BAĞLI LİSTE İLE KELİME SAYMA

Oğuzhan Koç

Bilgisayar Mühendisliği
Kocaeli Üniversitesi
oguzhankoc9855@hotmail.com

Yusuf Arslan

Bilgisayar Mühendisliği
Kocaeli Üniversitesi
ysf.arslan2017@gmail.com

Problem Tanımı

Bize verilen metin belgesi içerisindeki kelimeleri dizi kullanmadan kelimelerin metin içindeki tekrar sayısı değerlerinin büyüklüklerine göre sıralanması ve anlık olarak oluşturulan bağlı listeye eklenmesi istenmektedir.

Yapılan araştırmalar

-Projede kullanmamız gereken konu ve yöntemlere karar verdik.

-Bağlantılı liste hakkında araştırmalar yaptık.

-Verilen dosyadaki metini dizi kullanmadan almakta ve kıyaslamada sıkıntı yaşadık.

-Alınan metni stdlib.h kütüphanesindeki malloc fonksiyonu kullanılarak bellekte metnin tutulabileceği bir yer ayırdık. fgets() fonksiyonu kullanılarak metni kopyaladık.

-Alınan metindeki kelimeleri ayırırken sıkıntı yaşadık.

Alınan metni string.h kütüphanesindeki strtok() fonksiyonu kullanılarak kelimelere ve işaretlere ayırdık. Alınan kelimeleri malloc fonksiyonu kullanılarak bellekte tuttuk.

-Alınan kelimelerin metindeki sayılarının bulunmasında sıkıntı yaşadık.

Metindeki kelimelerin adet sayılarını bulurken kullandığımız iç içe yerleştirdiğimiz döngüler kıyaslama ve sayma yaparken gönderilen adresler de karışma oluyordu .

Bu hatalardan kurtulmak için kelimelerin karşılaştırılmasında kullanmak için yeni bir metin belgesi oluşturduk ve metni değiştirerek yeni metin belgesine kaydettik.

-Bağlı listeyi oluştururken eklenen kelimelerin tekrar etmelerini önlemekte sıkıntı yaşadık.

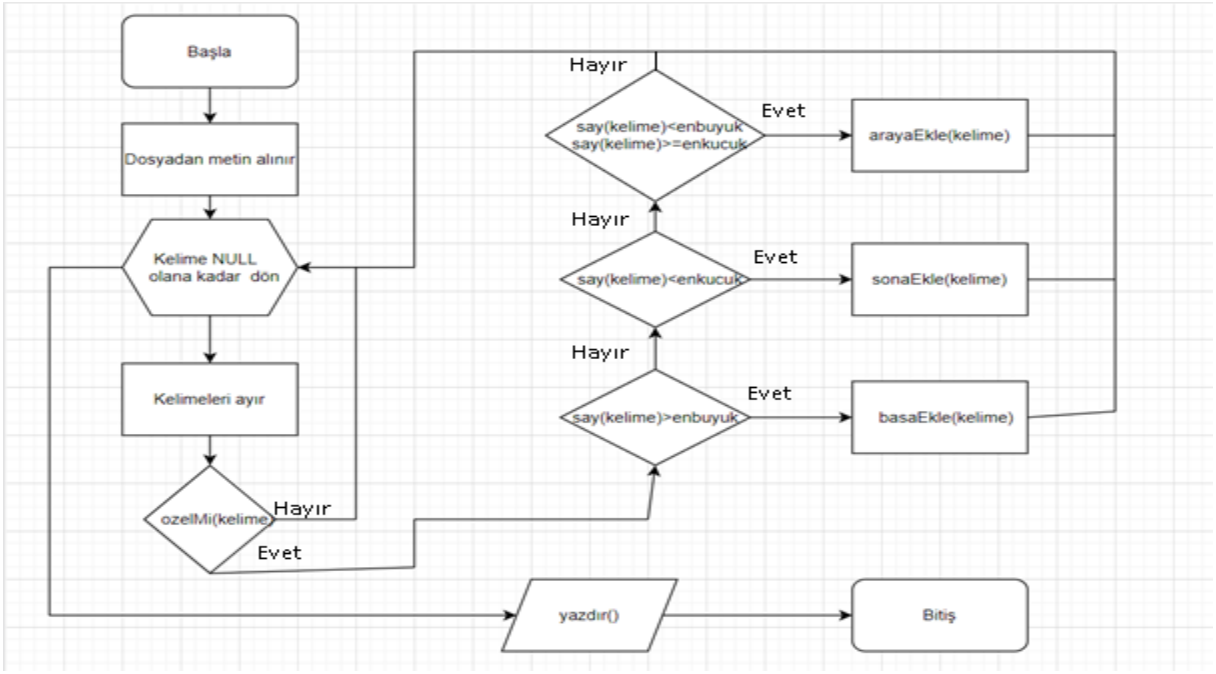
Kelimelerin tekrar etmelerini önlemek için bağlı listeyi dolaşan ve alınan kelimeyle kıyaslayan bir fonksiyon yazdık.

-Bağlı listeyi oluştururken kelimelerin adet sayılarını kıyaslanması ve konumlarının bulunmasında sıkıntı yaşadık.

Kelimelerin tekrar sayıları ve bağlı listede bulunmadıkları kontrol eden bir fonksiyon yazdık.

Tasarım

Akış Şeması



Yazılım Mimarisi

1-Düğüm İşlemleri

Projede yapmak istediğimiz metinde geçen kelimelerin adet sayılarına göre bağlı listedeki düğümlerde sıralamaktır. Bu işlemi yaparken her düğümü temsil eden bir struct kullandık. Bu struct içinde S_kelime adında kelimenin adresinin tutulduğu pointer ,bir sonraki düğümün adresini tutan kendi tipinden olan sonraki adlı pointer ve kelimenin metinde bulunma sıklığını tutan adet isimli integer tipinde değişkenimiz bulunmaktadır.Düğümü gösteren struct aşağıdaki şekildeki gibidir.

```
struct dugum{
    int adet;
    char *S_kelime;
    struct dugum *sonraki;
};
```

2-Fonksiyonlar

void ekle(char *kelime) fonksiyonu:Fonksiyonumuzun başlangıcında alınan pointerın adresinde bulunan kelimenin ozelMi fonksiyonu ile bağlı listede bulunup bulunmadığının kontrolü yapılır.Alınan kelimenin adet sayısının integer olarak tutulan enBuyuk ve enKucuk değerleriyle kıyaslanarak kelimenin basaEkle,sonaEkle veya arayaEkle fonksiyonlarından birine gönderilmesi sağlanarak bağlı listeye eklenir.

```
void ekle(char *kelime1){
    if(ozelMi(kelime1) && say(kelime1)>=1){
            if( say(kelime1)>enBuyuk){
                basaEkle(kelime1);
                enBuyuk=say(kelime1);
                if(enKucuk==0){
                    enKucuk=say(kelime1);}
            }else if( say(kelime1)<=enKucuk){
                sonaEkle(kelime1);
                enKucuk=say(kelime1);
            }else if(say(kelime1)>enKucuk &&  say(kelime1)<=enBuyuk){
                arayaEkle(kelime1);}}
}
```

int say(char * deneme1) fonksiyonu: Fonksiyonumuzun başında metin belgesinden okuma yapılır. Kullandığımız malloc fonksiyonu ile bellekten yer ayırılır. Ayırdığımız alana metin belgesinde bulunan metini kopyalanır. Say fonksiyonuna gönderilen pointer ile metin belgesindeki ayrılan kelimeler while döngüsü içinde karşılaştırılarak sayaç yardımıyla kelimenin metin içinde kaç adet olduğu bulunur ve return edilir.

```
int say(char *deneme1){
FILE *f;
f=fopen("Deneme.txt", "r");
char *charPointer =(char *)malloc(sizeof(char)*1000);
int sayi=0;
while (fgets(charPointer,1000,f)!=NULL) {
    char *kelime=NULL;
    kelime = strtok (charPointer," ");
    while (kelime != NULL){
        if(strcmp(deneme1,kelime)==0){
            sayi++;}
        kelime = strtok (NULL, " ");
    }}
return sayi;
}
```

int ozelMi(char *kelime) fonksiyonu: Fonksiyonumuzda bağlı listede bulunan düğümlerdeki S_kelime ile fonksiyona gönderilen kelime pointerının tuttukları adreslerdeki kelimeler karşılaştırılır.Kelimeler eşitse 0 değeri return edilir,değilse 1 değeri return edilir.Böylece aynı kelimenin tekrarından kaçınılmış olunur.

```
int ozelMi(char *kelime){
q=baslangic;
if(q==NULL){
    return 1;}
if(strcmp(q->S_kelime,kelime)==0){
        return 0;}
if(q->sonraki==NULL){
        return 1;}
if(strcmp(q->sonraki->S_kelime,kelime)==0){
        return 0;}
while(q->sonraki!=NULL ){
    if(strcmp(q->S_kelime,kelime)==0){
        return 0;}
        q=q->sonraki;}
return 1;
}
```

void basaEkle(char *kelime) fonksiyonu: Metinden aldığımız kelimenin adet sayısı bağlı listenin başlangıcındaki düğümde bulunan adet sayısından büyük ise kelime bağlı listenin başlangıç konumuna yerleştirilmesi için başaEkle fonksiyonuna gönderilir.Fonksiyonumuzda yeni bir düğüm oluşturulur ve bağlı listenin başlangıç konumuna yerleştirilir.

```
void basaEkle( char *kelime){
struct dugum *basa_eklenecek_dugum=(struct dugum*)malloc(sizeof(struct dugum));
basa_eklenecek_dugum->S_kelime=(char *)malloc(sizeof(char)*20);
for(int i=0;*(kelime+i)!='\0';i++){
    if(*(kelime+i)!='\n' && *(kelime+i)!=' '){
        *(basa_eklenecek_dugum->S_kelime+i)=*(kelime+i);}
    *(basa_eklenecek_dugum->S_kelime+i+1)='\0';
    }
basa_eklenecek_dugum->adet=say(kelime);
basa_eklenecek_dugum->sonraki=baslangic;
baslangic=basa_eklenecek_dugum;
}
```

void arayaEkle(char *kelime) fonksiyonu: Metinden aldığımız kelimenin adet sayısı bağlı listenin başlangıcındaki düğümde bulunan adet sayısından küçük ise ve integer olarak tutulan enKucuk değerinden büyük ise kelimenin adresi fonksiyona gönderilir.Fonksiyonumuzda yeni bir düğüm oluşturulur ve düğümün yerinin bulunması için bağlı listede karşılaştırılma yapılır.Bulunan konuma düğüm yerleştirilir.

```
void arayaEkle(char *kelime){
struct dugum* araya_eklenecek_dugum=(struct dugum*)malloc(sizeof(struct dugum));
araya_eklenecek_dugum->S_kelime=(char *)malloc(sizeof(char)*20);
for(int i=0;*(kelime+i)!='\0';i++){
    if(*(kelime+i)!='\n' && *(kelime+i)!=' '){
        *(araya_eklenecek_dugum->S_kelime+i)=*(kelime+i);}
    *(araya_eklenecek_dugum->S_kelime+i+1)='\0';}
    araya_eklenecek_dugum->adet=say(kelime);
    q=baslangic;
    while(q->sonraki->adet>=say(kelime)){
        q=q->sonraki;}
    struct dugum* onune=(struct dugum*)malloc(sizeof(struct dugum));
    onune=q->sonraki;
    q->sonraki=araya_eklenecek_dugum;
    araya_eklenecek_dugum->sonraki=onune;
}
```

void sonaEkle(char *kelime) fonksiyonu: Metinden aldığımız kelimenin adet sayısı integer olarak tutulan enKucuk değerinden küçük ise kelime bağlı listenin bitiş konumuna yerleştirilmesi için sonaEkle fonksiyonuna gönderilir.Fonksiyonumuzda yeni bir düğüm oluşturulur ve bağlı listenin bitiş konumuna yerleştirilir.

```
void sonaEkle(char *kelime){
struct dugum* sona_eklenecek_dugum = (struct dugum*)malloc(sizeof(struct dugum));
sona_eklenecek_dugum->S_kelime=(char *)malloc(sizeof(char)*20);
for(int i=0;*(kelime+i)!='\0';i++){
    if(*(kelime+i)!='\n' && *(kelime+i)!=' '){
        *(sona_eklenecek_dugum->S_kelime+i)=*(kelime+i);}
    *(sona_eklenecek_dugum->S_kelime+i+1)='\0';}
sona_eklenecek_dugum->adet=say(kelime);
sona_eklenecek_dugum->sonraki=NULL;
if(baslangic==NULL){
    baslangic=sona_eklenecek_dugum;}
else{
    q=baslangic;
while(q->sonraki!=NULL){
    q=q->sonraki;}
q->sonraki=sona_eklenecek_dugum;}
}
```

void yazdir() fonksiyonu: Fonksiyonumuzda bağlı listede bulunan düğümlerin elemanları olan adet ve S_kelime sırasıyla yazdırılır.

```
void yazdir(){
    q=baslangic;
    int i=1;
    while(q->sonraki!=NULL){
        printf("%d-)%s => %d\n ",i,q->S_kelime,q->adet);
        q=q->sonraki;
        i++;}
    printf("%d-)%s => %d\n ",i,q->S_kelime,q->adet);
}
```

Genel Yapı

Projemizde başlangıç olarak metin dosyası içinde bulunan metin alınır. Metin malloc fonksiyonu ile bellekte ayırdığımız alan içerisine yerleştirilir. Sonrasında metnin içerisinde düzenlemeler yapılır. Oluşturduğumuz yeni metin belgesine bu metin yerleştirilir ve yeni oluşan metin bellekteki ayırdığımız alana atılır. Sonrasında metindeki kelimeler teker teker ayrılarak ekle fonksiyonuna gönderilir. İlk olarak kelimenin tekrar sayısına ve bağlı listede bulunma durumuna bakılır. Bu kıyaslamalar yapıldıktan sonra kelimenin adet sayısına göre basaEkle , sonaEkle ve arayaEkle fonksiyonları ile kelimeler bağlı listeye eklenir. Ekleme işlemi bittikten sonra yazdır fonksiyonu yardımıyla kelimeler ekrana yazdırılır.

KAYNAKÇA

https://www.youtube.com/watch?v=lTFmKiyo0nU&list=PLUUSh8du1azZEsdngPxMQGLnatId0V4h9

http://edestek2.kocaeli.edu.tr/course/view.php?id=50

https://stackoverflow.com/

https://github.com/

https://www.geeksforgeeks.org/