



YILDIZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

ALGORİTMA ANALİZİ DERSİ 3. ÖDEV 2. SORU

DERS YÜRÜTÜCÜSÜ: M.ELİF KARSLIĞİL

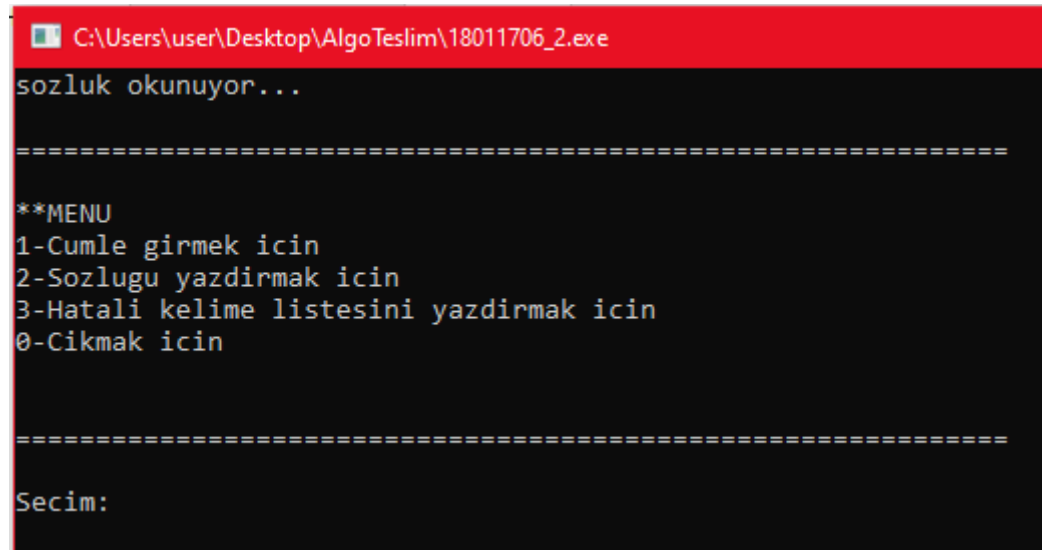
18011706

Burak Boz

Problem: Sorgulanan bir cümlede yanlış yazılmış kelimeler varsa bu kelimelerin yerine doğru kelimeler öneren bir sistem tasarlanacaktır.

Çözüm: Program çalıştırıldığı anda önce smallDictionary.txt isimli text dosyasından tüm kelimeler okunur, horner metoduyla hashlenerek hashtable isimli struct diziye eklenir. Kelimeler sözlüğe eklendikten sonra kullanıcının karşısına menü çıkarılır. 1. Seçenek seçilerek cümle girilir. Cümledeki kelimelerin hash tablosunda bulunup bulunmadığı kontrol edilir. Eğer kelimeler tabloda varsa doğru olarak değerlendirilir. Kelimeler tabloda yoksa yanlış olarak değerlendirilir ve Levenshtein distance metoduyla, hash tablosunda bulunan kelimelerle uzaklıkları değerlendirilir. Eğer mesafe 2 ve altı ise bu kelimeler kullanıcıya önerilir. Eğer hiçbir kelime 2 ve altı mesafede değilse önerilecek kelime olmadığı ve yerine yeni kelime girmesi istenir. Tüm kelimeler girildikten sonra cümlenin son hali ekrana gösterilir. Menüdeki diğer seçenekler, tüm hash tablosunu görmek veya hatalı kelimeler yerine önerilen kelimeleri görüntülemek için kullanılır. 0 seçimi ise bellekte programa ayrılan yerlerin tekrar işletim sistemine bırakılması ve programın kapatılması için kullanılır.

Menü:



```
C:\Users\user\Desktop\AlgoTeslim\18011706_2.exe
sozluk okunuyor...

=====

**MENU
1-Cumle girmek icin
2-Sozlugu yazdirmak icin
3-Hatali kelime listesini yazdirmak icin
0-Cikmak icin

=====

Secim:
```

Ekran Çıktıları:

Çıktı 1:

İnput: it is hood

İt doğru kelime

İs doğru kelime

Hood yerine kelimeler önerilmiştir ve girdi olarak **good** kelimesi girilmiştir. Cümlenin son hali yazdırılmıştır.

```
C:\Users\user\Desktop\AlgoTeslim\18011706_2.exe
sozluk okunuyor...

=====

**MENU
1-Cumle girmek icin
2-Sozlugu yazdirmak icin
3-Hatali kelime listesini yazdirmak icin
0-Cikmak icin

=====

Secim:1
lutfen cumlenizi girin:it is hood
it (kelime dogru)
is (kelime dogru)
hood - bunu mu demek istediniz?
    load
    hold
    hour
    had
    how
    good
    word
    look
    head
    Yeni kelimeyi girin (hatali kelime yerine cumleye eklenecektir.):good

Cumlenin son hali : it is good
```

Çıktı 2:

Bu örnekte rastgele kelimeler girilmiştir. Girilen kelimeler:

Commad (Yanlış) -> command

Burak (Sözlükte bulunmuyor) -> boz

Abotu (Yanlış) -> about

Suppose (Doğru)

Belov (Yanlış) -> below

Düzeltilmeler eklendikten sonra cümlelerin son hali: command boz about suppose below

```
C:\Users\user\Desktop\AlgoTeslim\18011706_2.exe

**MENU
1-Cumle girmek için
2-Sozlugu yazdirmek için
3-Hatali kelime listesini yazdirmek için
0-Cikmak için

=====

Secim:1
lutfen cumlenizi girin:commadd burak abotu suppose below
commadd - bunu mu demek istediniz?
    command
    Yeni kelimeyi girin (hatali kelime yerine cumleye eklenecektir.):command
burak - kelime yanlis fakat onerilecek kelime yok. Lutfen dogru halini tekrar yazin
(hatali kelime yerine cumleye eklenecektir.):boz
abotu - bunu mu demek istediniz?
    above
    about
    Yeni kelimeyi girin (hatali kelime yerine cumleye eklenecektir.):about
suppose (kelime dogru)
below - bunu mu demek istediniz?
    below
    Yeni kelimeyi girin (hatali kelime yerine cumleye eklenecektir.):below

Cumlenin son hali : command boz about suppose below
```

Girdiler yapıldıktan sonra hatalı kelime listesi yazdıralım;

```
C:\Users\user\Desktop\AlgoTeslim\18011706_2.exe

**MENU
1-Cumle girmek için
2-Sozlugu yazdirmek için
3-Hatali kelime listesini yazdirmek için
0-Cikmak için

=====

Secim:3
hatali hash tablosu:
abotu => above about
commadd => command
burak => boz
below => below
```

Çıktı 3:

Sözlüğe eklenen tüm kelimeleri görmek için 2 seçimi yapılırsa;

```
C:\Users\user\Desktop\AlgoTeslim\18011706_2.exe
812 - using
815 - available
819 - length
822 - letter
823 - hashing
824 - words
826 - lower
831 - size
835 - following
838 - friday
843 - with
845 - support
848 - deleted
856 - functions
858 - look
865 - successfully
873 - discover
876 - class
880 - keeping
884 - usual
888 - that
892 - incorrect
895 - handle
900 - submit
903 - debugging
905 - than
909 - information
915 - primary
916 - known
917 - exactly
923 - instructions
935 - adjacent
936 - works
937 - try
940 - string
944 - error
945 - dictionary
948 - algorithm
949 - expand
962 - collisions
963 - deducted
964 - produce
966 - something
968 - separated
972 - head
974 - udder
978 - factor
979 - variant
988 - messages
990 - command
992 - generated
997 - single
998 - below
1000 - warning
1005 - unlike
1006 - suppose
```

Çıktı 4:

Programdan çıkmak için 0 seçimi yapılırsa;

```
C:\Users\user\Desktop\AlgoTeslim\18011706_2.exe
sozluk okunuyor...

=====

**MENU
1-Cumle girmek icin
2-Sozlugu yazdirmek icin
3-Hatali kelime listesini yazdirmek icin
0-Cikmak icin

=====

Secim:0

Bellek temizlendi.

-----
Process exited after 0.8726 seconds with return value 0
Press any key to continue . . .
```

Kaynak Kodlar:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define M 1009
#define MM 1008

typedef struct { //Hash tipinde bir dizi tanımlanacak ve burada kelimeler saklanacaktır.
    char kelime[1000];
}HASH;

typedef struct { //Hatalıhash tipinde bir dizi tanımlanacak ve Hatalı girilen kelimeler burada tutulacaktır.
    char kelime[1000];
    struct Node* head;
}HATALIHASH;

struct Node{ //Bu düğümlerde hatalı girilen kelime yerine önerilecek kelimeler linkli liste tipinde tutulacaktır.
    char onerilen[1000];
    struct Node* next;
};

//Fonksiyon prototipleri - Tüm fonksiyonlara açıklamaları yazılmıştır.
struct Node * kelimeAra(char []);
void yeniEkle(char * hatali,char * guncel);
void sMerge(char * );
```

```
int errHash(char *,int);
int distance(char *,char *);
void hashTableYazdir();
void hataliHashTableYazdir();
int hash(char *,int);
int h1(int);
int h2(int);
int horner (char *);
int getPower(int,int);
void baslangic(char *);
```

```
HASH *hashTable;           //Sözlük kelimeleri bu dizide tutulacaktır
HATALIHASH *hashErrorTable; //Hatalı girilen kelimeler bu dizide tutulacaktır
int loadFactor;           //Sözlük dizisinin doluluk oranı bu değişkende saklanacaktır
int loadFactorError;      //Hatalı kelimeler dizisinin doluluk oranı bu değişkende saklanacaktır.
char birlesmisCumle[5000]; //En son merge edilen cümle bu dizide saklanacaktır.
int kelimeDurum=0;        //Kelimenin sözlükteki durumunu kontrol etmek için kullanılacaktır.
```

```
int main()
{
    char cumle[1000]; //Kullanıcıdan alınan cümle
    int i=0;          //Döngüler için kullanılacaktır
    int cumleSayac=0; //Kullanıcıdan alınan cümlelerin uzunluğunu tutacaktır.
    int kelimeSayac=0; //Cümledeki işlenen kelimenin uzunluğunu tutacaktır.
    hashTable = (HASH*)calloc(M, sizeof(HASH));
    hashErrorTable = (HATALIHASH*)calloc(M, sizeof(HATALIHASH));
    loadFactor=0;
    loadFactorError=0;
    birlesmisCumle[0]='\0';//Cümle boşaltıldı
    int secim;//Menü için kullanılacak.
    for(i=0;i<M;i++)//Hazırlanan tablolar temizleniyor.
    {
        memset(hashTable[i].kelime,0,sizeof(hashTable[i].kelime));
        memset(hashErrorTable[i].kelime,0,sizeof(hashErrorTable[i].kelime));
        hashErrorTable[i].head=NULL;
    }
    printf("sozluk okunuyor...");//Sözlük okunuyor ve ekrana yazdırılıyor
    baslangic("smallDictionary.txt");
    do
    {
        printf("\n\n=====\\n\\n");
        printf("**MENU\\n");
        printf("1-Cumle girmek icin\\n");
        printf("2-Sozlugu yazdirmek icin\\n");
        printf("3-Hatali kelime listesini yazdirmek icin\\n");
        printf("0-Cikmak icin\\n");
        printf("\n\n=====\\n\\n");
        printf("Secim:");
        scanf("%d",&secim);
        getchar();
        if(secim==0)
        {
            free(hashErrorTable);
            free(hashTable);
            printf("\nBellek temizlendi.\\n");
            return 0;
        }
    }
}
```

```

else if(secim==1)
{
    cumle[0]='\0';
    printf("lutfen cumlenizi girin:");//Kullanıcıdan cümle alınıyor.
    gets(cumle);
    int init_size = strlen(cumle);//Cümle kelimelere bölünüyor ve işlenmeye başlanıyor
    char delim[] = " ";
    char *ptr = strtok(cumle, delim);
    while(ptr != NULL)
    {
        char yeniKelime[100];
        struct Node* gelenKelime = kelimeAra(ptr);    //Kelime sözlükte aranmaya
başlanıyor.
        if(kelimeDurum==-1)// Kelime doğru (Eğer kelime sözlükte mevcutsa kelimeDurum
değişkeni -1 olarak güncellenir.)
        {
            printf("%s (kelime dogru)\n",ptr);
            sMerge(ptr);
        }
        else if(kelimeDurum==1)//kelimeDurum Kelime yanlış fakat önerilecek kelime yok.
Bu durumda kullanıcıdan alınan kelime hatalı kelimenin yanına önerilen olarak eklenir
        {
            printf("%s - kelime yanlis fakat onerilecek kelime yok. Lutfen dogru halini
tekrar yazin\n(hatali kelime yerine cumleye eklenecektir.):",ptr);
            gets(yeniKelime);
            yeniEkle(ptr,yeniKelime);
            sMerge(yeniKelime);
        }
        else//Diğer iki durum gerçekleşmemişse kelime yanlıştır ve listede önerilecek kelime
vardır. Bu kelimeler kullanıcıya önerilir
        {
            int sayac=0;
            printf("%s - bunu mu demek istediniz?\n",ptr);
            while(gelenKelime!=NULL)
            {
                sayac++;
                printf("\t%s\n",gelenKelime->onerilen);
                gelenKelime=gelenKelime->next;
            }
            printf("\tYeni kelimeyi girin (hatali kelime yerine cumleye eklenecektir.):");
            gets(yeniKelime);
            sMerge(yeniKelime);
        }
        ptr = strtok(NULL, delim);
    }
    printf("\n\nCumlenin son hali : %s\n\n",birlesmisCumle);//Cümlenin son hali ekrana yazdırılır
    memset(birlesmisCumle, '\0', sizeof birlesmisCumle);
}
else if(secim==2)//Sözlüğü yazdır
{
    hashTableYazdir();
}
else if(secim==3)//Hatalı kelimeleri yazdır
{
    hataliHashTableYazdir();
}
else
{
    printf("Gecersiz secim\n");
}

```



```

        }
    }while(secim!=0);
}
void yeniEkle(char * hatali,char * guncel)//(hatalı kelime, yerine girilecek güncel kelime) -> void
{
    int kontrol=0;
    int hornerKey = horner(hatali);
    int i= 0,hash = 0;
    do
    {
        i++;
        hash = (h1(hornerKey) + i*h2(hornerKey)) % M;
        if(strcmp(hashErrorTable[hash].kelime,hatali) == 0)//Hashlenen kelime doğrumu
        {
            kontrol=1;
            i==1010;
        }
    }while(((kontrol!=1) && (strlen(hashErrorTable[hash].kelime) != 0)) && i<1009);

    struct Node *yeniKelime = (struct Node *) calloc(1,sizeof(struct Node));
    strcpy(yeniKelime->onerilen,guncel);
    yeniKelime->next=NULL;
    hashErrorTable[hash].head=yeniKelime;
}
void sMerge(char * ptr)//(kelime) -> void | birleşmiş cümle değişkeninde uygun kelimeler birleştirilir.
{
    int kelimeSayac=strlen(ptr);
    int cumleSayac=strlen(birlesmisCumle);
    int i=0;
    for(i=0;i<kelimeSayac;i++)
    {
        birlesmisCumle[cumleSayac+i]=ptr[i];
    }
    birlesmisCumle[strlen(birlesmisCumle)]= ' ';
    birlesmisCumle[strlen(birlesmisCumle)+1]='\0';
}
struct Node * kelimeAra(char kelime[])//(AranacakKelime) return -> önerilecek kelimeler | önerilecek kelime yoksa
kelimeDurum değişkeni güncellenir.
{
    struct Node *gelenKelimeler;
    int kontrol=0;
    int hornerKey = horner(kelime);
    int i= 0,hash = 0;
    do
    {
        i++;
        hash = (h1(hornerKey) + i*h2(hornerKey)) % M;
        if(strcmp(hashTable[hash].kelime,kelime) == 0)//Hashlenen kelime doğrumu
        {
            kontrol=1;
            i==1010;
        }
    }while(((kontrol!=1) && (strlen(hashTable[hash].kelime) != 0)) && i<1009);
    if(kontrol==1)//Kelime tamam
    {
        gelenKelimeler=NULL;
        kelimeDurum=-1;
        return gelenKelimeler;
    }
}

```

```

else//Hatalı kelime tablosu işlemleri
{
    gelenKelimeler = (struct Node *)calloc(1,sizeof(struct Node));
    gelenKelimeler = hashErrorTable[errHash(kelime,horner(kelime))].head;
    if(gelenKelimeler==0)//Önerilecek kelime bulunamadı
    {
        kelimeDurum=1;
        return gelenKelimeler;
    }
    else
    {
        kelimeDurum=0;
        return gelenKelimeler;
    }
}
return gelenKelimeler;
}
int errHash(char kelime[1000],int hornerKey){//(hashlenecek Kelime, kelimenin horner methoduyla çevrilmiş sayısal hali) ->
return hesaplanan hash değeri döndürülür.
/*
    Bu fonksiyonda hatalı kelimeler hashlenir. Kelimenin durumu kontrol edildikten sonra
    tüm kelimelerle mesafe hesaplaması yapılır ve uygun bulunan kelimeler linkli liste
    şeklide eklenir. Eğer önerilebilecek kelime yoksa head kısmı Null olarak kalır.
    hesaplanan hash değerini döndürür.
*/
int i= 0,hash = 0;
do{
    int kelimeKontrol=0;
    i++;
    hash = (h1(hornerKey) + i*h2(hornerKey)) % M;
    if(strlen(hashErrorTable[hash].kelime)==0)//Yanlış gelen kelime error tablosuna hashlenMEMiştir
    {
        for(i=0;i<M;i++)//Tüm kelimelerle mesafeleri kontrol edilecek
        {
            if(strlen(hashTable[i].kelime)>0)//mesafesi kontrol edilecek kelime
            {
                int mesafe = distance(kelime,hashTable[i].kelime);
                if(mesafe<=2)
                {
                    kelimeKontrol++;

                    if(hashErrorTable[hash].head==NULL)
                    {
                        struct Node *yeniDugum = (struct Node
*)calloc(1,sizeof(struct Node));

                        hashErrorTable[hash].head=yeniDugum;
                        yeniDugum->next=NULL;
                        strcpy(yeniDugum->onerilen,hashTable[i].kelime);
                    }
                    else
                    {
                        struct Node *nodeTmp = hashErrorTable[hash].head;
                        while(nodeTmp->next != NULL)
                        {
                            nodeTmp=nodeTmp->next;
                        }
                        struct Node *yeniDugum = (struct Node
*)calloc(1,sizeof(struct Node));

```

```

nodeTmp->next=yeniDugum;
yeniDugum->next=NULL;
strcpy(yeniDugum->onerilen,hashTable[i].kelime);
    }
    }
    }
    strcpy(hashErrorTable[hash].kelime,kelime);
    loadFactorError++;
}
}while(strcmp(hashErrorTable[hash].kelime,kelime) != 0 && i<1009);
return hash;
}
int distance(char * kelime1,char * kelime2)//(kelime1, kelime2) -> return kelimeler arasındaki mesafe
{
    int len1=0;
    int len2=0;
    len1=strlen(kelime1);
    len2=strlen(kelime2);
    int j;
    int i;
    int matris[len1 + 1][len2 + 1];
    for (i = 0; i <= len1; i++) {
        matris[i][0] = i;
    }
    for (i = 0; i <= len2; i++) {
        matris[0][i] = i;
    }
    for (i = 1; i <= len1; i++) {

        char c1;

        c1 = kelime1[i-1];
        for (j = 1; j <= len2; j++) {
            char c2;

            c2 = kelime2[j-1];
            if (c1 == c2) {
                matris[i][j] = matris[i-1][j-1];
            }
            else {
                int sil;
                int ekle;
                int substitute;
                int minimum;

                sil = matris[i-1][j] + 1;

                ekle = matris[i][j-1] + 1;
                substitute = matris[i-1][j-1] + 1;
                minimum = sil;
                if (ekle < minimum) {
                    minimum = ekle;
                }
                if (substitute < minimum) {
                    minimum = substitute;
                }
                matris[i][j] = minimum;
            }
        }
    }
}

```

```

    return matris[len1][len2];
}

void hashTableYazdir() //Sözlüğün kaydedildiği hash tablosunu yazdırır.
{
    int i=0;
    printf("hash tablosu:\n");
    for(i=0;i<M;i++)
    {
        if(strlen(hashTable[i].kelime)>0)
        {
            printf("%d - %s\n",i,hashTable[i].kelime);
        }
    }
}

void hataliHashTableYazdir() //Hatalı kelimelerin ve bu kelimeler için bulunan önerileri ekrana yazdırır.
{
    int i=0;
    printf("hatali hash tablosu:\n");
    for(i=0;i<M;i++)
    {
        if(hashErrorTable[i].head!=NULL)
        {
            printf("%s => ",hashErrorTable[i].kelime);
            struct Node *nodeTmp = hashErrorTable[i].head;
            while(nodeTmp != NULL)
            {
                printf("%s ",nodeTmp->onerilen);
                nodeTmp=nodeTmp->next;
            }
            printf("\n");
        }
    }
}

int hash(char kelime[1000],int hornerKey){ //Kelimleri hashlemek için kullanılır.
    //printf("kelime : %s\n",kelime);
    int i= 0,hash = 0;
    do{
        i++;
        hash = (h1(hornerKey) + i*h2(hornerKey)) % M;
        if(strlen(hashTable[hash].kelime)==0)
        {
            strcpy(hashTable[hash].kelime, kelime);
            //printf("LoadFactor: %d \n",loadFactor);
            loadFactor++;
        }
        else
        {
            if(strcmp(hashTable[hash].kelime, kelime) == 0)
            {
                //printf("ortak kelime bulundu. bulunan kelime : %s",kelime);
            }
        }
    }while(strcmp(hashTable[hash].kelime, kelime) != 0 && i<1009);

    return hash;
}

```

```

int h1(int key){    //Hashlemek için kullanılan yardımcı fonksiyon
    return key % M;
}
int h2(int key){    //Hashlemek için kullanılan yardımcı fonksiyon
    return 1 + ( key % MM);
}
int horner (char kelime[1000])    //Kelimelerin horner methoduyla sayısal değerlere çevrilmesi
{
    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){ //Hashlemek için kullanılan yardımcı fonksiyon. Üs almak için kullanılır.
    int i,sum = 1;
    for(i = 0; i<power; i++){
        sum *= number;
    }
    return sum;
}
void baslangic(char hashDosyaAdi[])    //Başlangıçta sözlüğün okunup hashTable isimli diziye işlenmesi için
    kullanılan fonksiyon
{
    FILE *fp=fopen(hashDosyaAdi,"r");
    if(fp=fopen(hashDosyaAdi,"r")){
        char bilgi[1000];
        while(!feof(fp)){
            fscanf(fp,"%s",bilgi);
            strcpy(bilgi,tolower(bilgi));
            int hashParameter = hash(bilgi,horner(bilgi));
            //printf("%s - %d \n",bilgi,hashParameter);

        }
    }else{
        printf("Aradiginiz dosya yok.");
    }
}

```