Algoritma Analizi Dersi

Ömer Buğrahan Çalışkan - 17011076

Ödev Konusu: Bir kelimenin geçtiği dokümanları listeleyen bir sistem tasarlanacaktır. Bir kelimenin hangi dokümanlarda geçtiğini bulmak için her seferinde bütün dokümanlara bakmak çok zaman alıcıdır. Bunun yerine bu ödevde, yeni gelen bir dokümandaki kelimeler hashing ile bir sözlüğe yerleştirilecek ve bir kelime arandığında yine hashing yöntemi ile sözlükte aranarak içinde yer aldığı dokümanlar bulunacaktır.

Kod Açıklaması

Sozluk Structure: Kelime bilgileri structure yapısında tutuluyor. Key horner metodu sonucu oluşan değeri tutuyor, value kelimenin kendisini, dokuman ise hangi dokümanlarda geçtiğini tutuyor.

FindKey: Fonksiyon okunan kelimeyi, bir adet secim değişkeni ve hash tablosunu alıyor. Strlwr fonksiyonu ile eger kelimede buyuk harf var ise onu küçültüyoruz. Eğer seçimimiz 0 ise tabloya eleman ekleme yapıyoruzdur ve tabloda zaten var olan bir kelime ise bu işlemi yapmamak için return -1 ile çıkış yapılacak. Eğer yeni bir kelime yapılıyor ise key değeri uygun formülle hesaplanıyor.

InsertToHash: Fonksiyon structure ve hash tablosunu alıyor. h1 ve h2 değerleri formüle uygun bir şekilde hesaplanıyor en son da adresimiz bu iki değişkene bağlı olarak yapılıyor. Bu işlem Double Hashing. Eğer bulunan adres dolu ise ve i değeri m değerinden küçük ise döngümüz devam edecek ve i değeri artacağı için yeni bir adres bulunacak. Eğer döngüden adres değeri -1 yani boş olduğu için çıkıldıysa adrese bilgiler yerleştirilecek ve adres değeri döndürülecek eğer i, m değerini geçtiyse tablo doludur.

SearchToHash: Fonksiyon structure, aranan kelime ve hash tablosunu alıyor. Aranan kelimenin öncelikle key adresi bulunduğundan double hashing yöntemiyle tablodaki yerine bakılıp orada olup olmadığı kontrol edilir.

Main Fonksiyonu

Hash Tablosunu Okuma: Program çalıştırıldığında Hash Tablosunun yazıldığı sozluk.txt dosyasından okuma gerçekleştiriliyor bu okuma fscanf kullanılarak kelime kelime okuma yapıyor. Kelime alındığında keyi bulunuyor ardından kelimenin kendisi ve double hashing yöntemiyle adresi bulunuyor. Ardından bu kelimeden hemen sonra dokumanı yazılı olacağı için tekrar fscanf ile dokumanı okunuyor ve structure da dokuman değişkenine yazdırılıyor. Eğer dokumanda loadfactor kelimesi geçmiş ise dosyadan bu değer okunuyor ve loadfactor değerimiz güncelleniyor.

Okuma işlemi bittikten sonra Loadfactor değeri yazdırılıyor ve program bize dokuman ekleme veya kelime arama işlemlerinden hangisinin yapılacağı soruluyor.

Dokuman Ekleme: Dokuman dosyasının adı girilerek açılıyor. Her kelime fscanf fonksiyonuyla teker teker alınıyor. Eğer tabloda zaten olan bir değerle karşılaşılırsa key bulma, adres bulma, dokuman adı ekleme işlemlerinin hiçbiri yapılmıyor ve eğer aynı dokumandan okunmuyor ise yeni dokumanın adı tabloda zaten olan kelimenin dokuman bilgisine ekleniyor. Yeni kelime eklendiyse bu kelimenin keyi, değeri ve dokuman adı bulunup structure yapısına alınıyor. InsertToHash fonksiyonuyla da tabloya ekleniyor. Loadfactor değerimiz de her eklenen yeni kelimede 1 arttırılıyor. Bu döngünün içerisinde 2 tane kontrol var bir tanesi loadfactor değerini 0.8 ile diğeri de 1 ile karşılaştırıyor. Loadfactor değeri

0.8 değerini geçtiği an ekrana uyarı bastırılıyor, 1'e ulaştığında ise program ekleme yapmayı durduruyor ve yazdirilamayan kelimeler ekrana bastırılıyor.

Tabloya değerler alındıktan sonra sozluk.txt dosyamıza bu hash tablosu yazılıyor. Son olarak da loadfactor değerimiz dosyamıza ekleniyor.

Kelime Arama: Aranacak kelimenin keyi ve değeri structure yapısına ekleniyor ardından SearchToHash fonksiyonuna gönderilerek double hashing adresiyle hash tablosunda olup olmadığına bakılıyor.

Karmaşıklık Analizi

1) **Yeni Kelimenin Double Hashing yöntemiyle tabloya eklenmesi.**

   Best Case = O(1)

   Average Case = O(1)

   Worst Case = O(n)

2) **Double Hashing yöntemiyle kelime bulunması.**

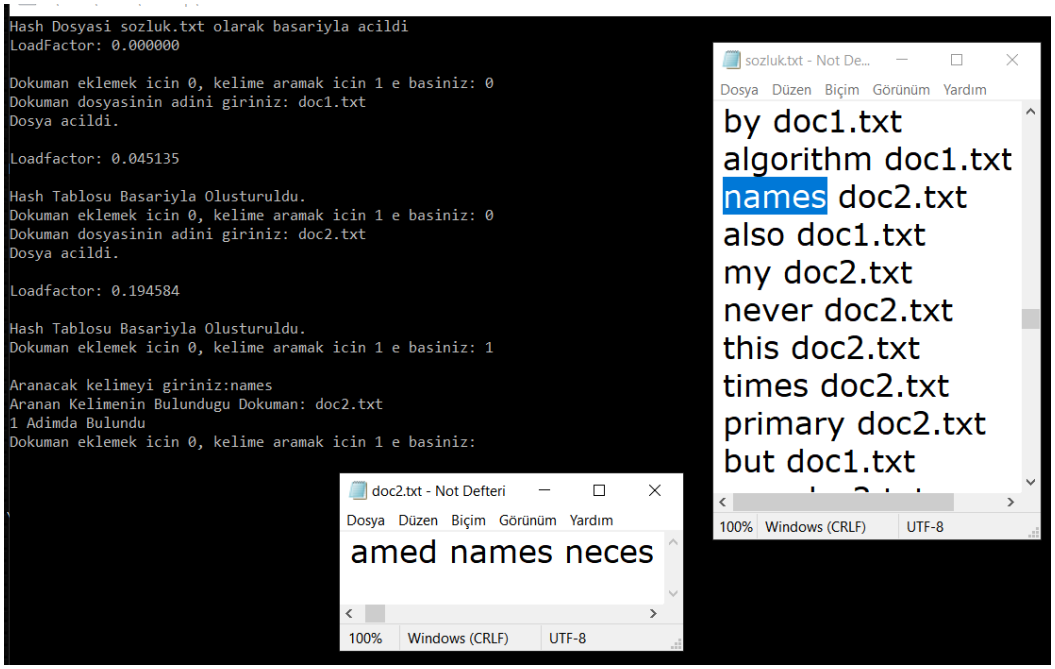
   Best Case = O(1)

   Average Case = O(1)

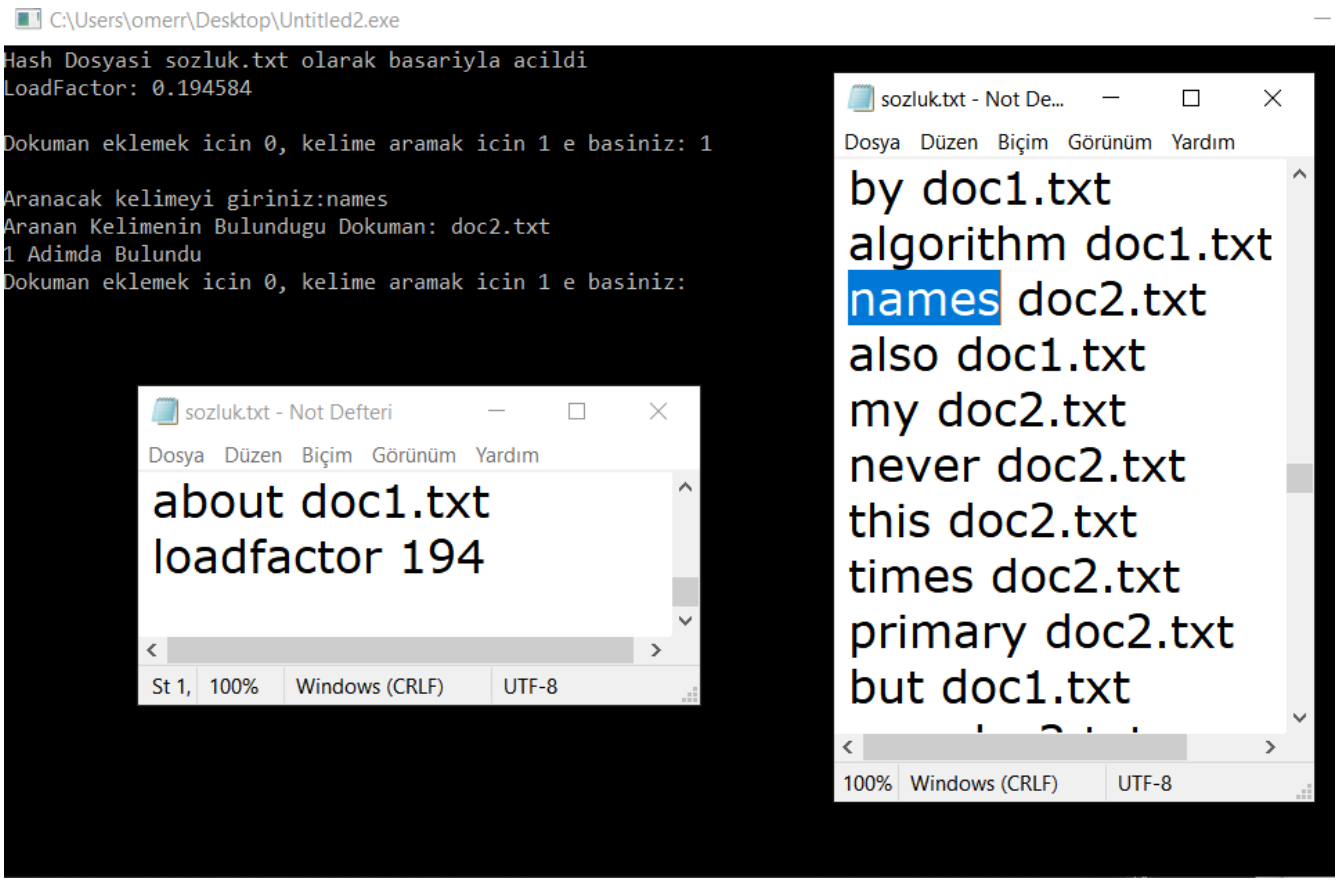
   Worst Case = O(n)

Ekran Görüntüleri

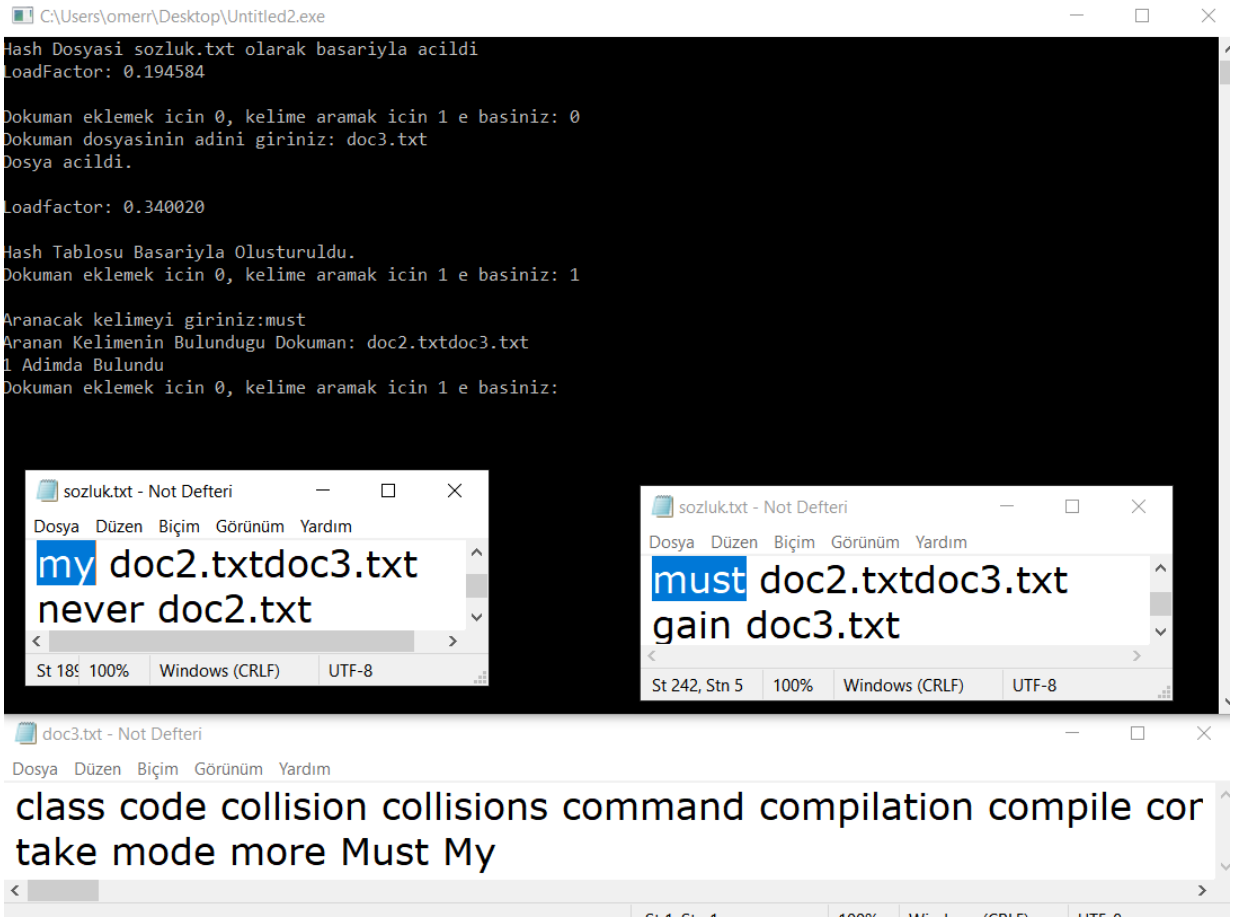
Aşağıdaki örnek sozluk.txt dosyamız boş olarak açılıp 2 farklı dokuman eklendiğinde arama işleminin doğru olarak yapıldığı ve loadfactor değerinin hesaplandığını göstermektedir.



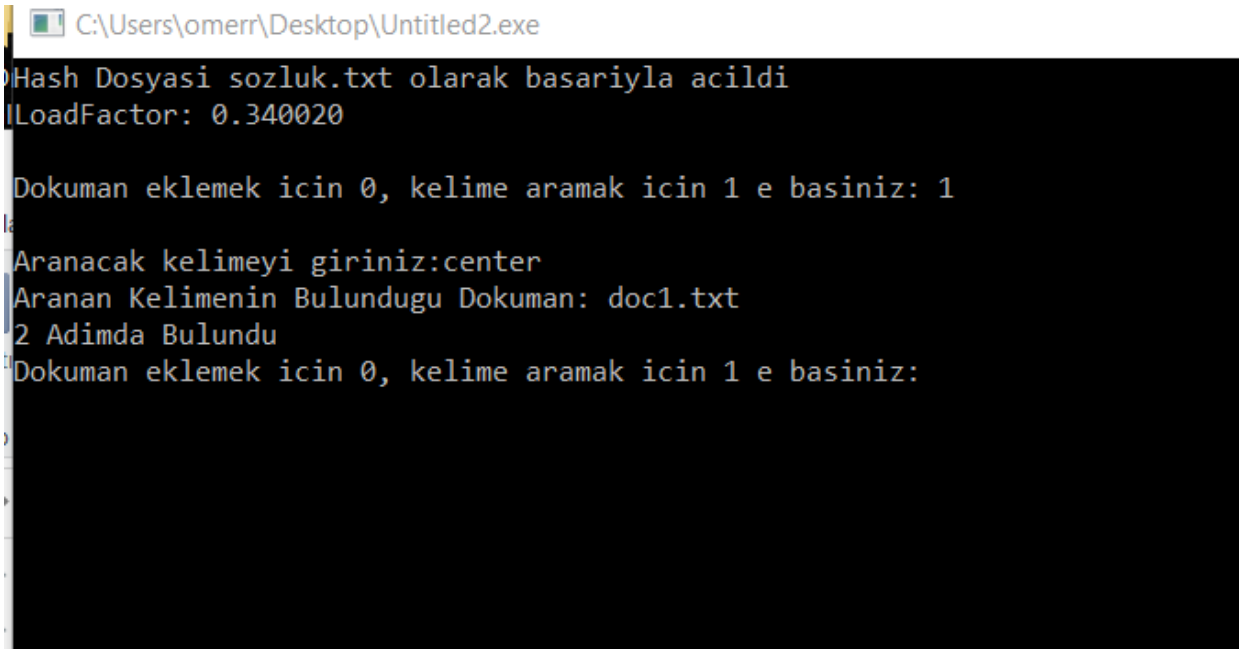
Aşağıdaki örnek sozluk.txt dosyasından tablonun başarıyla alındığını, loadfactor değerinin dosyadan başarıyla okunduğunu ve doğru arama yapabildiğini göstermektedir.



Aşağıdaki örnek yukarıdaki örneklerden kalan sözlük dosyasının tekrar açılması ardından doc3.txt dosyasının eklenmesi. Fakat burada doc3.txt dosyasında my ve must kelimelerinin ilk harfleri büyük olarak verilmiş sozluk dosyasında zaten bu kelimeler olduğundan sadece yeni dokuman dosyasının adı eklendi.(Aynı kelimelere dokuman eklemesi yapılırken loadfactor değeri artmamaktadır.)



2 adımda bulunan kelime örneği



Loadfactor Degerinin 0.8 ve 1 değerine ulaştığı ekran çıktısı. (Burada Hash Tablosu Basarıyla oluşturuldu yazısı yazdırılamayan kelimeler dısındaki kelimelerin eklendigini ifade etmektedir yazdirilamayan kelimeler tabloda arandıgında bulunmamaktadir.)

```
Hash Dosyasi sozluk.txt olarak basariyla acildi
LoadFactor: 0.797392

Dokuman eklemek icin 0, kelime aramak icin 1 e basiniz: 0
Dokuman dosyasinin adini giriniz: doc4.txt
Dosya acildi.

--- Uyari: LoadFactor 0.8 esigini gecti ---
--- Uyari: LoadFactor 1 oldu Ekleme Durduruldu ---

Kelime Yazdirilamadi: marvelous - Loadfactor=1
Kelime Yazdirilamadi: reward - Loadfactor=1
Kelime Yazdirilamadi: zinc - Loadfactor=1
Kelime Yazdirilamadi: standing - Loadfactor=1
Kelime Yazdirilamadi: puzzled - Loadfactor=1
Kelime Yazdirilamadi: nut - Loadfactor=1
Kelime Yazdirilamadi: successful - Loadfactor=1
Kelime Yazdirilamadi: sincere - Loadfactor=1
Kelime Yazdirilamadi: scatter - Loadfactor=1
Kelime Yazdirilamadi: entertain - Loadfactor=1
Kelime Yazdirilamadi: egg - Loadfactor=1
Loadfactor: 1.000000
Hash Tablosu Basariyla Olusturuldu.

Dokuman eklemek icin 0, kelime aramak icin 1 e basiniz: 1

Aranacak kelimeyi giriniz:sincere
Kelime Bulunamadi

Dokuman eklemek icin 0, kelime aramak icin 1 e basiniz:
```

C Kodu

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <math.h>
#define m 997

typedef struct sozluk{ // İcinde kelimenin degeri, keyi ve icinde oldugu dokuman bilgisini tutan structure
   int key;
   char value[50];
   char dokuman[100];
}v;

int FindKey(char buffer[50], int secim,v table[m]){ // Horner metodu kullanılarak kelime degerinin key hesaplanması
   int word_length = strlen(buffer);        // kelimenin uzunluğu
   int key=0,i=0;

   strlwr(buffer);

      if(secim==0){                    // Ekleme yapılırken aynı değer var ise -1 değeri dondurulur.
          for(i=0;i<m;i++){
          if(strcmp(buffer,table[i].value)==0){
                    return -1;
                                       }
                              }
                    }
      for(i=0;i<word_length;i++)             // Horner Metodu ile key hesaplanması
          key += buffer[i]*pow(3,i);
```

```
   return key;
}

int InsertToHash(v data,v table[m]){      // Double hashing kullanılarak key değerinin hash tablosuna yerlestirilmesi
   int i=0;

   int h1 = data.key % m;
   int h2 = 1 + (data.key%(m-1));
   int a = (h1 + i*h2) % m;

   while((table[a].key!=-1) && (i < m)){
                  i++;
      h1 = data.key % m;
                  h2 = 1 + (data.key%(m-1));
         a = (h1 + i*h2) % m;
   }
         if(table[a].key == -1){    // Adresteki yer boş ise yerleştirilme işlemi
                  table[a].key=data.key;
         strcpy(table[a].value,data.value);
         return a;
                  }
                  printf("Kelime Eklenemedi: %s\n",data.value);
                  return -1;
}

void SearchToHash(v data, char aranan[50],v table[m]){
   int i=0, flag=0;
                                                      // Aranan kelimenin hashing algoritması ile bulunması
   int h1 = data.key % m;
         int h2 = 1 + (data.key%(m-1));
```

```
   int a = (h1 + i*h2) % m;


          while((i < m) && (flag==0)){                    // Eğer tabloda bulunursa adım sayısı ve dokumanı yazdırılır ardından dongu sonlandırılır bulunamadı ise i arttırılarak devam edilir.
                    if(strcmp(table[a].value,data.value)==0){
      printf("Aranan Kelimenin Bulundugu Dokuman: ");
      puts(table[a].dokuman);
      printf("%d Adimda Bulundu\n", i+1);
      flag=1;
      break;
          }


                    i++;
      h1 = data.key % m;
                    h2 = 1 + (data.key%(m-1));
          a = (h1 + i*h2) % m;
   }
          if(flag==0)
      printf("Kelime Bulunamadi\n");
}


int main()
{
          v table[m];                              // hashtable
          int loadfactor=0;           // loadfactor baslanhngi0pp*0c degeri
   int i, secim=0,adr;      // yapılacak islem icin secim, donguler icin i ve double hashing isleminden sonra donecek olan adres
          char dosya[50];             // okunacak dosya
          int flag=1,flag2=1;         // loadfactor kontrolleri icin gerekli degiskenler


          FILE* sozluk;
   sozluk = fopen("17011076.txt","r");
```

```
    for(i=0;i<m;i++){
       table[i].key=-1;
       }
    if (sozluk == NULL){
       printf("Dosya acilamadi.");
          return 0;
          }
                else{
       printf("Hash Dosyasi 17011076.txt olarak basariyla acildi\n");
       char buffer[50];
       while (fscanf(sozluk, "%s", buffer) > 0){    // hash dosyasının kelime kelime okunması - sozluk.txt


         if(strcmp(buffer,"loadfactor")==0){                  // Eger sozluk.txt dosyasında loadfactor degeri varsa okunacak
                                              fscanf(sozluk, "%d", &loadfactor);
                                              continue;
                                    }
          v data;                   // Kelimenin structure a keyi ile beraber eklenmesi
          data.key=FindKey(buffer,0,table);
          strcpy(data.value,buffer);
          adr=InsertToHash(data,table);
          char temp[50];
                                    fscanf(sozluk, "%s", temp);                  // Dokuman bilgisinin okunması
          strcpy(table[adr].dokuman,temp);
             }
          }
  fclose(sozluk);


  printf("LoadFactor: %f\n",(float)loadfactor/m);
  while(secim==0 || secim==1){
```

```
        printf("\nDokuman eklemek icin 0, kelime aramak icin 1 e basiniz: ");
        scanf("%d",&secim);

        if(secim==0){
        printf("Dokuman dosyasinin adini giriniz: ");
        scanf("%s",dosya);

        FILE* dokuman;
        dokuman = fopen(dosya, "r");
    if (dosya == NULL){
       printf("Dosya acilamadi.");
       return 0;
                        }
              else{
       printf("Dosya acildi.\n\n");
       char buffer2[50];
        while (fscanf(dokuman, "%s", buffer2) > 0 && flag==1){
         v data;
              if(FindKey(buffer2,0,table)!=-1){                    // Dokumandan kelime eklenmesi
eger tabloda zaten var ise bu islemin yapılmasına gerek yok
            data.key=FindKey(buffer2,0,table);
                    strcpy(data.value,buffer2);
                    int a = InsertToHash(data,table);
                    strcpy(table[a].dokuman,dosya);
                    if(a!=-1)
                                        loadfactor++;
                                }
                                else{              // Zaten var olan bir kelime ekleniyor ise
dokuman bilgisinin alınması
                                    strlwr(buffer2);
                                    for(i=0;i<m;i++){
```

```
                                                if(strcmp(table[i].value,buffer2)==0 &&
strcmp(table[i].dokuman,dosya)!=0){
                                                        strcat(table[i].dokuman,dosya);
                                                                }
                                                        }
                                                }
                                        if(((float)loadfactor/m) > 0.8 && flag2==1){
                                                printf("--- Uyari: LoadFactor 0.8 esigini gecti --
-\n");
                                                flag2=0;
                                                }
                                        else if((float)loadfactor/m==1){
                                                printf("--- Uyari: LoadFactor 1 oldu Ekleme
Durduruldu ---\n\n");
                                                flag=0;
                                                }
                        }

        if(flag==0){
                while (fscanf(dokuman, "%s", buffer2) > 0){
                        printf("Kelime Yazdirilamadi: %s - Loadfactor=1\n",buffer2);
                                                }
                                }
                                        }
        fclose(dokuman);

        FILE *fp=fopen("17011076.txt","w");             // Hash tablosuna alınan degerlerin
17011076.txt dosyasına yazılması
          for(i=0;i<m;i++){
                if(table[i].key!=-1){
                        fprintf(fp,"%s %s\n",table[i].value,table[i].dokuman);
                                                }
```

```
                }

                        fprintf(fp,"%s %d","loadfactor",loadfactor);       // Loadfactor 17011076.txt
dosyasına yazdırılır.

        fclose(fp);

        printf("Loadfactor: %f\n",(float)loadfactor/m);    // Loadfactor degerinin m degerine bolunerek
yazdırılması int deger oldugu icin basina float koyduk.


         printf("Hash Tablosu Basariyla Olusturuldu.\n");

}

         if(secim==1){

                char aranan[50];                        // Aranan kelimenin key degerinin ve
kendisinin tabloda aramak uzerine structure yapısına eklenmesi

        printf("\nAranacak kelimeyi giriniz:");

        scanf("%s",aranan);

        v data;

        data.key=FindKey(aranan,1,table);

        strcpy(data.value,aranan);

        SearchToHash(data,aranan,table);

                                }

                        }

                return 0;

                }
```