Algoritma Analizi 2.Ödev Raporu

ÖĞRENCİNİN ADI SOYADI

ÖMER LÜTFÜ TORTUMLU

ÖĞRENCİNİN NUMARASI

16011110

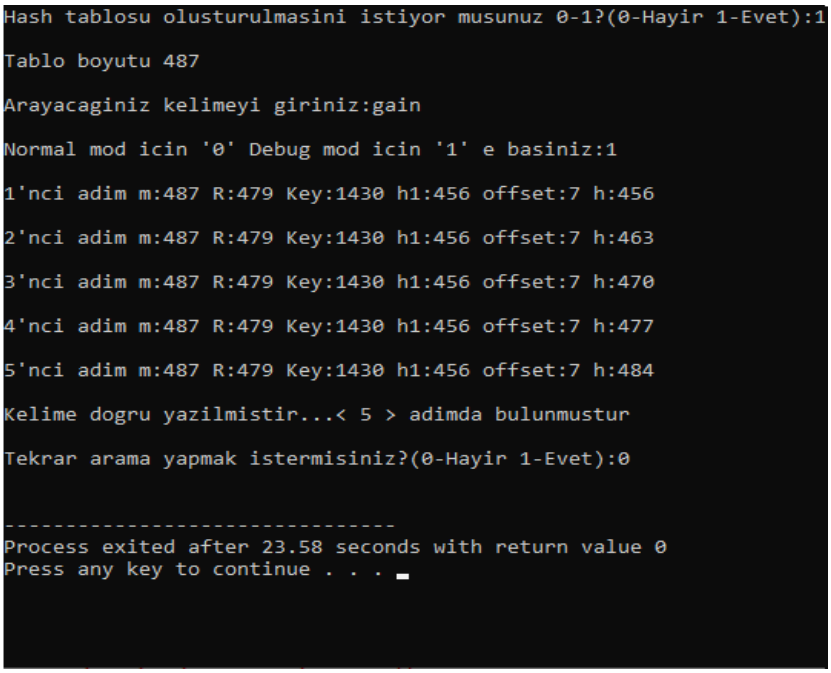
ÖDEV KONUSU

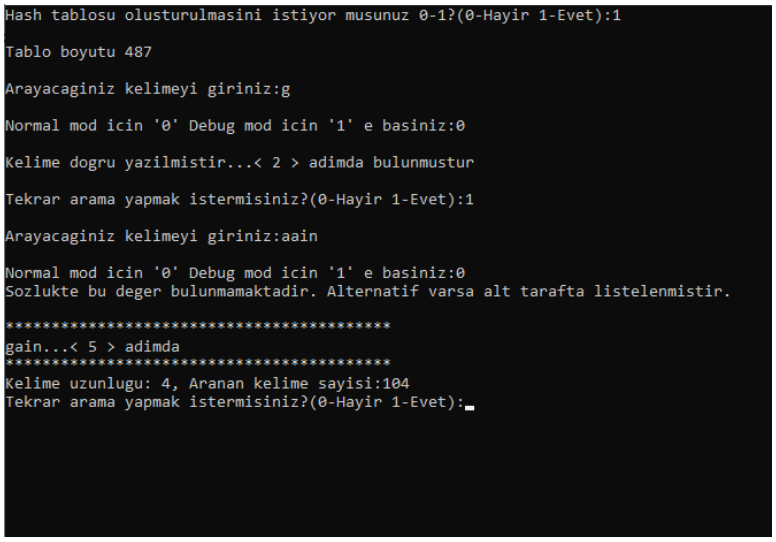
HASHİNG

YÖNTEM

- ➔ İlk olarak kullanıcıya hashlenmiş dosyanın olup olmadığı sorulur. Eğer varsa dosya ismi istenir. Dosya ismi doğruysa dosyadaki değerler hash tablosuna aktarılır.
- ➔ Kullanıcıdan aramak istediği kelime istenir. Kullanıcın sonucu debug mı normal mi görmek istediği sorulur normal bir arama isteniyorsa normal arama yöntemi fonksiyonu çağrılır eğer debug moda isteniyorsa debug fonksiyonunda sonuç yazdırılır. Kullanıcı kelimeyi girer kullanıcının girdiği değer hashlenir tablodaki değer ile aynıysa sözlükte olduğu yazısını gönderir eğer tabloda bulunmuyorsa kullanıcının bir harfi yanlış girdiği düşünülerek kelimedeki bütün harf kombinasyonları tek tek denenir. Bulunan değerler ekrana yazdırılır.
- ➔ Eğer kullanıcı tekrar bir arama yapmak isterse başa dönülür.

UYGULAMA





FACE ->Yok 461 Adım bakılıyor.
FACT ->Yok
GOLE ->Yok
GOLF ->Yok
HOLE ->Yok
HOLK ->Yok
BISK ->Yok
BITE ->Yok
DAME ->Yok
DAMN ->Yok
NIMS ->Yok
NINE ->Yok
PAIN ->
PAIR ->
LIVE ->
LOAD ->
YARD ->
YARE ->
ZERO ->
ZEST ->
FACX ->
FACZ ->
GOLT ->
GOLZ ->
HOLF ->
HOLR ->
BIST ->
BITF ->
DAMF ->
DAMZ ->
NIMT ->
NINF ->
PAIY ->
PAIZ ->
LIVT ->
LOAR ->
YARF ->
YART ->
ZERT ->
ZESZ->

KOD

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define m 487  //tablo uzunlugu icin gereken m = EnküçükAsalSayı >= TablodakiElemanSayısı / LoadFactor?
#define R 479   //R degerini m'den küçük ilk asal sayı olarak alınız.
#define l 15   //en uzun kelimeyi tutmak için gereken indis sayısı
int encrypt(char str[]){
	int i,key;
	for(i=key=0;i < strlen(str); i++)
		key +=(str[i]-'a')*26*i ;
	return key;
}
int h1(int key){
	return key % m;
}

int offset(int key){
	return R-(key%R);
}
int h(int key,int i){
	return (h1(key) + i*offset(key)) % m ;
}
int wordLength(char str[l]){
	int i,r=0;                                          //girilen kelimenin uzunluğunu
	for(i=0;i<l;i++){
		if(str[i]!='\0'){
			r++;
		}
	}
	return r;
}
void wordSearchNormal(char hash[m][l],char str[l],int *iteration,int *isthere){
	int r,j=0,i=0,k=0;
	r=wordLength(str);
	while(hash[h(encrypt(str),i)][0]!='\0' && str[j]!='\0' && k!=r){          //hash tablosunda bir değer varsa
		k=0;
		for(j=0;j<l;j++){
			if(hash[h(encrypt(str),i)][j]==str[j] && str[j]!='\0' && hash[h(encrypt(str),i)][j]!='\0' ){
				k++;          //	k kelimelerdeki eşit olan indislerin eşitlik sayısını tutuyor eğer girilen değer boyutuyla eş ise kelime bulunmuş oluyor.
			}
		}
		j=0;
		i++;
	}
	*iteration=i;
```

```
	*isthere=(k==r);
}
void wordSearchDebug(char hash[m][l],char str[l],int *iteration,int *isthere){
	int r,j=0,i=0,k=0;
	r=wordLength(str);
	while(hash[h(encrypt(str),i)][0]!='\0' && str[j]!='\0' && k!=r){		//hash tablosunda bir
değer varsa
		k=0;
		for(j=0;j<l;j++){
			if(hash[h(encrypt(str),i)][j]==str[j] && str[j]!='\0' &&
hash[h(encrypt(str),i)][j]!='\0' ){
				k++;		//	k kelimelerdeki eşit olan indislerin eşitlik
sayısını tutuyor eğer girilen değer boyutuyla eş ise kelime bulunmuş oluyor.
			}
		}
		j=0;
	printf("\n%d'nci adim m:%d R:%d Key:%d h1:%d offset:%d
h:%d\n",i+1,m,R,encrypt(str),h1(encrypt(str)),offset(encrypt(str)),h(encrypt(str),i));
		i++;
	}
	*iteration=i;
	*isthere=(k==r);
}
int main(){

	char word[l];
	char hash[m][l];
	int count = 0,i,j,key;		//hash tablosu varsa dosya okunurken değerin atanacağı
indisi tutuyor.
	for(i=0;i<m;i++){				//Hash tablosunu boşaltıyorum //m is defined
value
		for(j=0;j<l;j++){
			hash[i][j]='\0';
		}
	}
	j=0;
	int isHashTable;
	printf("Hash tablosu olusturulmasini istiyor musunuz 0-1?(0-Hayir 1-Evet):");
	scanf("%d",&isHashTable);
	getchar();
	if(isHashTable==1){
		char * filename = "smallDictionary.txt";
		FILE * fp = fopen(filename, "r");
		if (fp == NULL) return 1;		//Böyle bir dosya yoksa çalışma duruyor
		char c;					//okunan değerin atandığı karakter değişkeni
		while((c = fgetc(fp)) != EOF){			//Dosyanın son karakterine kadar
okunuyor
			if(c == ' ' || c == '\n')
			{
				i=0;
				while(hash[h(encrypt(word),i)][0]!='\0'){
					i++;
```

```
                        }
                        strcpy(hash[h(encrypt(word),i)],word);

                        for(j=0;j<l;j++){
                                word[j]='\0';
                        }
                        j=0;
                }
                else
                {
                        word[j]=c;
                        j++;
                }
        }
        fclose(fp);
        char * newHashFilename = "hashingTable.txt";
        FILE * fph = fopen(newHashFilename, "w");
        fclose(fph);
        for(i=0;i<m;i++){
                fopen(newHashFilename, "a+");
                if(hash[i][0]=='\0'){
                        fprintf(fph, "\n");
                }
                else{
                        for(j=0;j<l;j++){           //girilen kelimenin uzunluğunu
                                if(hash[i][j]!='\0'){
                                        fprintf(fph, "%c",hash[i][j]);
                                }
                        }
                        fprintf(fph, "\n");
                }
        }
        fprintf(fph, " ");          //En sona bir boşluk koyuyorum en sondaki cümleyi hash
tablosuna atamak için
        fclose(fph);
}
else{
        char oldHashFilename[100];// = "hashingTable.txt";
//      getchar();                  //scanf kullanarak bir değer okuduktan sonra program
otomatik olarak alt satıra geçer , scanf den sonra gets, fgets fonksiyonu kullandığınızda ise o karakter
dizisine '\n' karakteri gönderiyor.
        printf("Hashlenmis dosyanin adini giriniz:");

        gets(oldHashFilename);
        FILE * fpr = fopen(oldHashFilename, "r");
        if (fpr == NULL){
                printf("Boyle bir dosya bulunmamaktadir.");
//Böyle bir dosya yoksa çalışma duruyor
                return 1;
        }
        char cr;                                        //okunan değerin atandığı karakter
değişkeni
```

```
		while((cr = fgetc(fpr)) != EOF){			//Dosyanın son karakterine kadar okunuyor
			if(cr == ' ' || cr == '\n')
			{
				strcpy(hash[count],word);
				++count;
				for(j=0;j<l;j++){
					word[j]='\0';
				}
				j=0;
			}
			else
			{
				word[j]=cr;
				j++;
			}
		}
		fclose(fpr);
	}
//	for(i=0;i<m;i++){
//		printf("%s\n",hash[i]);
//	}
	printf("\nTablo boyutu %d\n\n",m);
	int repeat=1;
	while(repeat==1){

		printf("Arayacaginiz kelimeyi giriniz:");
	//	getchar();	//scanf kullanarak bir değer okuduktan sonra program otomatik olarak alt satıra geçer , scanf den sonra gets, fgets fonksiyonu kullandığınızda ise o karakter dizisine ‘\n’ karakteri gönderiyor.
		char str[l]={'\0'};
		gets(str);
		printf("\n");
		int iteration,isthere,debug,s=0;
		wordSearchNormal(hash,str,&iteration,&isthere);
		printf("Normal mod icin '0' Debug mod icin '1' e basiniz:");
		scanf("%d",&debug);
		if(isthere){
			if(debug==1)
				wordSearchDebug(hash,str,&iteration,&isthere);
			printf("\nKelime dogru yazilmistir...< %d > adimda bulunmustur\n", iteration);
		}
		else {
			printf("Sozlukte bu deger bulunmamaktadir. Alternatif varsa alt tarafta listelenmistir.\n");
			int a,b,search=0;			//search aranma sayisini tutuyor
			char tmp;
			for(a=0;a<wordLength(str);a++){
				tmp=str[a];
				for(b=0;b<26;b++){
					str[a]='a'+b;
```

```
                                        s++;
                                        wordSearchNormal(hash,str,&iteration,&isthere);
                                        search+=iteration;
                                        if(debug==1)
                                                printf("%s...<var mi? '%d' >  \n",str, isthere);
                                        if(isthere){

        printf("\n*****************************************");
                                                printf("\n%s...< %d > adimda \n",str, iteration);
                                                if(debug==1)

        wordSearchDebug(hash,str,&iteration,&isthere);


        printf("*****************************************\n");
                                        }
                                }
                                str[a]=tmp;
                        }
                        printf("Kelime uzunlugu: %d, Aranan kelime sayisi:%d, Adim
sayisi:%d",wordLength(str),s,search);
                        search=0;
                }
                printf("\nTekrar arama yapmak istermisiniz?(0-Hayir 1-Evet):");
                scanf("%d",&repeat);
                printf("\n");
                getchar();

        }

        return 0;
}
```