

KAHRAMANMARAŞ SÜTÇÜ İMAM ÜNİVERSİTESİ

**Algoritma Analizi ve Tasarımı Dersi Proje Uygulaması ve
Dokümantasyonu**

BİR KELİMENİN ANAGRAMLARININ BULUNMASI

HAZIRLAYANLAR

**Mehmet ÇAM
Alperen USLU**

HAZİRAN-2023

Kodun İşleyişi	3
Pseudo Kod	7
Hash Tablosu Oluşturma Karmaşıklık Analizi	10
Tabloda Arama Karmaşıklık Analizi	10
Hash Tablosu Oluşturma Çalışma Süresi	11
Tabloda Arama Çalışma Süresi	11

Kodun İşleyişi

İlk olarak kelimeler.txt dosyası okuma modunda açılır.

“while” döngüsü ile dosyanın sonuna gidene kadar “\n” leri sayarız, bu bize dosyada kaç tane satır olduğunu belirtir.

İmleci geri en başa almak için “rewind()” methodu kullanılır.

Her gerektiğinde “kelimeler.txt” dosyasını tekrar tekrar okumamak için “kelimelerDizisi” oluşturulur.

Kelimelerin tablodaki yerini bulmak için kaç defa double hashing yapıldığını her kelime için bir dizide tutmamız gerekir, aksi halde kelimenin hash tablosundaki yerini bulmak çok zor olacaktır.

Dizinin adı “doubleHashSayisi”.

İlk değer ataması olarak “doubleHashSayisi” dizisinin değerlerine “0” ataması yapılır.

“while” döngüsü ile “kelimelerDosyasi” ’ndaki tüm kelimeleri “kelimeler” dizisine satır satır kopyalayarak “kelimelerDosyasi” ’nin sonuna gidene kadar döngüye devam edilir.

“strcpy()” methodu ile kelimeler “kelimelerDizisi” ’ne kaydedilir.

Bu sayede dosyayı tekrar tekrar okumak yerine diziyi kullanarak işlemlerimizi gerçekleştirebiliriz.

Hashleme işlemi yapılır.

Hashleme işlemi bize “int” bir değer döndürür ve biz bu değeri “1” arttırarak tablodaki indekse atarız.

“1” arttırmamızda ki sebep eğer gelen değer “0” olursa tablonun ilk değer ataması “0” olduğu için değer var mı yok mu nereye atandı bilinemez.

Bu karmaşıklığı önlemek için atanacak değer indeksin “1” fazlası olacak şekilde “hashTablosu” doldurulur.

Tekrar “while” döngüsü ile tablonun ilgili yeri boş mu dolu mu (çakışma durumu) kontrol edilir.

Eğer çakışma varsa “doubleHashleme” yapılır.

İlgili satırın “doubleHashSayisi” değeri her double hashleme yapıldığında “1” artırılır ve “doubleHashSayisi” dizisine kaydedilir.

Bu sayede ilgili kelimeye kaç defa double hashing yapıldığı kaydedilmiş olur ve istendiği zaman hash tablosundaki indeksi tekrardan bulunabilir.

“hashleme” fonksiyonu kelimeyi , hash tablosunun boyutunu ve kelimenin boyutunu parametre olarak alır ve kelimenin her bir harfini teker teker “key” ile toplayıp, “61” ile çarparak, key değerine atama yapar.

Bu işlemde kelimenin harflerinin ascii karşılığı ile işlemler yapılır.

Döngü bittikten sonra “key” değeri “tabloBoyutu” ’na bölünüp kalan “return” edilir.

“doubleHashleme” fonksiyonu da “hashleme” fonksiyonu gibi aynı parametreleri alır ancak fazladan bir “int” parametre daha alır.

Aldığı bu değer double hashleme sayısıdır.

“h1” ve “h2” değişkenleri ile işlemler yapılır.

“h1” değerine “hashleme” fonksiyonundan döndürülen değer atanır.

“h2” değerine asal bir sayıdan (biz 19 seçmişiz) “hashleme” fonksiyonundan döndürülen değer in asal bir değere (biz yine 19 seçmişiz) bölümünden kalan atanır.

“m” değeri hashTablosu dizisinin boyutudur.

Bu değer satır sayısının yük faktörüne (load factor) bölünmesiyle bulunur.

Hash fonksiyondan gelen değer ile double hashleme sayısı toplanır, “h2” değeri ile çarpılır,

“m” değerine bölümünden kalan, “return” edilir.

Bu sayede oldukça karmaşık, girdi ile alakası olmayan bir değer bulunmuş olur.

Bu değer hash tablosuna kaydedilmeden önce tekrar tekrar her kelime için tablonun

ilgili alanının boş mu dolu mu olduğu kontrol edilir.

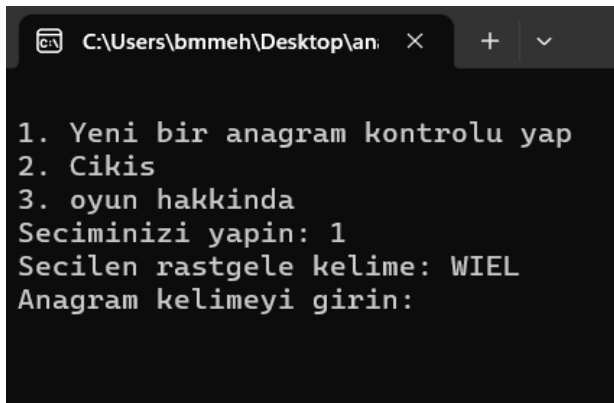
Dolu ise tekrar “doubleHashleme” fonksiyonu çağrılır ve aynı işlemler tekrarlanır değişen tek parametre kaç defa double hashleme yapıldığını tutan değişkendir.

Eğer uygun şartlar sağlanırsa hash tablosu oluşturulur.

kelimeler.txt dosyası kapatılır.

Kullanıcının asıl işlemlerini yaptığı “switch-case” yapısında ise ;

“case 1” Oyunu başlatır ve random olarak kelimeler dizisinden bir kelimeyi ekrana basar.

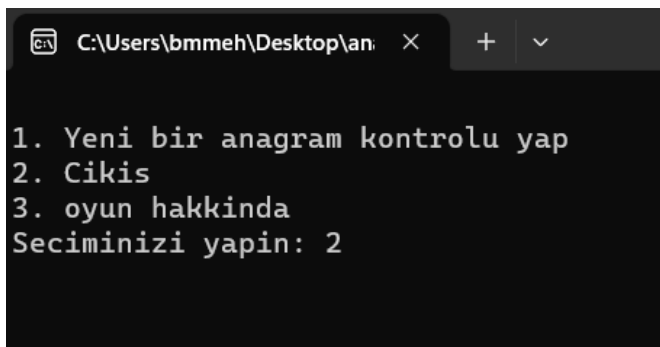


```

C:\Users\bmmeh\Desktop\an... X + v
1. Yeni bir anagram kontrolu yap
2. Cikis
3. oyun hakkında
Seciminizi yapin: 1
Secilen rastgele kelime: WIEL
Anagram kelimeyi girin:

```

“case 2” kullanıcı sonlandırmak istediğinde oyunu sonlandırır,



```

C:\Users\bmmeh\Desktop\an... X + v
1. Yeni bir anagram kontrolu yap
2. Cikis
3. oyun hakkında
Seciminizi yapin: 2

```

“case 3” oyun hakkında bilgi verir.

```
C:\Users\bmmeh\Desktop\an  x + v

1. Yeni bir anagram kontrolu yap
2. Cikis
3. oyun hakkında
Seciminizi yapin: 3
Her oyunun basinda bu secim ekranini goreceksiniz bu cok sinir bozucu biliyorum :)
Eger 1. secenegi secerseniz oyun baslar ve ekrana rastgele bir kelime cikacak.
Sizden o kelimenin anagramini girmeniz bekleniyor.
Siz kelimeyi girdikten sonra sonuc ekranda gorunecek herhangi bir tusa basarak devam edebilirsiniz.
Son olarak lutfen gidişiniz kelimelerin hepsi buyuk harflerden olussun.
Artik hazirsiniz, herhangi bir tusa basip secim ekranina gidebilirsiniz.
Iyi eglenceler.
```

Hatalı girdi de kullanıcıya bilgi verilir.

```
C:\Users\bmmeh\Desktop\an  x + v

1. Yeni bir anagram kontrolu yap
2. Cikis
3. oyun hakkında
Seciminizi yapin: 5a
Gecersiz secim!
```

Kullanıcıdan ekrana basılan kelimenin anagramını girmesi istenir.

Kullanıcıdan girdi alınır.

“for” döngüsünün içine girilir bu döngü satır sayısı kadar döner ama her zaman sonuna kadar çalışmaz.

“anagramMi” methodu çağrılır bu method “bool” değer dönderir.

Parametre olarak iki adet kelime alır ve bu kelimelerin anagram olup olmadığını belirler.

Çalışma mantığı ise sayac diye bir dizi oluşturulur ve bu dizinin boyutu ascii tablosu kadardır (256).

İlk “for” döngüsü satır sonuna kadar gider ve ilk kelimenin her harfinin “sayac” dizisindeki indeks değerini “1” artırılır.

İkinci “for” döngüsü satır sonuna kadar gider ve ilk kelimenin her harfinin “sayac” dizisindeki indeks değerini “1” azaltır.

Eğer kelimeler aynı harflerden oluşuyorsa sayac dizisindeki tüm değerler yeniden “0” lanmış olurlar.

Eğer “0” dan farklı bir değer varsa, bu kelimelerin anagram olmadığını gösterir.

“anagramMi” methodundaki son for döngüsü ise “256” defa dönerek tüm değerler “0” mı kontrol eder.

Eğer bir tane değer bile “0” dan farklı ise “false” “return” edilir aksi halde “true” “return” edilir.

Daha sonra girilen kelime “strcmp()” methodu ile “kelimelerDizisinde” var mı kontrol edilir.

Eğer yoksa “Girdiğiniz kelime hash tablosunda yok.” bilgisi ekrana bastırılır.

Eğer varsa kelimenin tablodaki yeri bulunur ve girilen kelime tekrar hashlenir.

Kaç defa double hashleme yapıldıysa,”doubleHashSayisi” dizisinden gelen hash değeri kadar double hashing yapılarak kelimenin hash değeri tekrar bulunur. Dizin değerinin “1” eksiği tablodaki indeksi verir.

Nedenini yukarıda açıklamıştık.

Son olarak ise ekrana bilgiler bastırılır ve tekrar menü ekranına dönülüp kullanıcıdan seçim yapması beklenir.

```
C:\Users\bmmeh\Desktop\an. x + v
1. Yeni bir anagram kontrolu yap
2. Cikis
3. oyun hakkında
Seciminizi yapin: 1
Secilen rastgele kelime: DAUR
Anagram kelimeyi girin: DURA
Girdiginiz kelimenin hash tablosundaki indeksi: 215
Cakisma var: 1 defa.
Girdiginiz kelime anagram.
```

```
C:\Users\bmmeh\Desktop\an. x + v
1. Yeni bir anagram kontrolu yap
2. Cikis
3. oyun hakkında
Seciminizi yapin: 1
Secilen rastgele kelime: EASY
Anagram kelimeyi girin: SAVE
Girdiginiz kelime anagram degil.
```

```
C:\Users\bmmeh\Desktop\an. x + v
1. Yeni bir anagram kontrolu yap
2. Cikis
3. oyun hakkında
Seciminizi yapin: 1
Secilen rastgele kelime: BADS
Anagram kelimeyi girin: SABD
Girdiginiz kelime hash tablosunda yok.
Girdiginiz kelime anagram.
```

Pseudo Kod

```
DECLARE LOAD_FACTOR = 0.6
DECLARE hashTablosu[8500] = {0}
```

```
FUNCTION hashleme(satir, tabloBoyutu, boyut)
  DECLARE key = 0
  FOR i = 0 TO boyut
    key = 61 * (key + satir[i])
  ENDFOR
  RETURN key % tabloBoyutu
ENDFUNCTION
```

```
FUNCTION doublehashleme(satir, tabloBoyutu, i, boyut)
  DECLARE h1 = hashleme(satir, tabloBoyutu, boyut)
  DECLARE h2 = 19 - (hashleme(satir, tabloBoyutu, boyut) % 19)
  DECLARE m = tabloBoyutu / LOAD_FACTOR
  RETURN ((h1 + i) * h2) % m
ENDFUNCTION
```

```
FUNCTION anagramMi(str1, str2)
  DECLARE sayac[256] = {0}
  FOR i = 0 TO length(str1)
    sayac[str1[i]]++
  ENDFOR
  FOR i = 0 TO length(str2)
    sayac[str2[i]]--
  ENDFOR
  FOR i = 0 TO 256
    IF sayac[i] != 0 THEN
      RETURN false
    ENDIF
  ENDFOR
  RETURN true
ENDFUNCTION
```

```
FUNCTION main()
  DECLARE LOAD_FACTOR = 0.6
  DECLARE hashTablosu[8500] = {0}
  DECLARE kelimeDosyasi
  DECLARE kelimeler[6] = ""
  DECLARE boyut = 0
  DECLARE dizin
```

```

kelimeDosyasi = open_file("kelimeler.txt", "r")
IF kelimeDosyasi == NULL THEN
    print("Dosya açma hatası!!!")
    RETURN 1
ENDIF
DECLARE satirSayisi = 1
DECLARE gez
WHILE gez = get_character(kelimeDosyasi) != EOF
    IF gez == '\n' THEN
        satirSayisi++
    ENDIF
ENDWHILE
rewind(kelimeDosyasi)
DECLARE kelimelerDizisi[satirSayisi][6]
DECLARE doubleHashSayisi[satirSayisi][1]
DECLARE say = satirSayisi - 1
WHILE say >= 0
    doubleHashSayisi[say][0] = 0
    say--
ENDWHILE
DECLARE i = 0
WHILE fgets(kelimeler, sizeof(kelimeler), kelimeDosyasi) != NULL
    strcpy(kelimelerDizisi[i], kelimeler)
    boyut = length(kelimeler) - 1
    dizin = hashleme(kelimeler, satirSayisi, boyut)
    DECLARE j = 0
    WHILE hashTablosu[dizin] != 0
        j++
        dizin = doublehashleme(kelimeler, satirSayisi, j, boyut)
    ENDWHILE
    hashTablosu[dizin] = dizin + 1
    i++
ENDWHILE
fclose(kelimeDosyasi)
DECLARE devamEt = true
WHILE devamEt
    print("\n1. Yeni bir anagram kontrolu yap\n")
    print("2. Cikis\n")
    print("3. Oyun hakkında\n")
    print("Seciminizi yapin: ")
    DECLARE secim
    scanf("%d", &secim)
    getchar()
    SWITCH secim

```


CASE 1

```

srand(time(NULL))
DECLARE rastgeleSatir = rand() % satirSayisi
print("Secilen rastgele kelime: %s", kelimelerDizisi[rastgeleSatir])
DECLARE anagram[6] = ""
print("Anagram kelimeyi girin: ")
fgets(anagram, sizeof(anagram), stdin)
DECLARE doubleHashCount = 0
DECLARE anagramBulundu = false
FOR i = 0 TO satirSayisi
    IF anagramMi(kelimelerDizisi[rastgeleSatir], anagram) THEN
        anagramBulundu = true
        IF strcmp(kelimelerDizisi[i], anagram) == 0 THEN
            dizin = hashleme(kelimelerDizisi[i], satirSayisi, boyut)
            DECLARE j=0
            WHILE (doubleHashSayisi[i][0]) != 0
                j++
            dizin = doubleHashleme(anagram, satirSayisi, j, boyut)
            (doubleHashSayisi[i][0])--
            ENDWHILE
            print("Girdiginiz kelimenin hash tablosundaki indeksi: %d\n", dizin-1)
            IF j==0 THEN
                printf("Cakisma yok.\n")
            ENDIF
            ELSE
                printf("Cakisma var: %d defa. \n", j)
            BREAK
        ENDIF
        IF i + 1 == satirSayisi THEN
            printf("Girdiginiz kelime hash tablosunda yok.\n")
            break
        ENDIF
    ENDIF
ENDFOR
IF anagramBulundu THEN
    print("Girdiginiz kelime anagram.\n")
ELSE
    print("Girdiginiz kelime anagram degil.\n")
ENDIF
BREAK

```

CASE 2

```

devamEt = false
BREAK

```

CASE 3

```
    print("Her oyunun basinda bu secim ekranini goreceksiniz bu cok sinir bozucu
biliyorum :) \n")
    print("Eger 1. secenegi secerseniz oyun baslar ve ekrana rastgele bir kelime
cikacak.\n")
    print("Sizden o kelimenin anagramini girmeniz bekleniyor.\n")
    print("Siz kelimeyi girdikten sonra sonuc ekranda gorunecek herhangi bir tusa
basarak devam edebilirsiniz.\n")
    print("Son olarak lutfen gidiğiniz kelimelerin hepsi buyuk harflerden olussun.\n")
    print("Artik hazirsiniz, herhangi bir tusa basip secim ekranina gidebilirsiniz.\n")
    print("Iyi eglenceler.")
    BREAK
DEFAULT
    print("Gecersiz secim!\n")
    BREAK
ENDSWITCH
clear_screen()
ENDWHILE
RETURN 0
ENDFUNCTION
```