ALGORİTMA ANALİZİ

DÖNEM PROJESİ - GRUP 1

KONU : KİTAP ÖNERİ SİSTEMİ

İrem ATILGAN

17061036

03.01.2021

1. FONKSİYON BİLGİLERİ

1. **void printTable(recTable* rt)**
   CSV Tablosu içeriğini yazdıran fonksiyondur. Parametre olarak tablonun tutulduğu recTable struct'ını alır.

2. **int searchUser(char* userno,recTable* rt)**
   CSV Tablosunda kullanıcının yer aldığı satır indisini bulan fonksiyon. Parametre olarak aranan kullanıcının ismi ve tablo verilir; Kullanıcının indisi dönülür.

3. **float calculateMean(recTable* rt, int ind)**
   Verilen indisteki kullanıcının kitaplara verdiği oyların ortalamasını hesaplayan fonksiyon. CSV tablosunu ve kullanıcı indisini parametre olarak alır; hesaplanan ortalamayı döner.

4. **float calculateSimilarity(recTable* rt, char* uno1, char* uno2)**
   İki kullanıcı arasındaki benzerlik puanını hesaplayan fonksiyon. Burada, ödevde verilen Pearson katsayısı formülü kullanılarak hesaplama yapılır (A ve B kullanıcılarının oylarının ortalamaları hesaplanırken ortak okudukları kitaplara bakılmaz.). Sonuç olarak benzerlik puanı döner.

5. **node* createNode(float val, char* userno)**
   node struct'ının oluşturulduğu ve niteliklerinin atandığı fonksiyon. Fonksiyon, oluşturulan node struct'ını döner.

6. **char* recommendBook(recTable* rt, char* userno, node* head, int k)**
   En çok benzerliği bulunan kullanıcılara bağlı olarak, verili bir kullanıcının oylamadığı kitaplar için tahmini oyların hesaplandığı, tüm kitap ve oyların sıralanarak en yüksek oya sahip kitabın önerildiği fonksiyondur. Parametre olarak CSV tablosunu, kullanıcı adını, önerilen K kullanıcıyı ve k sayısını alır. En yüksek tahmini puana sahip kitabın adını döner.

7. **node* recommendKUsers(recTable* rt, int k, char* userno)**
   Verilen bir kullanıcı için en benzer K kullanıcıyı bulan fonksiyon. Parametre olarak CSV tablosunun tutulduğu struct'ı, benzer k kullanıcı için "k" miktarını ve benzerliği aranan kullanıcının adını alır.

2. PROGRAM KODU

1. recTable ve node Struct'ı

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>


//CSV dosyasi ile ilgili bilgilerin tutuldugu struct
typedef struct recTable{
    char** users;   //kullanici isimlerinin tutuldugu dizi
    char** books;   //kitap isimlerinin tutuldugu dizi
    int** votes;    //kullanicilarin kitaplar icin verdikleri oylarin tutuldugu matris
    int row;    //kullanici sayisi
    int col;    //kitap sayisi
}recTable;

//benzer kullanicilarin veya tahminlemesi yapilan oylarin tutuldugu struct
//en yuksekten dusuge siralama yapilmasi icin linked-list yapisi kullanilmistir
typedef struct node{
    float val;      //hucre degeri (benzerlik/tahmini oy)
    char* name;     //benzeyen kullanici adi/oylanan kitap ismi
    struct node* next;
}node;
```

2. searchUser & calculateMean

```c
//Kullanicinin isminden bulundugu indisi bulan fonksiyon
int searchUser(char* userno,recTable* rt)
{
    int i;
    //kullanici adi bulunana veya tum kullanici dizisi gezilene kadar
    for(i = 0; strcmp(rt->users[i],userno) && (i < rt->row); i++);
    //kullanici bulunamazsa -1 donulur
    if(i >= rt->row)
        return -1;
    //bulunursa indisi donulur
    return i;

}

//Kullanicinin bir kitap icin verdigi oylarin ortalamasinni hesaplayan fonksiyon
float calculateMean(recTable* rt, int ind)
{
    int i;
    float total = 0;
    float m;
    for(i = 0; i < rt->col; i++)
    {
        total += rt->votes[ind][i]; //Oylar toplanir
    }
    m = total/(float)i; //Ortalamasi alinir

    return m;
}
```

3. calculateSimilarity

```
79  //Iki kullanicinin benzerligini hesaplayan fonksiyon
80  float calculateSimilarity(recTable* rt, char* uno1, char* uno2)
81  {
82      int i;
83      int j;
84      int ind1, ind2;
85      //Oylarin toplaminin tutuldugu degiskenler
86      float total1 = 0;
87      float total2 = 0;
88      int vote1,vote2;    //Iki kullanicinin verdigi oy
89      float mean1,mean2;  //Iki kullanicinin kitaplar icin verdigi oylarin ortalamasi
90      float sq1,sq2;
91      float similarity;   //Pearson korelasyon formulu sonucu
92
93      //Ilk iki kullanicinin csv matrisindeki konumları bulunur
94      ind1 = searchUser(uno1,rt);
95      ind2 = searchUser(uno2,rt);
96
97      for(j = 0; j < rt->col; j++)
98      {
99          vote1 = rt->votes[ind1][j];
100         vote2 = rt->votes[ind2][j];
101         total1 += vote1;
102         total2 += vote2;
103     }
104     mean1 = (float)total1/(rt->col);
105     mean2 = (float)total2/(rt->col);
106
107     total1 = 0;
108     sq1 = 0;
109     sq2 = 0;
```

```
110     for(j = 0; j < rt->col; j++)
111     {
112         vote1 = rt->votes[ind1][j];
113         vote2 = rt->votes[ind2][j];
114         if(vote1 && vote2) //iki kisi de kitaba oy verdiyse Person formulu icin gerekli islemler uygulanir
115         {
116             total1 += (vote1-mean1)*(vote2-mean2);
117             sq1 += pow((vote1-mean1),2);
118             sq2 += pow((vote2-mean2),2);
119         }
120     }
121     similarity = (total1)/(sqrt(sq1)*sqrt(sq2));    //benzerlik orani hesaplanir
122     return similarity;
123 }
```

4. createNode

```
125 //node struct'ı initialize etme fonksiyonu
126 node* createNode(float val, char* userno)
127 {
128     node* newNode = (node*)malloc(sizeof(node));
129     newNode->name = (char*)malloc(sizeof(char)*20);
130     strcpy(newNode->name,userno);   //verilen kullanici adi eklenir
131     newNode->val = val;
132     newNode->next = NULL;
133     return newNode;
134 }
```

5. recommendBook

```
136  //Kullaniciya kitap onerme fonksiyonu
137  char* recommendBook(recTable* rt, char* userno, node* head, int k)
138  {
139      int i;
140      int j;
141      //Onerilecek kitaplarin en yuksek oydan en az oya siralandigi linkli liste
142      node* root_books = NULL;
143
144      //Oneri yapilacak kullanicinin kullanici dizisinde yeri bulunur
145      int userind = searchUser(userno,rt);
146
147      //Kullanicinin kitaplar icin verdigi oylarin ortalamasi bulunur
148      float usermean = calculateMean(rt,userind);
149
150      //Kitap linkli listesi icinde gezebilmek icin olusturulan node* degiskeni
151      node* iterator;
152
153      //Benzerligi bulunan kullanicilarin verdigi oylarin ortalamasini tutan degisken
154      float mean2;
155
156      //Benzerligi bulunan kullanicilarin indislerini tutan degisken
157      int ind2;
158      //Odevde verili oy tahminleme formulunun pay ve payda bolumleri
159      float pay = 0;
160      float payda = 0;
161
162      float pred; //Tahmin sonucunun tutuldugu degisken
163
164      node* tmp; //Benzer kullanici linkli listesi icinde gezebilmek icin olusturulan node* degiskeni
```

```
166      for(i = 0; i < rt->col; i++)
167      {
168          //Kullanicinin oy vermedigi kitap bulunursa
169          if(rt->votes[userind][i] == 0)
170          {
171              tmp = head;
172              pay = 0;
173              payda = 0;
174              j = 0;
175              //Benzer kullanicilarin verdigi oylar kullanilir
176              while(tmp != NULL && j < k)
177              {
178                  ind2 = searchUser(tmp->name,rt);
179                  mean2 = calculateMean(rt,ind2);
180                  pay = pay + (tmp->val)*(float)(rt->votes[ind2][i]-mean2);
181                  payda += (tmp->val);
182                  tmp = tmp->next;
183                  j++;
184              }
185              //Tahmini oy hesaplanir
186              pred = usermean + (pay/payda);
187              //Eger linkli listeye hic kitap yerlestirilmediyse
188              if(root_books == NULL)
189              {
190                  //root iniialize edilir ve tahmini oy degeri yerlestirilir
191                  root_books = (node*)malloc(sizeof(node));
192                  root_books->name = (char*)malloc(sizeof(char)*20);
193                  strcpy(root_books->name,rt->books[i]);
194                  root_books->val = pred;
195                  root_books->next = NULL;
196              }
```

```
            else //Liste daha once olusturulmussa
            {
                //Tahmini deger, en yuksek oy degerinden buyukse basa yerlestirilir
                if(pred > root_books->val)
                {
                    node* newRec = createNode(pred,rt->books[i]);
                    newRec->next = root_books;
                    root_books = newRec;
                }
                else //Degilse linkli liste gezilerek buyuk oldugu noktaya yerlestirilir
                {
                    iterator = root_books;
                    while(iterator->next != NULL && iterator->next->val > pred)
                        iterator = iterator->next;
                    node* newRec = createNode(pred,rt->books[i]);
                    newRec->next = iterator->next;
                    iterator->next = newRec;
                }
            }


        }

    }
    printf("%s kullanicisinda okunmamis olan kitaplar icin hesaplanan tahmini begenme degerleri:\n",userno);
    iterator = root_books;
    i = 1;
    while(iterator != NULL)
    {
        printf("%d. %s, %f",i,iterator->name,iterator->val);
        iterator = iterator->next;
        printf("\n");
        i++;
    }
    printf("Sonuc olarak onerilen kitap : %s\n\n",root_books->name);
    return root_books;
}
```

6. recommendKUsers

```
//Verilen kullaniciya benzer K kullanicinin onerildigi fonksiyon
node* recommendKUsers(recTable* rt, int k, char* userno)
{
    int rows = rt->row;
    int i = 0;
    float sim; //Benzerlik miktarinin tutuldugu fonksiyon
    node* head = (node*)malloc(sizeof(node));
    node* newNode;
    node* iterator = head;

    //Verilen kullanici disinda bir kullanici bulunana dek kullanici dizisi gezilir
    while(!strcmp(rt->users[i],userno)) i++;
    //Benzer kullanicilarin tutuldugu linkli listenin head'i olusturulur
    if(strcmp(rt->users[i],userno))
    {
        //Verilen kullanici ile listedeki kullanici arasindaki benzerlik hesaplanir
        sim = calculateSimilarity(rt,userno,rt->users[i]);
        head->val = sim;
        head->name = (char*)malloc(sizeof(char)*10);
        strcpy(head->name,rt->users[i]);
        i = 0;

    }
```

```
    //'N' ile baslayan kullanici isimlerine kadar olan kullanicilar alinir
    while(rt->users[i][0] != 'N' && i < rows)
    {
        //Varolan kullanici, listedeki kullanici degilse
        if(strcmp(rt->users[i],userno))
        {
            //Benzerlik hesaplanir
            sim = calculateSimilarity(rt,userno,rt->users[i]);
            //Benzerlik degeri en buyuk benzerlik degerinden kucukse
            if(sim < head->val)
            {
                //Kendinden kucuk benzerlik degerine sahip bir node bulunana dek linkli liste gezilir
                iterator = head;
                while(iterator->next != NULL && iterator->next->val > sim)
                    iterator = iterator->next;

                newNode = createNode(sim,rt->users[i]);
                newNode->next = iterator->next;
                iterator->next = newNode;

            }
            else //Benzerlik degeri, en buyuk benzerlik degerinden buyukse
            {
                //linkli listenin basina yerlestirilir
                newNode = createNode(sim,rt->users[i]);
                newNode->next = head;
                head = newNode;
            }

        }
        i++;
    }

    printf("%s kullanicisina en yakin kullanicilar(k=%d) ve hesaplanan pearson benzerlikleri sirasiyla,\n",userno,k);
    iterator = head;
    i = 0;
    while(iterator != NULL && i < k)
    {
        printf("%d. %s (%f)\n",i+1,iterator->name,iterator->val);
        iterator = iterator->next;
        i++;
    }
    printf("\n");
    return head;
}
```

7. main

```
int main()
{
    char line[1024];
    char* tmp;
    char* word = (char*)malloc(sizeof(char)*20);
    char* token;
    int num_books = 0;
    int num_users = 0;
    int i;

    FILE* stream = fopen("recdataset.csv", "r"); //csv dosyasi okunur
    recTable* table = malloc(sizeof(recTable)); //csv icerigining tutuldugu tablo olusturulur
    table->books = (char**)calloc(15,sizeof(char*)); //tablodaki kitaplar icin yer ayrilir

    //Kitap isimleri icin yer ayrilir
    for(i = 0; i < 15; i++)
    {
        table->books[i] = (char*)calloc(20, sizeof(char));
    }
    //Kullanici isimlerinin tutuldugu string dizisi olusturulur ve yer ayrilir
    table->users = (char**)calloc(50,sizeof(char*));
    for(i = 0; i < 50; i++)
    {
        table->users[i] = (char*)calloc(10, sizeof(char));
    }

    //KITAP BASLIKLARI ALINIR
    i = 0;
    fgets(line, 1024, stream);
    tmp = strdup(line);
    word = strchr(tmp,'\n');
    if(word)
        *word = 0;
    token = strtok(tmp,";");
    token = strtok(NULL,";");

    while(token)
    {
        strcpy(table->books[i++],token);
        num_books += 1;
        token = strtok(NULL,";");
    }
    table->books[i] = '\0';
    table->col = num_books;
    printf("\n");

    //KULLANICI ADI VE KULLANICI OYLARI ALINIR
    int counter;
    int val;
    char* num = malloc(sizeof(char)*5);
    int ind_letter = 0;

    table->votes = (int**)calloc(50,sizeof(int*));
    for(i = 0; i < 50; i++)
    {
        table->votes[i] = (int*)calloc(num_books,sizeof(int));
    }

    while (fgets(line, 1024, stream))
    {
        i = 0;
        num_users++;
        counter = 0;
        while(line[i] != ';')
        {
            word[ind_letter++] = line[i++];
        }
        word[ind_letter] = '\0';
        strcpy(table->users[(num_users-1)], word);
        ind_letter = 0;
        i++;
```

```
        while(line[i] != '\n')
        {
            if(line[i] == ';' || line[i] == " ")
            {
                num[ind_letter] = '\0';

                if(ind_letter == 0)
                {
                    val = 0;
                    ind_letter = 0;
                }
                else
                {
                    val = atoi(num);
                    ind_letter = 0;
                }

                table->votes[(num_users-1)][counter++] = val;
            }
            else
                num[ind_letter++] = line[i];

            i++;
        }

        num[ind_letter] = '\0';
        if(ind_letter == 0)
        {
            val = 0;
            ind_letter = 0;
        }
        else
        {
            val = atoi(num);
            ind_letter = 0;
        }
        table->votes[(num_users-1)][counter] = val;
    }
```

```
    //Tablodaki kullanici ve kitap sayisi tablo struct'ında tutulur
    table->col = counter+1;
    table->row = num_users;

    printf("Kitap onerisi yapilacak kullanici adi : ");
    scanf("%s",username);
    printf("Benzer kullanici sayisi(k): ");
    scanf("%d",&k);


    node* list = recommendKUsers(table,k,username);
    recommendBook(table,username,list,k);

    return 0;
}
```

3. EKRAN GÖRÜNTÜLERİ

```
NU1 kullanicisina en yakin kullanicilar(k=3) ve hesaplanan pearson benzerlikleri sirasiyla,
1. U5 (0.966098)
2. U9 (0.908952)
3. U19 (0.857015)

NU1 kullanicisinda okunmamis olan kitaplar icin hesaplanan tahmini begenme degerleri:
1. THE DA VINCI CODE, 3.402100
2. RUNNY BABBIT, 1.713880
Sonuc olarak onerilen kitap : THE DA VINCI CODE

NU2 kullanicisina en yakin kullanicilar(k=3) ve hesaplanan pearson benzerlikleri sirasiyla,
1. U2 (0.990737)
2. U1 (0.969564)
3. U1 (0.969564)

NU2 kullanicisinda okunmamis olan kitaplar icin hesaplanan tahmini begenme degerleri:
1. TRUE BELIEVER, 2.073698
2. THE KITE RUNNER, 1.426302
3. HARRY POTTER, 1.411849
Sonuc olarak onerilen kitap : TRUE BELIEVER

NU3 kullanicisina en yakin kullanicilar(k=3) ve hesaplanan pearson benzerlikleri sirasiyla,
1. U16 (0.657263)
2. U14 (0.556223)
3. U13 (0.361217)

NU3 kullanicisinda okunmamis olan kitaplar icin hesaplanan tahmini begenme degerleri:
1. THE WORLD IS FLAT, 1.737929
2. MY LIFE SO FAR, 1.214990
Sonuc olarak onerilen kitap : THE WORLD IS FLAT

NU4 kullanicisina en yakin kullanicilar(k=3) ve hesaplanan pearson benzerlikleri sirasiyla,
1. U10 (0.971165)
2. U2 (0.922018)
3. U4 (0.915574)

NU4 kullanicisinda okunmamis olan kitaplar icin hesaplanan tahmini begenme degerleri:
1. THE TAKING, 2.404359
2. RUNNY BABBIT, 1.826996
Sonuc olarak onerilen kitap : THE TAKING

NU5 kullanicisina en yakin kullanicilar(k=3) ve hesaplanan pearson benzerlikleri sirasiyla,
1. U9 (0.986036)
2. U18 (0.890229)
3. U7 (0.888733)

NU5 kullanicisinda okunmamis olan kitaplar icin hesaplanan tahmini begenme degerleri:
1. HARRY POTTER, 3.829835
2. TRUE BELIEVER, 1.082499
3. THE KITE RUNNER, -0.526778
Sonuc olarak onerilen kitap : HARRY POTTER
```

```
Kitap onerisi yapilacak kullanici adi : NU1
Benzer kullanici sayisi(k): 3
NU1 kullanicisina en yakin kullanicilar(k=3) ve hesaplanan pearson benzerlikleri sirasiyla,
1. U5 (0.966098)
2. U9 (0.908952)
3. U19 (0.857015)

NU1 kullanicisinda okunmamis olan kitaplar icin hesaplanan tahmini begenme degerleri:
1. THE DA VINCI CODE, 3.402100
2. RUNNY BABBIT, 1.713880
Sonuc olarak onerilen kitap : THE DA VINCI CODE


--------------------------------
Process exited after 3.009 seconds with return value 0
Press any key to continue . . .
```