

# **ALGORİTMA ANALİZİ**

## **Dönem Projesi**

**Konu: Kitap Öneri Sistemi**

**Ugur ALTINDAL      17011043**

# 1) Genel Açıklama :

Kodumun içerisinde de belirttiğim gibi son yapılan duyuruda bizimle paylaşılan hesaplama türlerinden 2.sini tercih ettim.

Proje dökümanında istenilen formüller ve sonrasında paylaşılan yeni çözümlerle ilgili, sizden gelen farklı mailler doğrultusunda, Öncelikle problemin 1. aşamasında Pearson benzerliği için:

Hesaplanan  $ra$  ve  $rb$  için, literatürde karşılığı olan iki farklı ortalama hesaplaması mevcuttur:

1)  $a$  ve  $b$  kullanıcılarının **ortak okuduğu** kitaplar için hesaplanan,  $ra$  ve  $rb$  ortalamaları.

2)  $a$  ve  $b$  kullanıcılarının **okuduğu** kitaplar için hesaplanan,  $ra$  ve  $rb$  ortalamaları.

Bu her iki durumda da doğru çözüme giden yaklaşımlar doğru kabul edilecektir.

Böylece  $a$  ve  $b$  kullanıcılarını karşılaştırırken okudukları kitaplar üzerinden hesaplama yaptım ve eğer 2'si de o kitabı OKUMADIYSA hesaba katmadım. İşlemlerimi bu doğrultuda yaptım.

```
for (k = 0; k < nKitap; k++) // Kitap sayımız 8. Kullanıcıların kitap puanlarına göre benzerliklerini burada alıyoruz.
{
    if(users[i].deger[k]!=0 || users[j].deger[k]!=0) // Son paylaşılan duyuruda 2 tane pearson hesaplama mantığı vardı.Ben 2. olanı ter
    { // Bu if sayesinde 2'sinin de okumadığı kitaplar için o kitabı hesaba katmıyoruz.
        topA += users[i].deger[k];
    }
}
```

Sorunun 2.şikkında ise şu yöntemi tercih ettim.

```
for (j = 0; j < nKisi; j++) // İstenilen kişi sayısı
{
    max = temp[0];
    index = 0;
    for (i = 1; i < nUser; i++)
    {
        if (max < temp[i] && i!=m) // SORUNUN 2.ŞIKKINDA İSTENDİĞİ GİBİ TÜM DEĞERLERİ SIRALAMADIM.
        { // İstenilen kişi sayısı kadar işlem yaptım.Diziyi sıralamış olsaydım sıralama algoritmalarına göre (n^2) veya (n log n) gibi değerlerle bu işlemi yapacaktım.
            max = temp[i]; // Fakat burada örneğin kişi sayısı 3 ise O(n)*3 ile işlemi yapıyorum ve zaman karmaşıklığım bu yüzden O(n) ile max benzerlikleri elde ediyorum.
            index = i;
        }
    }
    users[m].mostSim[j] = index;
    temp[index]=-1.0;
}
```

Tüm diziyi sıralamak yerine kaç eleman isteniyorsa (ödevde3) o kadar max değerini bulup geçici diziyeye attım. Yani bu durumu  $O(n)*3 = O(n)$  time complexity ile çözdüm.

**Ek olarak deneyen/kontrol eden hocamızın programın çalışmasını kendi test etmek isterse diye tek tek elle değer girmesi saçma olacağı için kullanıcı kolaylığı için RAR içerisine koyduğum puanlar.txt dosyasından Kullanıcı/Kitap puan listesi otomatik olarak çekiliyor. Kullanıcıya sadece kitapların ismini girmek kalıyor.**

C:\Users\TULPAR\Desktop\Okul 2020\Dersler\Algoritma Analizi\devler\AlgAnaliziProje\17011043.exe

Sirasiyla okucuyularin okuduklari kitaplari giriniz..

```
1.Kitap Adi : TRUE BELIEVER
2.Kitap Adi : THE DA VINCI CODE
3.Kitap Adi : THE WORLD IS FLAT
4.Kitap Adi : MY LIFE SO FAR
5.Kitap Adi : THE TAKING
6.Kitap Adi : THE KITE RUNNER
7.Kitap Adi : RUNNY BABBIT
8.Kitap Adi : HARRY POTTER
```

Dosya basariyla acildi.

25 adet kullanıcı bilgisi dosyadan cekildi..

**Programımın önceki halinde hangi User'ın kontrol edileceği programın kullanıcı tarafından input olarak giriliyordu fakat ödevde tablo halinde NU1-NU5 değerleri istendiği için onu da otomatik bir şekilde aşağıdaki gibi yaptım.**

```
for(m=21;m<26;m++)
{
    mostSims(users,nUser,m-1);
    printf("\n\n");
    printf("\n\t(NU%d)Kullanicisinin EN BENZER oldugu 3 kisi : ",indx);

    for(j=0;j<kKisi;j++)
    {
        printf(" (U%d) ",users[m-1].mostSim[j]+1);
    }

    |
    prediction(users,m-1,kitaplar,indx);
    indx++;
    sleep(1);
}
```

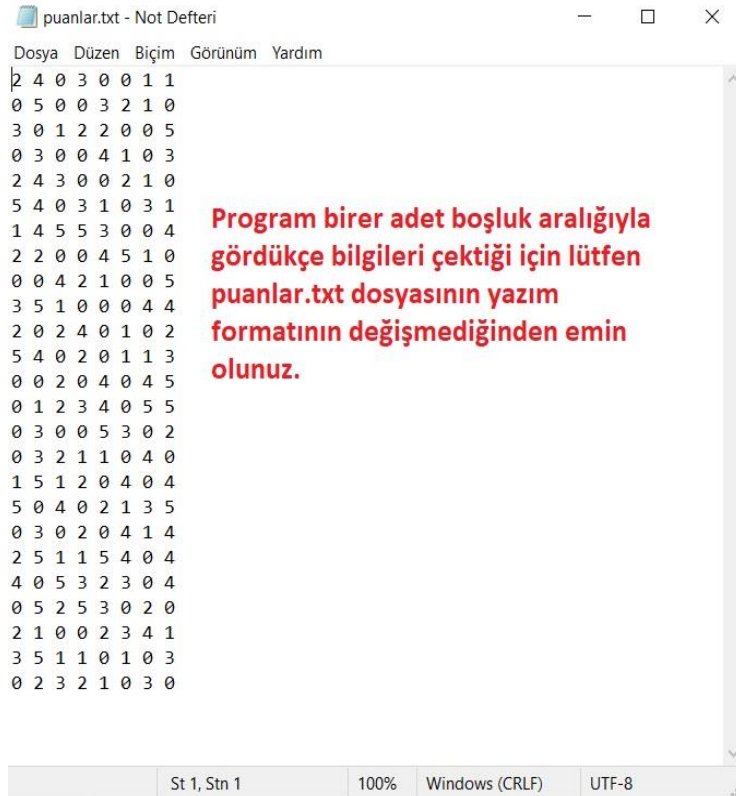
**Özet olarak program kullanıcısının yapması gerekenler**

**1) puanlar.txt dosyasının 17011043.c uzantılı kodla aynı dosyada bulunması gerekiyor.**

**2) Sadece kitapların ismini girmesi gerekiyor.**

**3) Ekstra testler için farklı değerler girmek yerine dilerseniz puanlar.txt dosyasından puanların değerlerini sırasını değiştirebilirsiniz. Değiştirdikten sonra lütfen her index arasında sadece 1 adet boşluk olduğundan emin olunuz.**

**Teşekkür ederim.**



puanlar.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
2 4 0 3 0 0 1 1
0 5 0 0 3 2 1 0
3 0 1 2 2 0 0 5
0 3 0 0 4 1 0 3
2 4 3 0 0 2 1 0
5 4 0 3 1 0 3 1
1 4 5 5 3 0 0 4
2 2 0 0 4 5 1 0
0 0 4 2 1 0 0 5
3 5 1 0 0 0 4 4
2 0 2 4 0 1 0 2
5 4 0 2 0 1 1 3
0 0 2 0 4 0 4 5
0 1 2 3 4 0 5 5
0 3 0 0 5 3 0 2
0 3 2 1 1 0 4 0
1 5 1 2 0 4 0 4
5 0 4 0 2 1 3 5
0 3 0 2 0 4 1 4
2 5 1 1 5 4 0 4
4 0 5 3 2 3 0 4
0 5 2 5 3 0 2 0
2 1 0 0 2 3 4 1
3 5 1 1 0 1 0 3
0 2 3 2 1 0 3 0
```

**Program birer adet boşluk aralığıyla gördükçe bilgileri çektiği için lütfen puanlar.txt dosyasının yazım formatının değişmediğinden emin olunuz.**

St 1, Stn 1 100% Windows (CRLF) UTF-8

## 2) KOD

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <unistd.h>

#define nKitap 8

#define kKisi 3

typedef struct kullanicilar{

    int deger[10];

    float benzerlik[30];

    int mostSim[3];

}USER;

void coefficient(USER *users,int nUser)

{

    int i,j,k;

    int n;

    int topA,topB,topAB;

    int sqrTop_a, sqrTop_b;

    float corr;

    for(i=0; i < nUser-1; i++) // Tüm kullanıcılar için yapılıyor.

    {

        for(j=i+1; j < nUser; j++) // j=i+1 deme sebebimiz.U1 ve U15 değeri çoktan karşılaştırılmış

        olacak.

        {

            // Bu sayede efektif bir şekilde

            U15'in benzerlik değerlerine bakarken kontrole U16'dan başlayıp son değere kadar gidiyoruz.

            topA = 0, topB = 0, topAB = 0,sqrTop_a = 0,sqrTop_b = 0;

            n=nKitap;
```

for (k = 0; k < nKitap; k++) // Kitap sayımız 8. Kullanıcıların kitap puanlarına göre benzerliklerini burada alıyoruz.

{

if(users[i].deger[k]!=0 || users[j].deger[k]!=0) // Son paylaşılan duyuruda 2 tane pearson hesaplama mantığı vardı.Ben 2. olanı tercih ettim yani 2 kullanıcı da bir kitabı okumadıysa o kitabı hesaba katmıyorum.

{

// Bu if sayesinde 2'sinin de okumadığı kitaplar için o kitabı hesaba katmıyoruz.

topA += users[i].deger[k];

topB += users[j].deger[k];

topAB += users[i].deger[k] \* users[j].deger[k]; // IF'e girdiysek gerekli işlemleri yapıyoruz.Adım adım toplama ve karekok işlemleri dahil.

sqrTop\_a += users[i].deger[k] \* users[i].deger[k];

sqrTop\_b += users[j].deger[k] \* users[j].deger[k];

}

else

{

n--;

// IF'e girmediyse demek ki o

kitabı 2 kullanıcı da okumamış demektir.Bu kitabı hesaba katmadığımız için bu 2 kullanıcının karşılaştırmasında kitap sayısı değerini 1 eksilttik.

}

}

float corr = (float)(n \* topAB - topA \* topB) / sqrt((n \* sqrTop\_a - topA \* topA) \* (n \* sqrTop\_b - topB \* topB));

users[i].benzerlik[j]=corr; // Benzerlik değerlerini struct yapısında sakladık.

users[j].benzerlik[i]=corr; // Karşılaştırılan kullanıcıların benzerlik değerlerini struct yapısındaki o kullanıcıların benzerlik arraylerine kaybettik.

}

}

```
}
```

```
int input(USER *users,char kitaplar[][30]) // Kullanıcı/Puanlar tablosunun kullanıcı/kontrol eden  
kolaylığı için puanlar.txt dosyasından otomatik olarak çekimi..
```

```
{
```

```
    int i=0;
```

```
    FILE *data=fopen("puanlar.txt","r");
```

```
    if(data==NULL)
```

```
    {
```

```
        printf("\n\n\tLutfen deneme yaparken degerleri tek tek girmemek icin rar icerisindeki  
puanlar.txt dosyasinin .c uzantili kodla ayni konumda oldugundan emin olunuz..");
```

```
    }
```

```
    else
```

```
    {
```

```
        printf("\n\n\tDosya basariyla acildi.");
```

```
    }
```

```
    while(!feof(data)) // Programın sonuna kadar gider ve degerleri satir satir almis olur.
```

```
    {
```

```
        fscanf(data,"%d %d %d %d %d %d %d %d  
%d",&users[i].deger[0],&users[i].deger[1],&users[i].deger[2],&users[i].deger[3],&users[i].deger[4],&users[i].de  
ger[5],&users[i].deger[6],&users[i].deger[7]);
```

```
        i++;
```

```
    }
```

```
    // Degerleri tek tek elle girip surekli denemek zor olacagi icin dosyadan cekmeyi dusundum.
```

```
    // Bu sayede 25 kullanıcı ve 8 kitap icin 200 girdi girmek yerine size atacagım dosya üzerinden ödevde  
istenilen çıktıları siz de görebilirsiniz.
```

```
    return i; // dosyanin kac satir indigine gore kullanici sayimizi elde ettik.
```

```
}
```

```
void prediction(USER *users,int m,char kitaplar[][30],int indx)
```

```
{
```

```
    float topAranan;
```

```

float topAranan2;

float pay;

int i,j,k;

float topSim=0.0;

int temp=8;

int okunmayan[8];

int count=0;

float puanlar[8];

printf("\n\n\t(NU%d) Kullanicisinin OKUMADIGI kitaplar   : ",indx);

for(i=0;i<kKisi;i++)

{

    topSim+=users[m].benzerlik[users[m].mostSim[i]];           // Benzerlik deęerlerinin
    toplamına bۆleceęimiz iin ilk bařta benzerlik deęerleri toplamını bulduk.

}

for(i=0;i<nKitap;i++)

{

    if(users[m].deger[i]==0)                                   // Okunmayan kitaplar hesaba katılmayacak

    {

        okunmayan[count]=i;                                   // Okunmayan kitapların index
        deęerlerini geici bir diziye attık.

        count++;

        printf("(%)s  ",kitaplar[i]); // Aranan kullanıcının hangi kitapları okumadığını yazdırdık.

        temp--;

    }

    else

    {

        topAranan+=users[m].deger[i];

    }

}

topAranan=topAranan/temp;                                     // Aramak istedigimiz RN(Nu1) iin
ortalama puan deęerini elde ettik.

int temp2=8;

```



```
printf("\n\n\t(NU%d) Kullanicisi icin puan tahminleri : ",indx); // Kullanıcının okumadığı  
kitapların hepsi için puan tahminlerini yazdırdık.
```

```
for(i=0;i<count;i++)
```

```
{
```

```
    pay=0;
```

```
    for(j=0;j<kKisi;j++)
```

```
    {
```

```
        temp2=8;
```

```
        topAranan2=0;
```

```
        for(k=0;k<nKitap;k++)
```

```
        {
```

```
            if(users[users[m].mostSim[j]].deger[k]==0)
```

```
            {
```

```
                temp2--;
```

```
            }
```

```
            else
```

```
            {
```

```
                topAranan2+=users[users[m].mostSim[j]].deger[k];
```

```
            }
```

```
        }
```

```
        topAranan2=topAranan2/temp2; // Burası Aranan Kullanıcının en benzer  
olduğu kullanıcıların ortalama kitap puan değerlerinin hesaplanıp aralarındaki benzerlikle çarpıldığı kısım.
```

```
        topAranan2=users[users[m].mostSim[j]].deger[okunmayan[i]]-topAranan2;
```

```
        pay+=topAranan2*users[m].benzerlik[users[m].mostSim[j]]; // Pay kısmı += ile en  
yakın 3 kişi istendiği için toplanarak devam ediyor.
```

```
    }
```

```
    float sonuc=topAranan+(pay/topSim); // En son topAranan = rA pay =  
benzerlik* (rBkitap-rBort) topSim=En başta bulduğumuz en benzer 3 değer için toplamBenzerlik değeri.
```

```
    if(sonuc<0)
```

```
    {
```

```

        sonuc=0; // Testlerime göre ve Kullanıcı/Kitap puan listesinde genelde en benzer kullanıcıların aynı kitapları okumamasıyla oluştuğu için
    } //
    Çünkü 2'si de o kitabı okumadıysa o benzerlik hesaplamasında hesaba katılmıyor.

    printf(" (%s) Tahmini Puan --> (%1.2f) ",kitaplar[okunmayan[i]],sonuc); // Okunmayan her kitap için tahmini puanlar

    puanlar[okunmayan[i]]=sonuc; // Puanlar geçici dizisine puanları atarak orada en yüksek puanı bulup hangi kitap olduğunu kullanıcıya önereceğiz.
}

```

```

float max=puanlar[0];
int index;
for(i=0;i<nKitap;i++)
{
    if(max<puanlar[i])
    {
        max=puanlar[i]; // En yüksek puana sahip kitabın bulunması
        index=i;
    }
}

```

```

printf("\n\n\t(NU%d) Kullanicisi icin ONERILEN kitap : (%s) Tahmini Puan --> (%1.2f) \n",indx,kitaplar[index],max); // En yüksek puana sahip kitabın önerilmesi ve puanının yazdırılması
}

```

```

void mostSims(USER *users,int nUser,int m) // En benzer kullanıcıları kaydettiğimiz fonksiyon
{
    int i,j;
    float max;
    int index;
    float temp[30];
    int count=0;

```

```

        for(i=0; i < nUser; i++)
        {
            if(i!=m)
            {
                temp[i]=users[m].benzerlik[i];           // Benzerlik değerlerini önce geçici diziye
attık kontrol için ve users listemizi bozmamak için.

                count++;

                if(count==8)
                {
                    count=0;
                }
            }
        }

        for (j = 0; j < kKisi; j++)                       // İstenilen kişi sayısı
        {
            max = temp[0];

            index = 0;

            for (i = 1; i < nUser; i++)
            {
                if (max < temp[i] && i!=m)                 // SORUNUN 2.ŞIKKINDA İSTENDİĞİ GİBİ TÜM DEĞERLERİ
SIRALAMADIM.

                {
                    //
İstenilen kişi sayısı kadar işlem yaptım.Diziyi sıralamış olsaydım sıralama algoritmalarına göre (n^2) veya (n log
n) gibi değerlerle bu işlemi yapacaktım.

                    max = temp[i];                       // Fakat burada örneğin kişi sayısı 3 ise O(n)*3 ile
işlemi yapıyorum ve zaman karmaşıklığım bu yüzden O(n) ile max benzerlikleri elde ediyorum.

                    index = i;
                }
            }

            users[m].mostSim[j] = index;

            temp[index]=-1.0;
        }

```

```

}

int main()
{
    int nUser;                // Kullanıcı sayısı
    char kitaplar[10][30];    // Kitapların olduğu dizi
    int i,j,m;
    USER users[30];          // Kullanıcı listesi
    char secim;
    int indx=1;
    printf("\n\t Sirasiyla okucuyularin okuduklari kitaplari giriniz.. \n\n");
    for(j=0;j < nKitap; j++)
    {
        printf("\n %d.Kitap Adi : ",j+1);
        gets(kitaplar[j]);
    }
    nUser=input(users,kitaplar);
    printf("\n\n\t %d adet kullanici bilgisi dosyadan cekildi.. \n\n",nUser);
    coefficient(users,nUser);
    for(m=21;m<26;m++)
    {
        mostSims(users,nUser,m-1);
        printf("\n\n");
        printf("\n\t (NU%d)Kullanicisinin EN BENZER oldugu 3 kisi : ",indx);

        for(j=0;j<kKisi;j++)
        {
            printf(" (U%d) ",users[m-1].mostSim[j]+1);
        }

        prediction(users,m-1,kitaplar,indx);
        indx++;
        sleep(1);
    }
}

```

```
printf("\n\n\n\n\t\t\t\t\t Program Sonlandırılıyor...\n\n");

sleep(2);


return 0;

}
```

### 3) EKRAN ÇIKTISI

Users\TULPAR\Desktop\Okul 2020\Dersler\Algoritma Analizi\devler\AlgAnaliziProje\17011043.exe



```
(NU1)Kullanıcısının EN BENZER olduğu 3 kişi : (U9) (U18) (U11)
(NU1) Kullanıcısının OKUMADIGI kitaplar : (THE DA VINCI CODE) (RUNNY BABBIT)
(NU1) Kullanıcisi için puan tahminleri : (THE DA VINCI CODE) Tahmini Puan --> (0.63) (RUNNY BABBIT) Tahmini Puan --> (1.49)
(NU1) Kullanıcisi için ONERILEN kitap : (RUNNY BABBIT) Tahmini Puan --> (1.49)

(NU2)Kullanıcısının EN BENZER olduğu 3 kişi : (U1) (U7) (U2)
(NU2) Kullanıcısının OKUMADIGI kitaplar : (TRUE BELIEVER) (THE KITE RUNNER) (HARRY POTTER)
(NU2) Kullanıcisi için puan tahminleri : (TRUE BELIEVER) Tahmini Puan --> (1.81) (THE KITE RUNNER) Tahmini Puan --> (0.98) (HARRY POTTER) Tahmini Puan --> (2.42)
(NU2) Kullanıcisi için ONERILEN kitap : (HARRY POTTER) Tahmini Puan --> (2.42)

(NU3)Kullanıcısının EN BENZER olduğu 3 kişi : (U8) (U16) (U14)
(NU3) Kullanıcısının OKUMADIGI kitaplar : (THE WORLD IS FLAT) (MY LIFE SO FAR)
(NU3) Kullanıcisi için puan tahminleri : (THE WORLD IS FLAT) Tahmini Puan --> (0.61) (MY LIFE SO FAR) Tahmini Puan --> (0.35)
(NU3) Kullanıcisi için ONERILEN kitap : (THE WORLD IS FLAT) Tahmini Puan --> (0.61)

(NU4)Kullanıcısının EN BENZER olduğu 3 kişi : (U12) (U1) (U10)
(NU4) Kullanıcısının OKUMADIGI kitaplar : (THE TAKING) (RUNNY BABBIT)
(NU4) Kullanıcisi için puan tahminleri : (THE TAKING) Tahmini Puan --> (0.00) (RUNNY BABBIT) Tahmini Puan --> (1.48)
(NU4) Kullanıcisi için ONERILEN kitap : (RUNNY BABBIT) Tahmini Puan --> (1.48)

(NU5)Kullanıcısının EN BENZER olduğu 3 kişi : (U16) (U5) (U7)
(NU5) Kullanıcısının OKUMADIGI kitaplar : (TRUE BELIEVER) (THE KITE RUNNER) (HARRY POTTER)
(NU5) Kullanıcisi için puan tahminleri : (TRUE BELIEVER) Tahmini Puan --> (0.16) (THE KITE RUNNER) Tahmini Puan --> (0.05) (HARRY POTTER) Tahmini Puan --> (0.27)
(NU5) Kullanıcisi için ONERILEN kitap : (HARRY POTTER) Tahmini Puan --> (0.27)
```

- Değerler gelebildiği için tahmini puan <0 kontrolü yaptım. 0'dan küçükse 0 puan tahmin ediyor

**Hocam ödevimi ayrıca internet üzerindeki  
calculatorlar üzerinden test ettim ve  
hesaplarımı teyit ederek  
ilerledim.Puanlar.txt içerisindeki  
değerleri değıştirsenez bile programın her  
değer için doğru çalıştığını görebilirsiniz.**