2020-2021 Güz Yarıyılı
Algoritma Analizi
Ödev – 2

Ders Yürütücüleri
Doç. Dr. M. Elif KARSLIGİL
Dr. Öğr. Üyesi M. Amaç GÜVENSAN

Burak Boz

18011706

Konu : Hashing Algoritmasının Kullanımı

Problem : Bu ödevde, 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.

Sistem 3 alt bölümden oluşacaktır :

1. Yeni gelen dokümanı sözlüğe ekleme : Bir doküman geldiğinde dokümanda bulunan kelimeleri hash fonksiyonundan geçirerek eğer hash tablosunda yoksa hem kelimeyi hem de dokümanın adını tabloya ekleyin, kelime tabloda varsa sadece dokümanın adını bu kelimenin bulunduğu adrese ekleyin. Hash tablosunu bir dosyada saklayın. Program yeniden çalıştırıldığında mevcut dosyayı kullanarak, yeni bilgileri ekleyin.

2. Sorgulanan kelimeyi arama : Bir kelime sorgulandığında hash fonksiyonundan geçirerek kelimeyi tabloda arayın. Eğer tabloda varsa geçtiği dokümanların isimlerini ekrana yazdırın. Tabloda yoksa, olmadığını mesajla belirtin. Arama işleminin kaç adımda tamamlandığını ekrana yazdırınız.

3. Hash tablosunu oluşturma : Hash tablosu aşağıdaki kurallara göre oluşturulacaktır :

   a. Hash tablosunu oluştururken openaddress, çakışma problemini çözmek için double hashing yöntemleri kullanılacaktır. Buna göre: h(key,i) = [h1(key) + i*h2(key)] mod M h1(key) = key mod M h2(key) = 1 + (key mod MM)

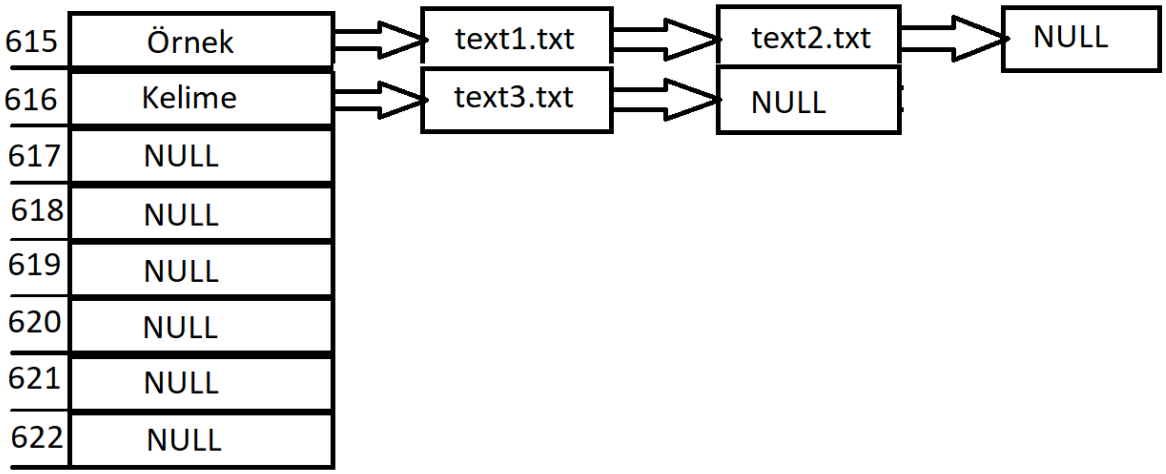
   b. Dosyaların içinde sadece kelimeler olduğu, özel karakterler olmadığı varsayılmıştır. Kelimeler küçük-büyük harf karışık olabilir. Örneğin “Araba” ve “araba” aynı kelimelerdir. Kelimeleri sayıya çevirirken Horner Metodu kullanılacaktır.

   c. Tablo uzunluğunu gösteren M değerini 1000 değerine en yakın ASAL sayı olarak alınız. • h2 fonksiyonunda MM = M -1 alınız. • Loadfactor’ü her yeni eklenen kelime için ekrana yazdırıp LoadFactor 0.8’i geçtiğinde kullanıcıya UYARI MESAJI gösteriniz. LoadFactor 1.0 olduğunda EKLEME işlemini DURDURUNUZ ve dosyada EKLENEMEYEN kelimeleri EKRANA yazdırınız. • İPUCU: LoadFactor’ün son değerini dosyada saklayınız. Yeni eklenen kelimeler için bu değeri GÜNCELLEYİNİZ.

Çözüm: ***.**txt dosyalarından alınan kelimeler, horner metodu ile hashlenerek key haline getirilmiş ve uygun adreslere saklanmıştır. Tablo boyutu olarak 1000'e en yakın asal sayı olan 1009 alınmıştır. Horner metodunda sayılar key değerlerine çevrilirken üs olarak genellikle 31 değeri kullanılmaktadır fakat dilin kısıtlarından dolayı yaptığım testler sonucu (bazı uzun kelimeler keye çevrilirken boyut problemi oluyor. Örneğin "coronavirus" kelimesi üs olarak 7 kullanıldığı zaman çevrilemiyor.) bu üs 2 olarak belirlenmiştir. Oluşturulan keyler Double-Hashing metodu ile 1009 gözlü tabloda uygun yerlere yerleştirilmiştir. Kelimelerin bulunduğu dosya isimleri komşuluk matrisi benzeri bir yöntem ile saklanmıştır.

Tüm kelimeler için struct yapısı kullanılmıştır. Kelimeler 1009 gözlü struct bir dizide saklanmıştır. Kelimelerin bulunduğu adresler başlangıç adresi olarak alınmış ve bulundukları dosya adları yeni bir struct yapı içerisinde linkli liste olarak saklanmıştır.

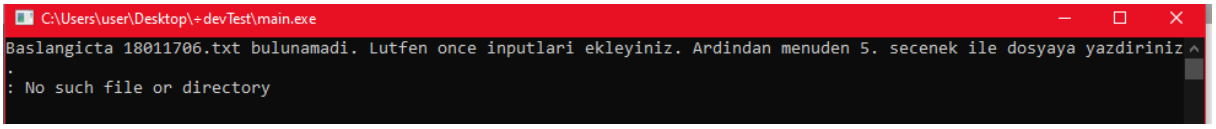
Örnek Yapı:



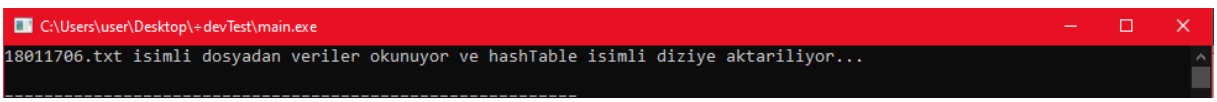
Başlangıç: Program başlar başlamaz 1009 gözlü alan açılır ve bellekteki gözler boşaltılır. Ardından 18011706.txt isimli dosyadan veriler okunarak bu tablodaki yerlerine yerleştirilir. Eğer bu txt dosyası yoksa kullanıcıya veri girişi yapması için uyarı verilir. Dosya var ise yerleştirilen veri miktarı loadFactor değişkeninde saklanır.

Ekran çıktıları:

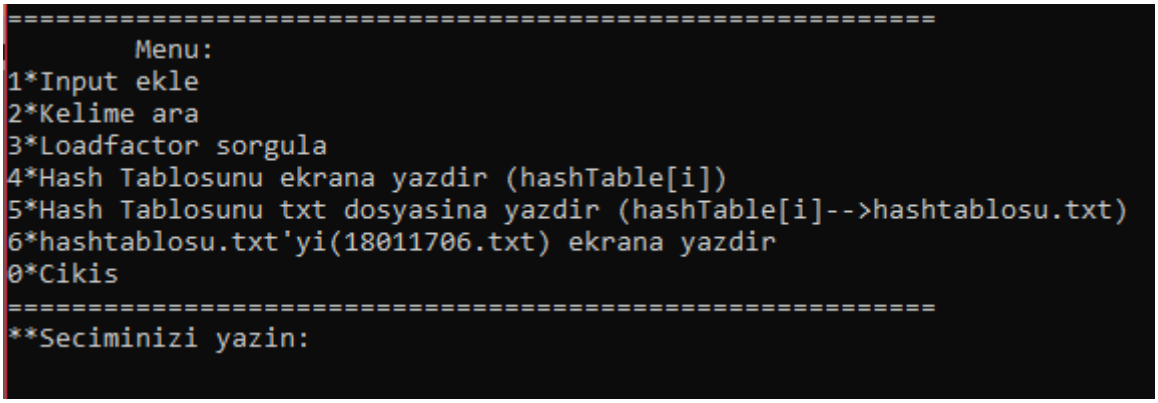
Tablo yoksa:



Tablo varsa:



Menü: Okuma yapıldıktan sonra ekrana menü açılır.



Not: Köşeli parantezler veri girişini temsil etmektedir. Verileri girerken [] kullanmayınız.

1) Yeni input eklemek için kullanılır. Seçim yapıldıktan sonra [dosyaadi.txt] şeklinde giriş yapılır ve kelimeler tablolara yerleştirilir.
2) Kelime aramak için kullanılır. Seçim yapıldıktan sonra [kelime] şeklinde giriş yapılır. Bu kelime horner metoduyla hashlenir ve tabloda bulunur. Ardından hangi dosyalarda geçtiği ekrana yazdırılır.
3) Tablonun ne kadar dolu olduğunu anlamak için bu seçim yapılır.
4) Hash tablosunun tamamını ekrana yazdırmak için kullanılır. Açıklama satırları ile kapanan bölüm açılırsa sadece dolu gözleri ekrana yazacaktır.
5) Hafızada doldurulmuş olan hashTable isimli struct dizisi 18011706.txt isimli text dosyasına yazdırılır.
6) 18011706.txt isimli dosyadan satır satır okuma yapılır ve ekrana yazdırılır.
0) Çıkış yapmak için kullanılır. İlgili Free() fonksiyonları burada tanımlanmıştır.

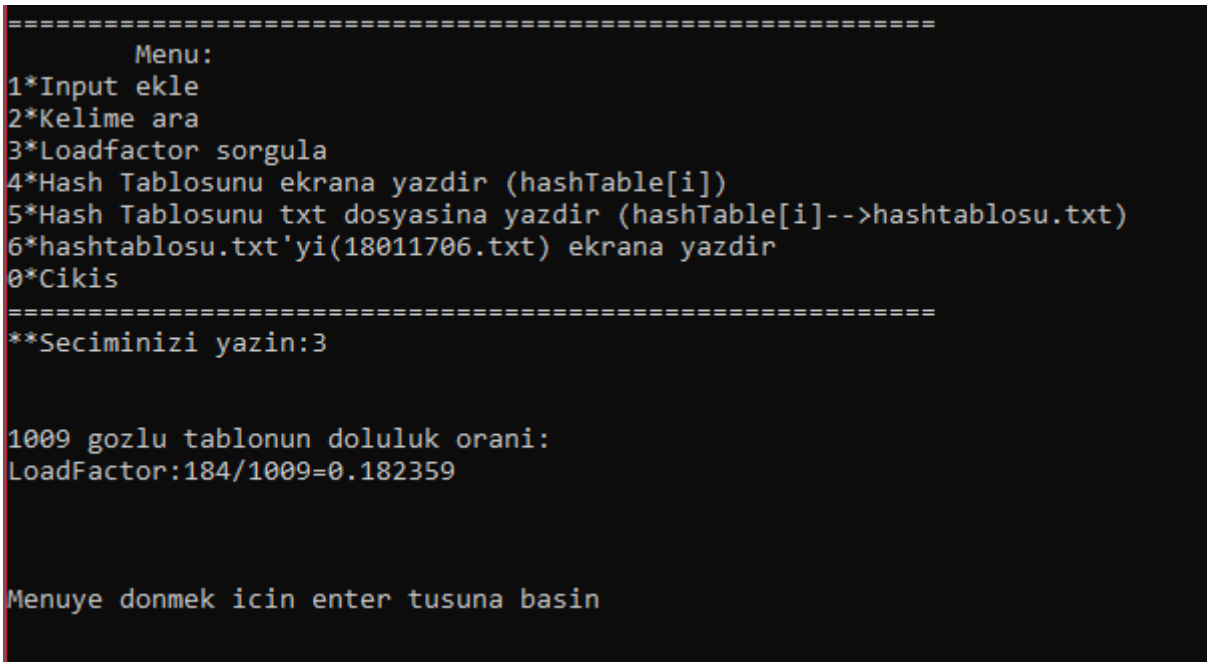
1)Input Ekleme

```
C:\Users\user\Desktop\÷devTest\main.exe
============================================================
        Menu:
1*Input ekle
2*Kelime ara
3*Loadfactor sorgula
4*Hash Tablosunu ekrana yazdir (hashTable[i])
5*Hash Tablosunu txt dosyasina yazdir (hashTable[i]-->hashtablosu.txt)
6*hashtablosu.txt'yi(18011706.txt) ekrana yazdir
0*Cikis
============================================================
**Seciminizi yazin:1

Eklenecek txt dosyasini girin (Ornek:input1.txt):test.txt
LoadFactor: 184 Eklenen dosya adi: test.txt
One - 398
LoadFactor: 185 Eklenen dosya adi: test.txt
advanced - 579
LoadFactor: 186 Eklenen dosya adi: test.txt
diverted - 339
LoadFactor: 187 Eklenen dosya adi: test.txt
domestic - 882
LoadFactor: 188 Eklenen dosya adi: test.txt
sex - 586
LoadFactor: 189 Eklenen dosya adi: test.txt
repeated - 767
LoadFactor: 190 Eklenen dosya adi: test.txt
bringing - 917


Menuye donmek icin enter tusuna basin
```

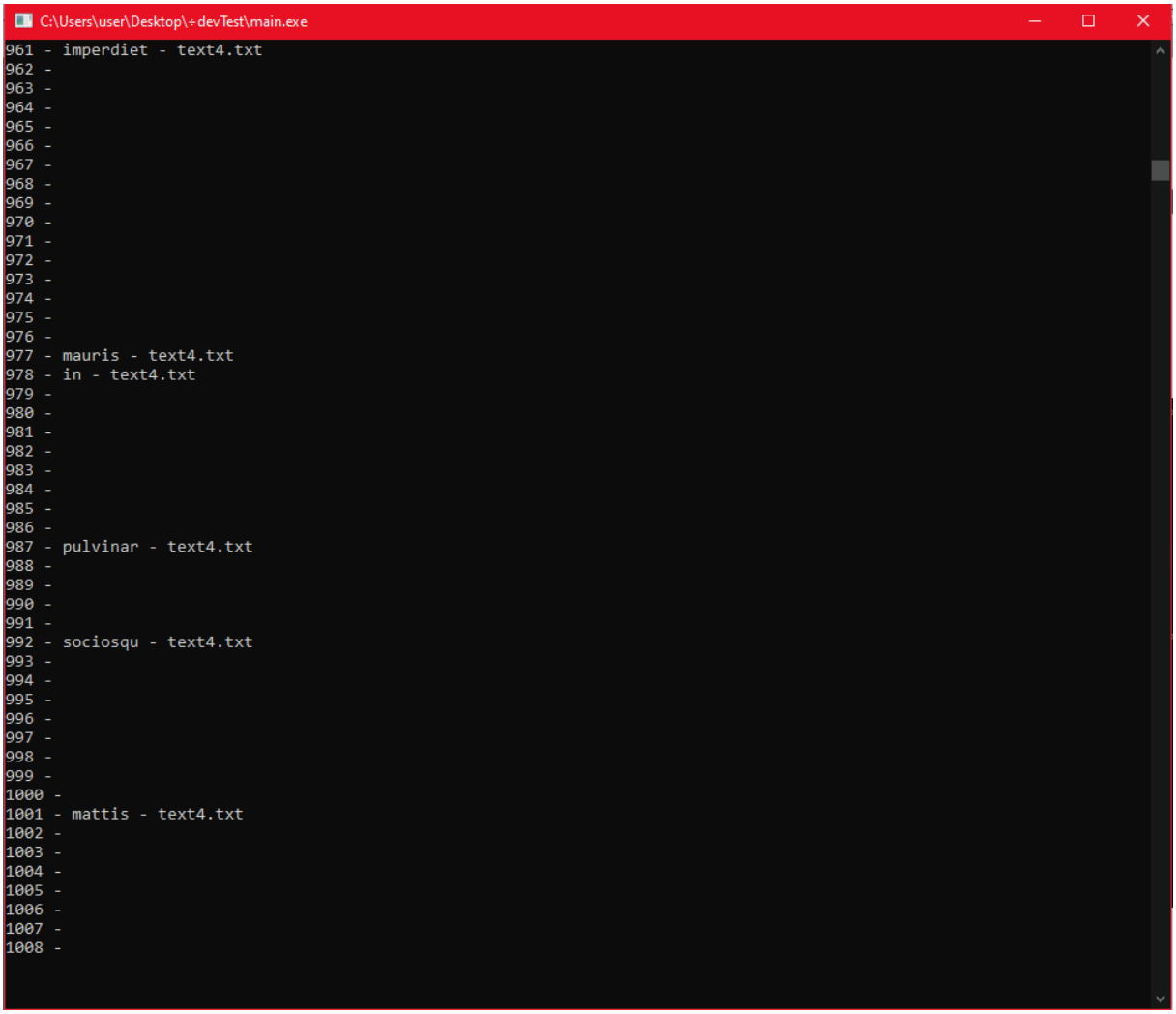
2)Arama

```
============================================================
        Menu:
1*Input ekle
2*Kelime ara
3*Loadfactor sorgula
4*Hash Tablosunu ekrana yazdir (hashTable[i])
5*Hash Tablosunu txt dosyasina yazdir (hashTable[i]-->hashtablosu.txt)
6*hashtablosu.txt'yi(18011706.txt) ekrana yazdir
0*Cikis
============================================================
**Seciminizi yazin:2
Aradiginiz kelimeyi girin:domestic
882aranan hash : 882
Tablodan gelen kelime: domestic
 Bulunduklari dosyalar: test.txt

Menuye donmek icin enter tusuna basin
```

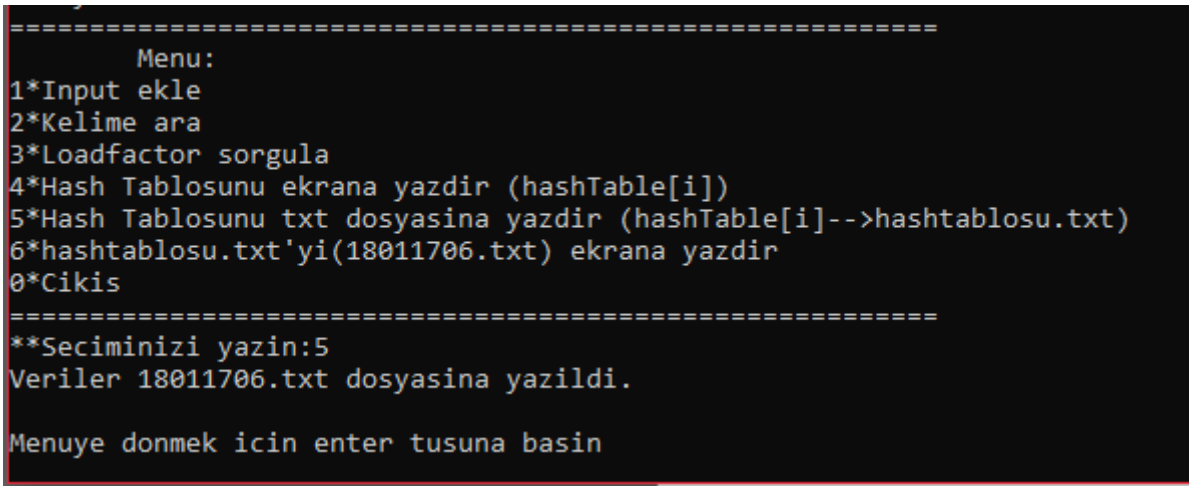
3)LoadFactor sorgulama

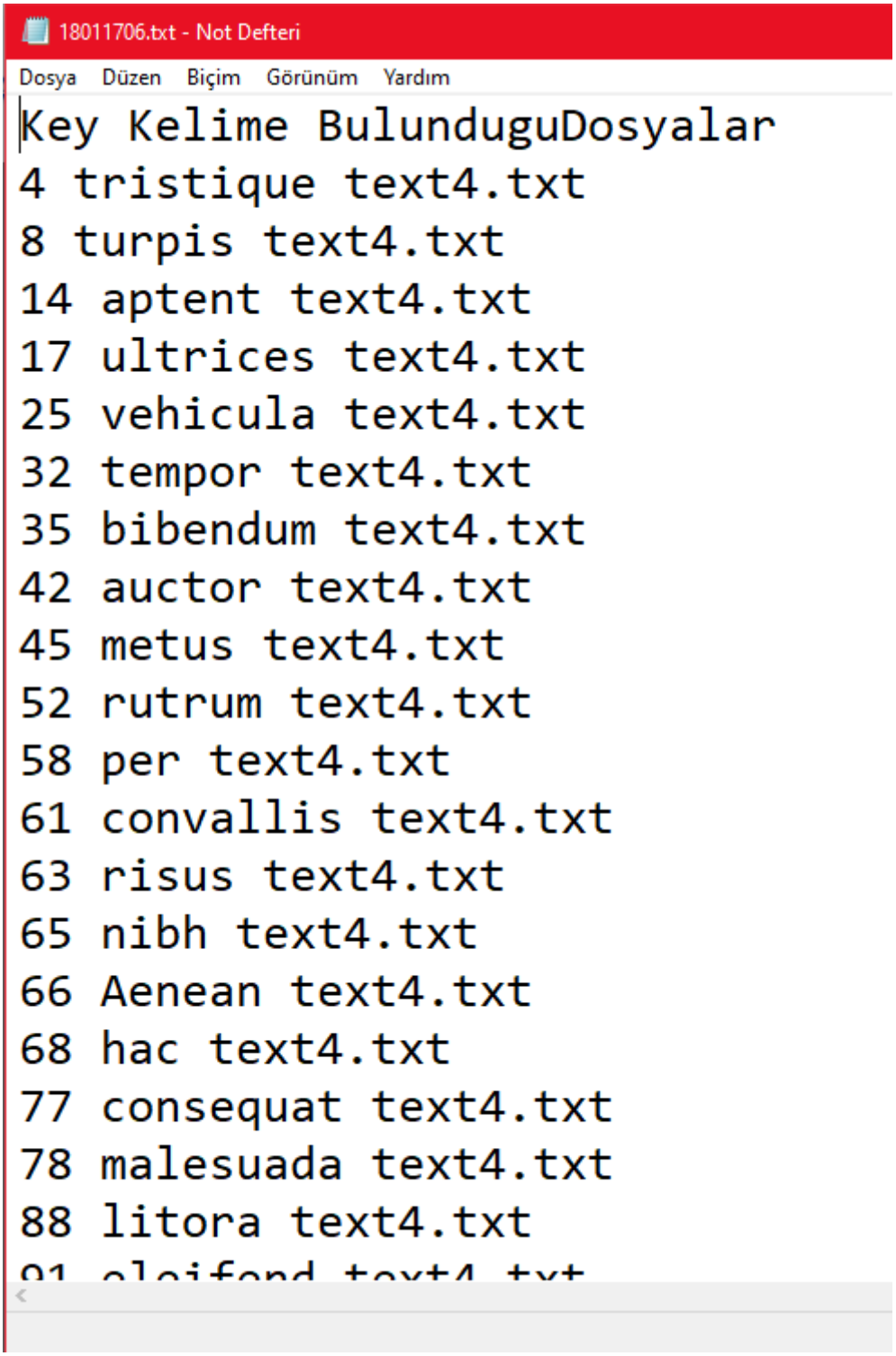


4)Hash tablosunu ekrana yazdırma



5)Hash tablosunu 18011706.txt dosyasına yazdırma




18011706.txt - Not Defteri
Dosya Düzen Biçim Görünüm Yardım
Key Kelime BulunduguDosyalar
4 tristique text4.txt
8 turpis text4.txt
14 aptent text4.txt
17 ultrices text4.txt
25 vehicula text4.txt
32 tempor text4.txt
35 bibendum text4.txt
42 auctor text4.txt
45 metus text4.txt
52 rutrum text4.txt
58 per text4.txt
61 convallis text4.txt
63 risus text4.txt
65 nibh text4.txt
66 Aenean text4.txt
68 hac text4.txt
77 consequat text4.txt
78 malesuada text4.txt
88 litora text4.txt

6)18011706.txt dosyasındaki verileri ekrana yazdırma

```
============================================================
        Menu:
1*Input ekle
2*Kelime ara
3*Loadfactor sorgula
4*Hash Tablosunu ekrana yazdir (hashTable[i])
5*Hash Tablosunu txt dosyasina yazdir (hashTable[i]-->hashtablosu.txt)
6*hashtablosu.txt'yi(18011706.txt) ekrana yazdir
0*Cikis
============================================================
**Seciminizi yazin:6

Hash Tablosu
Key Kelime BulunduguDosyalar
4 tristique text4.txt
8 turpis text4.txt
14 aptent text4.txt
17 ultrices text4.txt
25 vehicula text4.txt
32 tempor text4.txt
35 bibendum text4.txt
42 auctor text4.txt
45 metus text4.txt
52 rutrum text4.txt
58 per text4.txt
61 convallis text4.txt
63 risus text4.txt
65 nibh text4.txt
66 Aenean text4.txt
```

0)Çıkış

```
============================================================
        Menu:
1*Input ekle
2*Kelime ara
3*Loadfactor sorgula
4*Hash Tablosunu ekrana yazdir (hashTable[i])
5*Hash Tablosunu txt dosyasina yazdir (hashTable[i]-->hashtablosu.txt)
6*hashtablosu.txt'yi(18011706.txt) ekrana yazdir
0*Cikis
============================================================
**Seciminizi yazin:0

**Bellek Temizlendi**

--------------------------------
Process exited after 2.634 seconds with return value 0
Press any key to continue . . .
```

Fonksiyonlar ve Değişken tanımlamaları:

Fonksiyonlar

- int kelimeAra(char[],int);//Kelime ararken kullanılacak fonksiyondur. (aranacakKelime,horner metodundan gelen değer)
- int h2(int);//DobuleHashing yaparken kullanılacak fonksiyon
- int h1(int);//DobuleHashing yaparken kullanılacak fonksiyon
- struct Node * newNode();//Yeni düğüm oluşturmak için kullanılacak fonksyion. Bu düğümlerde dosya isimleri saklanacak
- int hash(char[],int,char[]);//Hashing yaparken kullanılacak fonksiyondur. (hashlenecek kelime,horner metodundan gelen değer, bulunduğu dosy adı)
- int getPower(int, int);//Üs almak için kullanılacak fonksyion (Taban,Üs)
- int horner (char[]);//Kelimeye horner metodu uygulanarak üretilecek olan değer (kelime)
- void satirOku(char[]);//Txt dosyasından satır satır okumayı sağlayan fonksiyon (dosya adı)
- void kelimeOku(char[]);//Txt dosyasından kelime kelime okumayı sağlayan foknsiyon (dosya adı)

Struct Yapılar

- struct Node { //Bu düğümlerde dosya adları saklanacaktır
  char dosyaAdi[100];
  struct Node* next;
  };
- typedef struct {//Hash tipinde bir dizi tanımlanacak ve burada kelimeler saklanacaktır.
  char kelime[1000];
  struct Node* head;
  }HASH;

Değişkenler

- int loadFactor; //Doluluk oranı için sayaç olarak kullanılacak. Farklı fonksiyonlarda arttırılacağı için global tanımlanmıştır.
- float dolulukOrani=0.0f;//Doluluk oranı bu değişkende hesaplanacak
- char arananKelime[1000];//Aranan kelime bu değişkende tutulacak

- int arananHash=0;//Aranan kelimenin hash değeri bu değişkende tutulacak
- char hashDosya[100];//Hash dosyasının adı burada saklanacak
- char inputDosya[100];//input dosyasının adı burada saklanacak
- int secim=-1;//Menü için kullanılacaktır.

Tüm Kodlar:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define M 1009
#define MM 1008
//18011706 Burak Boz
/*
Açıklama:
Program çalışır çalışmaz bulunduğu konumda 18011706.txt isimli dosyayı çalıştırır. Bu dosya içerisindeki veriler
1 indexli satırdan itibaren (0. index başlık satırıdır) verileri okuyarak hashTable isimli structlardan oluşan diziye doldurur.
Bu dosyada satırlar;
	key kelime bulunduguDosya1 bulunduguDosya2 ...
şeklinde düzenlenmiştir. Tekrar kaydedileceği zaman yine aynı düzende kayıt edilecektir.
***MENÜ KULLANIMI***
1)İnput eklemek için kullanılır. dosya adını dosyaadi.txt şeklinde yazınız.
2)Kelime aramak için kullanılır. İnputlar eklendikten sonra sadece kelime adını yazarak arama yapınız
3)LoadFactor'ü yazdırmak için kullanılır. Seçimi yapmanız yeterlidir.
4)Tüm hashTable değerlerini ekrana yazdırmak için kullanılır.
5)Elde bulunan hashTable dizisindeki tüm değerleri 18011706.txt isimde ve exe dosyasıyla aynı lokasyonda olacak şekilde dosyaya yazar
6)18011706.txt dosyasının içindeki verileri ekrana yazdırmak için kullanılır.
0)Çıkış yapmak için
*/

int kelimeAra(char[],int);//Kelime ararken kullanılacak fonksiyondur. (aranacakKelime,horner metodundan gelen değer)
int h2(int);//DobuleHashing yaparken kullanılacak fonksiyon
int h1(int);//DobuleHashing yaparken kullanılacak fonksiyon
struct Node * newNode();//Yeni düğüm oluşturmak için kullanılacak fonksyion. Bu düğümlerde dosya isimleri saklanacak
int hash(char[],int,char[]);//Hashing yaparken kullanılacak fonksiyondur. (hashlenecek kelime,horner metodundan gelen değer, bulunduğu
dosy adı)
int getPower(int, int);//Üs almak için kullanılacak fonksyion (Taban,Üs)
int horner (char[]);//Kelimeye horner metodu uygulanarak üretilecek olan değer (kelime)
void satirOku(char[]);//Txt dosyasından satır satır okumayı sağlayan fonksiyon (dosya adı)
void kelimeOku(char[]);//Txt dosyasından kelime kelime okumayı sağlayan foknsiyon (dosya adı)

struct Node { //Bu düğümlerde dosya adları saklanacaktır
	char dosyaAdi[100];
	struct Node* next;
};
typedef struct {//Hash tipinde bir dizi tanımlanacak ve burada kelimeler saklanacaktır.
	char kelime[1000];
	struct Node* head;
}HASH;
HASH *hashTable;
int loadFactor;
int main()
{
	int i=0;
	hashTable = (HASH*)calloc(M, sizeof(HASH));
	loadFactor=0;//Doluluk oranı için sayaç olarak kullanılacak
```

```
float dolulukOrani=0.0f;//Doluluk oranı bu değişkende hesaplanacak
char arananKelime[1000];//Aranan kelime bu değişkende tutulacak
int arananHash=0;//Aranan kelimenin hash değeri bu değişkende tutulacak
char hashDosya[100];//Hash dosyasının adı burada saklanacak
char inputDosya[100];//input dosyasının adı burada saklanacak
int secim=-1;//Menü için kullanılacaktır.
for(i=0;i<M;i++)//Hazırlanan tablo temizleniyor.
{
	hashTable[i].head=NULL;
	memset(hashTable[i].kelime,0,sizeof(hashTable[i].kelime));
}
baslangic("18011706.txt");
do
{
	printf("\n============================================================");
	printf("\n\tMenu:\n");
	printf("1*Input ekle\n");
	printf("2*Kelime ara\n");
	printf("3*Loadfactor sorgula\n");
	printf("4*Hash Tablosunu ekrana yazdir (hashTable[i])\n");
	printf("5*Hash Tablosunu txt dosyasina yazdir (hashTable[i]-->hashtablosu.txt)\n");
	printf("6*hashtablosu.txt'yi(18011706.txt) ekrana yazdir\n");
	printf("0*Cikis\n");
	printf("============================================================\n");
	printf("**Seciminizi yazin:");
	scanf("%d",&secim);

	if(secim==1)//Veri girişi
	{
		printf("\nEklenecek txt dosyasini girin (Ornek:input1.txt):");
		scanf("%s",&inputDosya);
		kelimeOku(inputDosya);

		printf("\n\nMenuye donmek icin enter tusuna basin");
		getch();
	}
	else if(secim==2)//Kelime ara
	{
		printf("Aradiginiz kelimeyi girin:");
		scanf("%s",&arananKelime);
		arananHash=kelimeAra(arananKelime,horner(arananKelime));
		if(arananHash!=-1)
		{
			printf("aranan hash : %d\n",arananHash);
			printf("Tablodan gelen kelime: %s\n ",hashTable[arananHash].kelime);
			printf("Bulunduklari dosyalar: ");
			struct Node *nodeTmp = hashTable[arananHash].head;
			while(nodeTmp != NULL)
			{
				printf("%5s ",nodeTmp->dosyaAdi);
				nodeTmp=nodeTmp->next;
			}
		}
		else{
			printf("Aradiginiz kelime bulunamadi");
		}

		printf("\n\nMenuye donmek icin enter tusuna basin");
		getch();
	}
	else if(secim==3)//LoadFactor'ü yazdırmak için
	{
		dolulukOrani=(float) loadFactor/1009;
		printf("\n\n1009 gozlu tablonun doluluk orani:\n");
		printf("LoadFactor:%d/1009=%2f\n\n",loadFactor,dolulukOrani);
```

```
			printf("\n\nMenuye donmek icin enter tusuna basin");
			getch();
		}
		else if(secim==4)//hashTable içerisindeki tüm değerler ekrana yazdırılır. Aşağıdaki açıklama satırları açılırsa sadece dolu gözleri yazar.
		{
			if(loadFactor>0)
			{
				for(i=0;i<M;i++)
				{
					//if(hashTable[i].head!=NULL)
					//{
						printf("%d - %s",i,hashTable[i].kelime);
						struct Node *nodeTmp = hashTable[i].head;
						while(nodeTmp != NULL)
						{
							printf(" - %s",nodeTmp->dosyaAdi);
							nodeTmp=nodeTmp->next;
						}
						printf("\n");
					//}
				}
			}
			else
			{
				printf("Once inputlari ekleyerek tabloyu doldurunuz.\n");
			}
			printf("\n\nMenuye donmek icin enter tusuna basin");
			getch();
		}
		else if(secim==5)//hashTable dizisi içerisindeki verileri 18011706.txt dosyasına yazar.
		{
			if(loadFactor>0)
			{
				FILE *yaz;
				if(yaz=fopen("18011706.txt","w")){
					fprintf(yaz,"Key Kelime BulunduguDosyalar\n");
					for(i=0;i<M;i++)
					{
						if(hashTable[i].head!=NULL)
						{
							fprintf(yaz,"%d %s",i+1,hashTable[i].kelime);
							struct Node *nodeTmp = hashTable[i].head;
							while(nodeTmp != NULL)
							{
								fprintf(yaz," %s",nodeTmp->dosyaAdi);
								nodeTmp=nodeTmp->next;
							}
							fprintf(yaz,"\n");
						}
					}
					fclose(yaz);
					printf("Veriler 18011706.txt dosyasina yazildi.");
				}
			}
			else
			{
				printf("Once inputlari ekleyerek tabloyu doldurunuz.\n");
			}

			printf("\n\nMenuye donmek icin enter tusuna basin");
			getch();
		}
		else if(secim==6)//18011706.txt isimli text dosyasındaki tüm verileri satır satır ekrana yazdırır.
		{
			printf("\nHash Tablosu\n");
```

```
                                satirOku("18011706.txt");
                                printf("\n\nMenuye donmek icin enter tusuna basin");
                                getch();
                        }
        }while(secim!=0);
        free(hashTable);
        printf("\n**Bellek Temizlendi**\n");
        return 0;
}
void kelimeOku(char dosyaAdi[100])
{
        FILE *fp=fopen(dosyaAdi,"r");
        if(fp=fopen(dosyaAdi,"r")){
                char bilgi[1000];
                while(!feof(fp)){
                        fscanf(fp,"%s",bilgi);
                        strcpy(bilgi,tolower(bilgi));
                        int hashParameter = hash(bilgi,horner(bilgi),dosyaAdi);
                        printf("%s - %d \n",bilgi,hashParameter);
                }
        }else{
                printf("Aradiginiz dosya yok.");
        }
}
void satirOku(char dosyaAdi[100])
{
        FILE *fp = fopen(dosyaAdi, "r");
        if(fp == NULL) {
    perror("Aradiginiz dosya yok. Lutfen once inputlari ekleyiniz. Ardindan menuden 5. secenek ile dosyaya yazdiriniz.\n");
  }
  else
  {
     char satir[1000];
     while(fgets(satir, sizeof(satir), fp) != NULL) {
                        fputs(satir, stdout);
                }
                fclose(fp);
        }
}
int horner (char kelime[1000])
{
        int i=0,len=0, x =2;
        int sonuc=0;
        len = strlen(kelime);
        while(kelime[i]!='\0')
        {
                sonuc += kelime[i] * getPower(x,i);
                i++;
        }
        return sonuc;
}
int getPower(int number, int power){
        int i,sum = 1;
        for(i = 0; i<power; i++){
                sum *= number;
        }
        return sum;
}
int hash(char kelime[1000],int hornerKey,char dosyaAdi[100]){
        int i= 0,hash = 0;
        do{
                i++;
                hash = (h1(hornerKey) + i*h2(hornerKey)) % M;
        }while((hashTable[hash].head != NULL && strcmp(hashTable[hash].kelime,kelime) != 0));
        if(strcmp(hashTable[hash].kelime,kelime) == 0){
                printf("ortak kelime bulundu\n");
```

```
                struct Node *tmp = hashTable[hash].head;
                while(tmp->next != NULL && strcmp(tmp->dosyaAdi,dosyaAdi) != 0){
                        tmp = tmp->next;
                }
                if(tmp->next == NULL && strcmp(tmp->dosyaAdi,dosyaAdi) != 0){
                        struct Node * node = (struct Node *)calloc(1,sizeof(struct Node));
                        tmp->next = node;
                        strcpy(tmp->next->dosyaAdi,dosyaAdi);
                        printf("Ayni kelime icin yeni dosya adi eklendi\n");
                }
                else
                {
                        printf("%s Kelimesi icin dosya zaten eklendi\n",hashTable[hash].kelime);
                }
        }

        else if(hashTable[hash].head == NULL){
                if(loadFactor==800)//LoadFactor 0.8e ulaştığı zaman kullanıcıya uyarı veriliyor. Eğer kullanıcı devam etmek isterse
veriler eklenmeye devam ediyor
                {
                        char durum="";
                        printf("Doluluk orani 0.8'e ulasti. Devam etmek icin 'd'. progami sonlandirmak icin 'c'\n");
                        do
                        {
                                durum=getchar();
                        }while(!(durum=='d' || durum=='c'));
                        if(durum=='c')//Eger kullanıcı c girerse program kapatılıyor.
                        {
                                exit(1);
                        }
                }
                if(loadFactor<=1009)//Tabloda müsait yer varsa veriler eklenmeye devam eder.
                {
                        strcpy(hashTable[hash].kelime,kelime);
                        struct Node * node = (struct Node *)calloc(1,sizeof(struct Node));
                        hashTable[hash].head = node;
                        strcpy(node->dosyaAdi,dosyaAdi);
                        printf("LoadFactor: %d Eklenen dosya adi: %s\n",loadFactor,dosyaAdi);
                        loadFactor++;
                }
                else//Tablo doluysa veri eklemesi durdurulur ve kullanıcıya uyarı verilir.
                {
                        printf("\n\n***Yetersiz bellek boyutu***\n\n");
                }
        }
        return hash;
}
struct Node * newNode(){
        struct Node * node = (struct Node *)calloc(1,sizeof(struct Node));
        return node;
}
int h1(int key){
        return key % M;
}
int h2(int key){
        return 1 + ( key % MM);
}
int kelimeAra(char kelime[1000],int hornerKey){
        int i= 0,hash = 0;
        do{
                i++;
                hash = (h1(hornerKey) + i*h2(hornerKey)) % M;
        }while((hashTable[hash].head != NULL && strcmp(hashTable[hash].kelime,kelime) != 0));
        printf("%d",hash);
        if(strcmp(hashTable[hash].kelime,kelime) == 0){
                return hash;
```

```
		}
		else{
			return -1;
		}

}
void baslangic(char hashDosyaAdi[])
{
		int sayac=0,sayac2=0;;
		int key;
		char kelime[1000];
		char dosyalar[1000];

		FILE *fp = fopen("18011706.txt", "r");
		if(fp == NULL) {
	perror("Baslangicta 18011706.txt bulunamadi. Lutfen once inputlari ekleyiniz. Ardindan menuden 5. secenek ile dosyaya
yazdiriniz.\n");
  }
  else
  {
		printf("18011706.txt isimli dosyadan veriler okunuyor ve hashTable isimli diziye aktariliyor...\n");
		char satir[1000];
		  while(fgets(satir, sizeof(satir), fp) != NULL) {
				if(sayac>=1)//Kelimeleri ekle
				{
					sscanf(satir,"%d %s\n",&key,kelime);
					strcpy(hashTable[key].kelime,kelime);
					loadFactor++;

					int init_size = strlen(satir);
					char delim[] = " \n";
					sayac2=0;
					char *ptr = strtok(satir, delim);

					while(ptr != NULL)
					{
						if(sayac2>=2)//Dosya adlarını ekle
						{
							struct Node * node = (struct Node *)calloc(1,sizeof(struct Node));
							if(hashTable[key].head==NULL)
							{
								hashTable[key].head = node;
								strcpy(node->dosyaAdi,ptr);
							}
							else
							{
								hashTable[key].head->next=node;
								strcpy(node->dosyaAdi,ptr);
							}
						}
						ptr = strtok(NULL, delim);
						sayac2++;
					}
				}
				sayac++;
			}
			fclose(fp);
		}
}
```