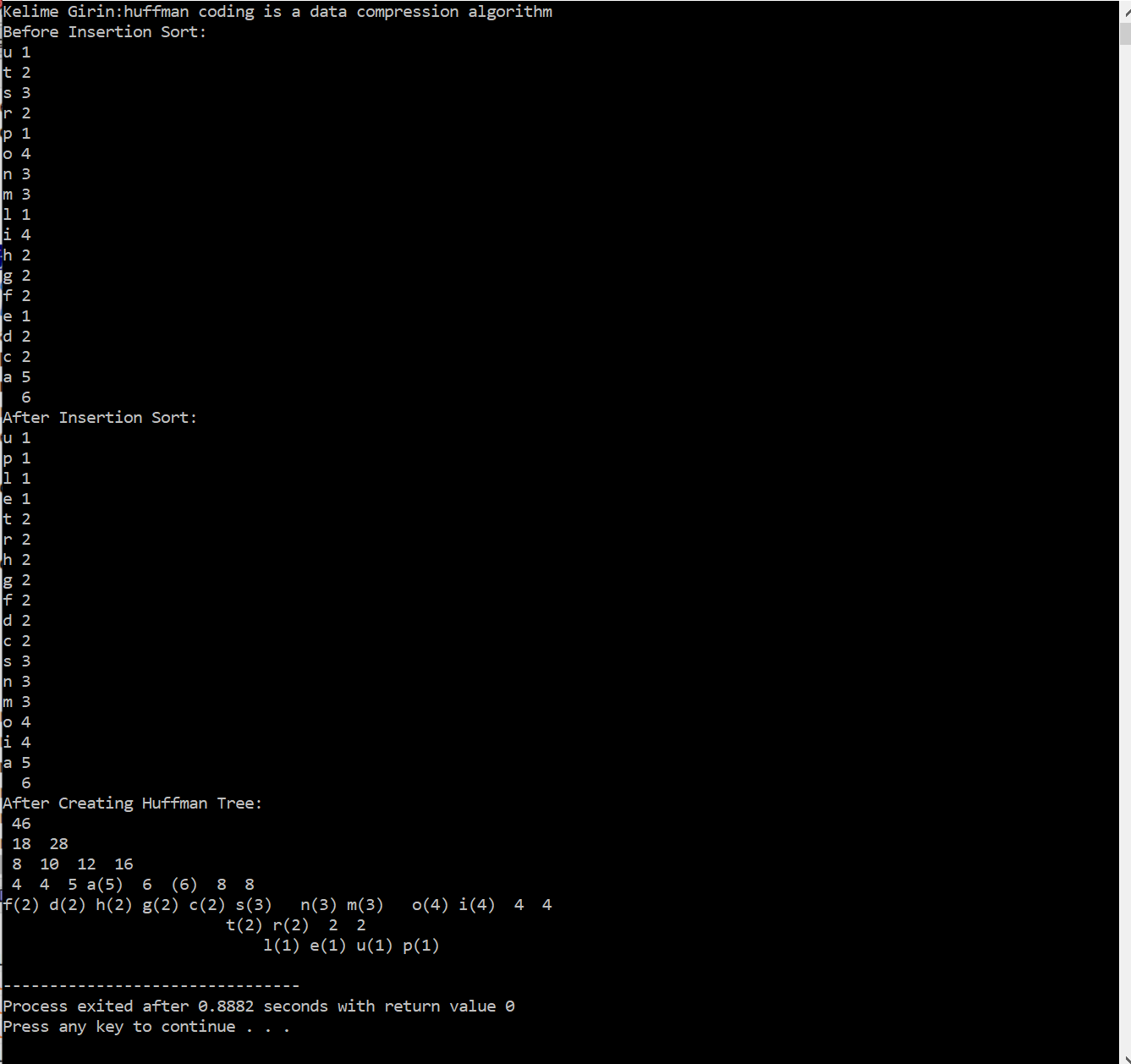
5- **int levelSayisi(struct node \*startNode):**Huffman ağacımızın uzunluğunu ölçen fonksiyon.

6- **void printLevel(struct node \*head, int level):**Gönderilen parametreler ile huffman ağacının mevcut seviyesindeki nodeları yazdıran fonksiyon.

7- **void printHuffmanTree():**Huffman ağacının yazdıran fonksiyon.

**EKRAN GÖRÜNTÜLERİ:**

**Örnek1:**

****

**PROGRAMIN KODLARI:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stddef.h>

struct node**{** // BELIRTILEN SEKILDE OLUSTURULAN STRUCT YAPIMIZ

char harf**;**

int freq**;**

struct node**\*** left**;**

struct node**\*** right**;**

struct node**\*** next**;**

**};**

struct node **\***head**=NULL;** //GLOBAL OLARAK TANIMLANAN ILK NULLA ESIT OLAN BOS NODEUMUZ

void insert**(**char harf**,**int freq**){** //BASA NODE EKLEME

**if(**head**==NULL){**//ILK ELEMAN EKLENIRKEN

struct node**\*** temp**;**

temp**=(**struct node**\*)**malloc**(sizeof(**struct node**));**

temp**->**harf**=**harf**;**

temp**->**freq**=**freq**;**

temp**->**next**=NULL;**

temp**->**left**=NULL;**

temp**->**right**=NULL;**

head**=**temp**;**

**}**

**else{** // IKINCI VE DAHA SONRASINA AIT OLAN ELEMANLARIMIZI EKLERKEN

struct node**\*** temp**;**

temp**=(**struct node**\*)**malloc**(sizeof(**struct node**));**

temp**->**harf**=**harf**;**

temp**->**freq**=**freq**;**

temp**->**next**=**head**;**

temp**->**left**=NULL;**

temp**->**right**=NULL;**

head**=**temp**;**

**}**

**}**

void insertionSort**()** **{**

**if(**head**!=NULL){** //LINKLI LISTEMIZIN BOS OLMADIGINDAN EMIN OLMAMIZ ICIN YAPILAN KONTROL

struct node **\***temp **=** **(**head**)->**next**;**

char tempHarf**,**harfTemp**;**

int freq**,**freqTemp**,**count**;**

**while(**temp **!=** **NULL** **)** **{** //TEMP STRUCTIMIZA ATILAN HEADIMIZIN NEXTI BOS DEGILSE

freq **=** temp**->**freq**;**

tempHarf **=** temp**->**harf**;**

count **=** 0**;**

struct node **\***tempp **=** head**;**

**while(**tempp **!=** temp**)** **{** //IKINCI TANIMLANAN TEMPP STRUCTIMIZIN ILK TANIMLANAN TEMP STRUCTINA ESIT OLMAMASI DURUMUNDA;

**if((**tempp**->**freq **>** temp**->**freq**)** **&&** count**==**0**)** **{** //TEMPPIN FREKANSI ILK TEMPIN FREKANSINDAN BÜYÜKMÜ VE KONTROL İÇİN OLUSTURDUGUMUZ COUNT DEGISKENININ DEGERI 0MI(AND)

freq **=** tempp**->**freq**;** //IKINCI OLUSTURDUGUMUZ TEMPP STRUCTIMIZ ILKINDEN BUYUK OLDUGU ICIN GEREKLI DEGISIMLER YAPILIYOR

tempHarf **=** tempp**->**harf**;**

tempp**->**freq **=** temp**->**freq**;**

tempp**->**harf **=** temp**->**harf**;**

tempp **=** tempp**->**next**;** //KONTROLUN DEVAMI ICIN TEMPP DEGISKENIMIZ BI SONRAKI ELEMANA GIDIYOR

count **=** 1**;** //BIR SONRAKI BLOGU YAKALAMASI ICIN BIR NEVI SWITCH GOREVINI USTLENEN DEGISKENI 1LIYORUZ BOYLELIKLE BU ŞART SAĞLANMADIĞI ZAMAN GEÇİCİ DEĞERLERİMİZİN TUTULMA ISLEMLERININ

//YAPILDIGI BIR SONRAKI BLOGA YAKALANSIN.

**}**

**else{**

**if(**count **==** 1**)** **{**//GEREKLI YEDEKLEME ISLEMLERI

freqTemp **=** freq**;**

harfTemp **=** tempHarf**;**

freq **=** tempp**->**freq**;**

tempHarf **=** tempp**->**harf**;**

tempp**->**freq **=** freqTemp**;**

tempp**->**harf **=** harfTemp**;**

**}**

tempp **=** tempp**->**next**;**

**}**

**}**

tempp**->**freq **=** freq**;**

tempp**->**harf **=** tempHarf**;**

temp **=** temp**->**next**;**

**}**

**}**

**}**

void createHuffman**(){**

int temp**;**

struct node **\***count**;**

count**=**head**;**

**if(**head**->**next**!=NULL)**//LINKLI LISTEDE BIR ELEMAN YOK ISE

temp**=**head**->**freq**+**head**->**next**->**freq**;** //ILK IKI FREKANSIN TOPLAMI

**while(**count**!=NULL** **&&** temp**!=NULL){** //ITERASYON YAPTIGIMIZ STRUCTIMIZ VE TOPLAM DEGERIMIZIN TUTULDUGU DEGISKEN NULL OLMADIGI SURECE;

**if(**count**->**next**!=NULL** **&&** temp **>=** count**->**freq **&&** temp **<=** count**->**next**->**freq**){** //ITERASYON YAPTIGIMIZ COUNT STRUCTIMIZIN SIRADAKI DEGERI NULL OLMIYACAK VE TEMPDE TUTULAN FREKANS DEGERI MEVCUT COUNTDA BULUNAN FREKANS DEGERI VE BIR SONRASINDAKI

//NODEUN FREKANS DEGERININ ARASINDAYSA O ARAYA YERLESTIRME ISLEMLERININ YAPILDIGI BLOK(ROOT OLUSTURMA)

struct node **\***NODE**=(**struct node**\*)**malloc**(sizeof(**struct node**));**

NODE**->**next**=**count**->**next**;**

NODE**->**left**=**head**;**

NODE**->**right**=**head**->**next**;**

count**->**next**=**NODE**;**

NODE**->**freq**=**head**->**freq**+**head**->**next**->**freq**;**

NODE**->**harf**=NULL;**

head**=**head**->**next**->**next**;**

count**=**head**;**

temp**=**head**->**freq**+**head**->**next**->**freq**;**

**}**

**else** **if(**count**->**next**==NULL** **&&** temp **>** count**->**freq**){** //EGERKI DEGER ARADA DEGIL SONDAYSA,ROOTUN SONA EKLENDIGI BLOK

struct node **\***NODE**=(**struct node**\*)**malloc**(sizeof(**struct node**));**

NODE**->**next**=NULL;**

NODE**->**left**=**head**;**

NODE**->**right**=**head**->**next**;**

count**->**next**=**NODE**;**

NODE**->**freq**=**head**->**freq**+**head**->**next**->**freq**;**

NODE**->**harf**=NULL;**

head**=**head**->**next**->**next**;**

count**=**head**;**

**if(**head**->**next**!=NULL){**

temp**=**head**->**freq**+**head**->**next**->**freq**;**

**}**

**}**

**else{**//ITERASYON

count**=**count**->**next**;**

**}**

**}**

**}**

void printLinkedList**(){** //SADECE LINKLI LISTEYI BASTIRIR

struct node **\***temp**;**

temp**=**head**;**

**while(**temp**!=NULL){**

printf**(**"\n%c %d"**,**temp**->**harf**,**temp**->**freq**);**

temp**=**temp**->**next**;**

**}**

**}**

int levelSayisi**(**struct node **\***startNode**){** //HUFFMAN AGACIMIZIN UZUNLUGUNU DONDUREN FONKSIYON

**if** **(**startNode **==** **NULL){** // GELEN BASLANGIC ROOTU MEVCUT DEGILSE 0 UZUNLUKLU

**return** 0**;**

**}**

int maxLeft **=** 1 **+** levelSayisi**(**startNode**->**left**);** //0 DEGILSE ILK KADEME ICIN 1 + LEFTI GONDERIR VE DALLANMA BU SEKILDE LEFTIN ROOT ROLUNU ALARAK TEKRAR MAX SOL VE MAX SAG DOGRU BULUP DONDURMESINI SAGLAR

int maxRight **=** 1 **+** levelSayisi**(**startNode**->**right**);** //RECURSIVE OLARAK AYNI ISLEMIN SAG NODE ICIN YAPAR.

**if** **(**maxLeft **>** maxRight**){** //TUM NODELARIN SAG VE SOLLARINA INILIP EN DERIN SAG TARAFMI SOL TARAFMI DIYE KARSILASTIRMA YAPILDIGI YERDIR SOL ISE SOL ICIN TOPLANAN DEGERI,SAG ISE SAG ICIN TOPLANAN DEGER DONDURULUR.

**return** maxLeft**;**

**}**

**else{**

**return** maxRight**;**

**}**

**}**

void printLevel**(**struct node **\***head**,** int level**){** //BU FONKSIYONDA EKRANA SADECE GONDERILEN NODEUN BULUNDUGU LEVELDEKI NODELARI YAZDIRILIR

**if** **(**head **!=** **NULL** **&&** level **==** 0**){** //YOLLANAN HEADIN BOS OLMAMASI VE LEVEL SAYISINIIN YA 0 YADA EKSILEREK 0LANMASI DURUMUNDA MEVCUT NODE YAZDIRILIR

**if(**head**->**harf**!=NULL)**

printf**(**"%c(%d) "**,** head**->**harf**,**head**->**freq**);**

**else**

printf**(**"%c%d "**,**head**->**harf**,**head**->**freq**);**

**}**

**else** **if** **(**head **!=** **NULL){** //YUKARDAN FARKLI OLARAK EGERKI LEVEL 0DAN FARKLI(DAHA BUYUK) ISE 0LIYANA KADAR RECURSIVE OLARAK AZALTILIR VE SIRASIYLA SOL VE SAG NODELARI ASSAGI DOGRU DALLANARAK ILK BLOGU YAKALAMASI SAGLANIR VE YAZDIRILIR.

printLevel**(**head**->**left**,** level **-** 1**);**

printLevel**(**head**->**right**,** level **-** 1**);**

**}**

**else** **if(**head**==NULL)** // EN SON OLARAK ULASAN NODE MEVCUT DEGILSE(TUM ELVELLERIN YAZDIRILIP EN SON CHILDSIZ BI ROOT OLMASI) EKRANA BOSLUK YAZDIRILIR.

printf**(**" "**);**

**}**

void printHuffmanTree**(){** //HUFFMAN AGACIMIZIN YAZDIRILDIGI FONKSIYON

int i**,**level**;** //DONGU ICIN i DEGISKENI,KAC DEFA DONMESI GEREKTIGINI(AGACIN SEVIYESINI) TUTAN level DEGISKENI.

level **=** levelSayisi**(**head**);** //level DEGISKENIMIZE AGACIMIZIN level SAYISI ATILIR.

**for** **(**i **=** 0**;** i**<**level**;** i**++){** //O KADAR DONEREK HER LEVEL YUKARIDAN ASSAGIDA DOGRU DALLANARAK AYRI AYRI SPESIFIK OLARAK YAZDIRILIR.

printLevel**(**head**,** i**);**

printf**(**"\n"**);**

**}**

**}**

int main**(**int argc**,** char **\***argv**[])** **{**

char str**[**256**];**

int i**,**freq**[**256**]** **=** **{**0**};**

printf**(**"Kelime Girin:"**);**

gets**(**str**);** //girilen kelime str stringine atılır.

**for(**i **=** 0**;** str**[**i**]** **!=** '\0'**;** i**++){** //elimizdeki stringin sonuna ulasincaya kadar dönen bir döngü

freq**[**str**[**i**]]++;** //mevcut bulunan karakterlerin ayrı ayrı frekansının arttırılması.

**}**

**for(**i **=** 0**;** i **<** 256**;** i**++){** //ascii karakter tablosunda bulunan karakterleri taramak için 0dan 256ya kadar giden döngü.

**if(**freq**[**i**]** **!=** 0**){** //frekansı 0dan farklı(frekansı olan) olan karakterler için;

insert**(**i**,**freq**[**i**]);** //karakterin,i=ascii tablosundaki numarasını ve frekansını parametre olarak insert fonksiyonuna gönderir ve linkli listeye başa ekleme işlemi gerçekleştirilir.

**}**

**}**

printf**(**"Before Insertion Sort:"**);**

printLinkedList**();**

insertionSort**();**

printf**(**"\nAfter Insertion Sort:"**);**

printLinkedList**();**

printf**(**"\nAfter Creating Huffman Tree:"**);**

createHuffman**();**

printf**(**"\n"**);**

printHuffmanTree**();**

**return** 0**;**

**}**