Algoritma Analizi Dersi 4.Ödev Raporu

Graf üzerinde BFS(Breadth First Search) Algoritması Uygulaması

Bu uygulamada verilen input dosyasındaki film ve aktör bilgisinden faydalanılarak graf oluşturulmuş, oluşturulan graf üzerinde BFS ve Backtracking yapılarak herbir aktörün Kevin Bacon sayısı ve path oluşturulmuştur.

M.Yasin SAĞLAM

15011804

ALGORİTMA ANALİZİ

GRUP 1

Algoritma Analizi Dersi 4.Ödev Raporu

Graf üzerinde BFS(Breadth First Search) Algoritması Uygulaması

Yöntem

Problemde verilen giriş dosyası için, her Kevin Bacon sayısında toplam kaç aktör olduğunun hesaplanması ve adı verilen bir aktörün Kevin Bacon sayısını bularak bağlantıyı gösteren bilginin istenilen formatta yazdırılması istenmektedir.
İşlem adımları olarak öncelikle filmler ve aktörler tekrarsız olacak şekilde dosyalarda indislenerek saklanmıştır. Bunun yanında input dosyası film1 id actor id[1] actor id[2] olacak şekilde indisli hale getirilerek oluşturulmuştur. Bu işlemler büyük dosya boyutlarının işlem yükünü azaltmak ve string kıyaslama operasyonlarının getireceği yükü azaltmak için yapılmıştır. Çözüme modüler olarak gidilmiştir. Dataset çözümleyen create_Actors(), create_files(),read_from_files() fonksiyonları dataseti indisli hale getirip ilgili dosyalara yazmaktadır. Bunun yanında queue ve graf implementasyonları da struct ve fonksiyonları ile birlikte yapılmıştır. Kritik işlem ise bfSearch adlı fonksiyonda yapılmaktadır. Bu fonksiyon grafın visited adlı dizisinde her bir düğümün Kevin Bacon a olan uzaklığını tutmaktadır. Eğer alt ağda aranan aktör bulunursa backtrack adlı diziden kb numarası aranan aktörden 1 az olan kişilerin komşulukları arasında aranarak parent düğümlere ve en sonda ise kevin bacon a oynanılan film bilgileri nodelarda kenar bilgisi olarak saklanmıştır. Böylece istenilen output film bilgileri ve aktörler olacak şekilde yazdırılmaktadır.

Uygulama

İstenilen Sonuçlar İçin Ekran Görüntüleri

1-)

Input1.txt

```
RESULTS
Sonsuz olanlarin sayisi 3
KB = 1 olanlarin sayisi 1494
KB = 2 olanlarin sayisi 0
KB = 3 olanlarin sayisi 0
KB = 4 olanlarin sayisi 0
KB = 5 olanlarin sayisi 0
KB = 6 olanlarin sayisi 0
KB = 7 olanlarin sayisi 0
KB = 8 olanlarin sayisi 0
KB = 9 olanlarin sayisi 0
KB = 10 olanlarin sayisi 0
KB = 11 olanlarin sayisi 0
KB = 12 olanlarin sayisi 0
```

Input2.txt

```
RESULTS
Sonsuz olanlarin sayisi 1176
KB = 1 olanlarin sayisi 61
KB = 2 olanlarin sayisi 25
KB = 3 olanlarin sayisi 36
KB = 4 olanlarin sayisi 120
KB = 5 olanlarin sayisi 182
KB = 6 olanlarin sayisi 248
KB = 7 olanlarin sayisi 76
KB = 8 olanlarin sayisi 302
KB = 9 olanlarin sayisi 237
KB = 10 olanlarin sayisi 43
KB = 11 olanlarin sayisi 6
KB = 12 olanlarin sayisi 0
Devam icin 1 cikis icin 0 giriniz
```

Input3.txt

```
RESULTS
Sonsuz olanlarin sayisi 717
KB = 1 olanlarin sayisi 1372
KB = 2 olanlarin sayisi 93798
KB = 3 olanlarin sayisi 72980
KB = 4 olanlarin sayisi 1636
KB = 5 olanlarin sayisi 14
KB = 6 olanlarin sayisi 0
KB = 7 olanlarin sayisi 0
KB = 8 olanlarin sayisi 0
KB = 9 olanlarin sayisi 0
KB = 10 olanlarin sayisi 0
KB = 11 olanlarin sayisi 0
KB = 12 olanlarin sayisi 0
```

```
Name is :Streep, MerylIndex of actor is 254
Streep, Meryl 's Kevin Bacon number is 1
Streep, Meryl-Bacon, Kevin : River Wild, The (1994)
Press any key to continue . . .
```

```
Name is :Cage, Nicolas
Index of actor is 2790
Cage, Nicolas 's Kevin Bacon number is 2
Cage, Nicolas-Dillon, Matt : Rumble Fish (1983)
Dillon, Matt-Bacon, Kevin : Wild Things (1998)
Press any key to continue . . .
```

```
Name is :Samaha, Elie
Index of actor is 1256
Samaha, Elie 's Kevin Bacon number is 3
Samaha, Elie-Carrere, Tia : 20 Dates (1998)
Carrere, Tia-Herman, Paul : Top of the World (1997)
Herman, Paul-Bacon, Kevin : Sleepers (1996)
Press any key to continue . . .
```

```
Name is :Fanning, Dakota
Index of actor is 92989
Fanning, Dakota 's Kevin Bacon number is 2
Fanning, Dakota-Wiest, Dianne : I Am Sam (2001)
Wiest, Dianne-Bacon, Kevin : Footloose (1984)
Press any key to continue . . .
```

```
Name is :Nasit, Adile
Index of actor is -1
Sonsuz
Press any key to continue . . .
```

Sonuç

Kullanıcıdan input okuma vs gibi işlemleri en büyük değeri etkilemeyeceği için karmaşıklık hesabına katmadan asıl işlemlerin yapıldığı fonksiyona bakacak olursak bu fonksiyon bir bfs olduğundanh karmaşıklık başlangıç düğümüne olan uzaklık **d** olarak alınırsa ve dallanma sayısı **d** olarak alındığında karmaşıklık üstel büyüme faktörüne bağlı olarak ***kabaca*** **O(b^(d+1)) olacaktır.**

Kod

```
/**
@file

Bu uygulamada verilen input dosyasindaki film ve aktor bilgisinden faydalanilarak graf olusturulmus,
olusturulan graf uzerinde BFS ve Backtracking yapilarak,
her bir aktorun Kevin Bacon sayisi hesaplanmis ve path olusturulmustur.

@author

Name              :      Muhammed Yasin SAGLAM
Student No        :      15011804
Date              :      03/12/2017
E-Mail            :      myasinsaglam1907@gmail.com
Compiler Used     :      GCC
IDE               :      DEV-C++(Version 5.11)
Operating System  :      Windows 10 educational edition
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define Act_SIZE
#define Film_Num 46//193//14129 //input dosyalarindaki film sayilari yani satir sayilari
#define LINE_BUF 3800
#define NAME_BUF 150
#define FNAME_BUF 500
#define file1 "input-1.txt" //ilgili input dosyasini tutan makro
#define F_FILMS "films1.txt" //output dosyasi filmleri ve id lerini tutar
#define F_ACTORS "actors1.txt" //output dosyasi aktorleri ve id lerini tutar
#define F_GRAPH "graph1.txt"  //input olarak verilen dosyanin id lerle olusturulmus hali string operasyonlarindan kurtulmak icin
#define delim "/"

char films[Film_Num][FNAME_BUF]; //film isimlerini tutan dizi
char **actors; //aktorleri tutan pointer
int asize=0; //toplam tekrarsiz aktor sayisi

typedef struct{
    int f_id;
    int *actors;
    int actor_num;
}PLAY;
//verilen filmde hangi aktorlerin oynadigini tutan yapi
PLAY playing[Film_Num];

//aktorlerin oynadiklari filmleri tutan yapi komsuluk olusturmada kullanilacaktir
typedef struct{
    int a_id;
    int *films;
    int film_num;
}PLAYA;
PLAYA *actor_film;

//grafin 1 node u dugum olarak aktor id si kenar olarak ise birlikte oynadiklari film id si tutulmaktadir
struct node{
```

```
56.     int a_id; //aktor id si
57.     int f_id; //film id si
58.     int kbnum; //kbnumarasi
59.     struct node* next; //graf icin sonraki dugumun adresi
60. };
61. 
62. //Tanimlanan Graph yapisinda komsuluk listeleri , dugum sayisi ve ziyaret edilen dugumler tutu
    lmaktadir.
63. struct Graph{
64.     int numVertices; //dugum sayisi
65.     struct node** adjLists; //komsuluk listesi
66.     int *visited; //ziyaret edilen dugumler
67. };
68. 
69. //node initialize eden, olusturan fonksiyon grafa ekleme yaparken kullanilmistir.Parametre ola
    rak aktor idsi ve film id si alir
70. struct node* createNode(int actor,int film){
71.     struct node* New = malloc(sizeof(struct node));
72.     New->a_id=actor;
73.     New->f_id=film;
74.     New->next=NULL;
75.     New->kbnum=0;
76.     return New;
77. }
78. 
79. //graph olusturan fonksiyon
80. struct Graph* createGraph(int vertices){
81.     int i;
82.     struct Graph* graph= malloc(sizeof(struct Graph));
83.     graph->numVertices=vertices;
84.     graph->adjLists=malloc(vertices * sizeof(struct node*));
85.     graph->visited=malloc(sizeof(int)*vertices);
86.     for(i=0;i<vertices;i++){
87.         graph->adjLists[i]=NULL;
88.         graph->visited[i]=0;
89.     }
90.     return graph;
91. }
92. 
93. //grafa dugum ekleyen fonksiyon
94. void addEdge(struct Graph* graph, int v1,int e1,int v2, int e2){
95.     struct node* New = createNode(v2,e2);
96.     New->next=graph->adjLists[v1];
97.     graph->adjLists[v1]=New;
98. 
99.     New = createNode(v1,e1);
100.            New->next=graph->adjLists[v2];
101.            graph->adjLists[v2]=New;
102.        }
103. 
104. 
105. 
106.        //kuyruk elemanlarinin yapisi olusturuluyor
107.        typedef struct queue_element{
108.            struct node value;
109.            struct queue_element *next;
110.        }queue_element;
111. 
112.        //kuyruk yapisi
113.        struct Queue {
114.            struct queue_element *front, *rear;
```

```
115.        };
116.         //kuyruk icin eleman olusturup donduren fonksiyon
117.        struct queue_element* newElement(struct node * node){
118.            struct queue_element *temp = (struct queue_element*)malloc(sizeof(struct queue_elem
    ent));
119.            temp->value=*node;
120.            temp->next=NULL;
121.            return temp;
122.        }
123.        //kuyruk olusturan fonksiyon
124.        struct Queue* createQueue(){
125.            struct Queue *q = (struct Queue*)malloc(sizeof(struct Queue));
126.            q->front=q->rear=NULL;
127.            return q;
128.        }
129.        //kuyruk bos mu kontrolu yapan fonksiyon
130.        int isEmpty(struct Queue *q){
131.            if(q->front==NULL){
132.                //printf("\nQueue is empty");
133.                return 1;
134.            }
135.            else{
136.                return 0;
137.            }
138.        }
139.        //kuyruga eleman ekleyen fonksiyon
140.        void enqueue(struct Queue *q, struct node *value){
141.            struct queue_element *element= newElement(value);
142.            element->value=*value;
143.            element->next=NULL;
144.            if (q->rear==NULL){//kuyruk bossa
145.                //ilk eleman demektir
146.                q->front=q->rear=element;
147.            }
148.            else{ //bos degilse sona eklenecektir
149.                q->rear->next=element;
150.                q->rear=element;
151.            }
152.        }
153.        //kuyruktan eleman ceken fonksiyon
154.        struct queue_element* dequeue(struct Queue *q){
155.            if(q->front==NULL){//kuyruk bossa null dondur
156.                return NULL;
157.            }
158.            struct queue_element *temp = q->front; //eski ilk elemanin adresini sakla
159.            q->front=q->front->next; //elemani kuyruktan cikar
160.            if(q->front== NULL) //eger kuyruk bosaldiysa rear i da null yap
161.                q->rear=NULL;
162.            return temp;
163.        }
164.
165.        //BFS ve aranan eleman bulunursa backtrack ederek path i yazdiran fonksiyon
166.        void bfSearch(struct Graph *graph, int startActorIndex, int searchIndex){
167.            int i,j;
168.            int count =0;
169.            int not_found=1;
170.            int *backtrack=malloc(sizeof(int));
171.            struct Queue *queue =createQueue(); //kuyruk olusturuluyor
172.            graph->visited[startActorIndex]=-1; //kevin bacon icin visited dizisinde -1 yazar
173.            //printf("\nFirst actor id > %d Name > %s",startActorIndex,actors[startActorIndex])
    ;
```

```
174.            struct node first;
175.            first.a_id=startActorIndex;
176.            first.kbnum=0;
177.            first.next=NULL;
178.            enqueue(queue,&first);
179.            while(!isEmpty(queue)){
180.                //kuyrugu yazdir burada biyerde
181.                struct queue_element *temp = dequeue(queue); //kuyruktan elemani cek
182.                backtrack=realloc(backtrack, sizeof(int)*(count+1));
183.                backtrack[count++]=temp->value.a_id;
184.
185.                //printf("\nVisited actor id > %d Name > %s ",temp->value.a_id,actors[temp-
    >value.a_id]);
186.                struct node *iter = graph->adjLists[temp->value.a_id]; //ilk komsusunu al
187.                while(iter){
188.                    if(graph->visited[iter-
    >a_id]==0){ //kuyruktan cekilenin komsusu ziyaret edilmemisse
189.                        iter->kbnum = temp->value.kbnum+1;
190.                        graph->visited[iter->a_id]=iter-
    >kbnum; //visited icinde kb numberlar saklanir 0 ise ziyaret edilmemis demektir
191.                        enqueue(queue,iter); //ekle queue ya
192.                    }
193.                    //backtracking
194.                    if(searchIndex==iter->a_id && graph->visited[iter->a_id]==iter->kbnum){
195.                        //printf("\nFound...\n\nactor id > %d actor name > %s Film is > %s Kb n
    umber is %d\n\n",iter->a_id,actors[iter->a_id],films[iter->f_id],iter->kbnum);
196.                        not_found=0;
197.                        printf("\n%s 's Kevin Bacon number is %d\n",actors[iter->a_id],iter-
    >kbnum);
198.                        int a1_id = iter->a_id; //alt agda bulunan aktor id si
199.                        int a1_kb=iter->kbnum; //alt agda bulunan aktor kb no su
200.                        int a2_id;
201.                        int a2_kb;
202.                        struct node *temp;
203.                        for(i=0;i<count;i++){
204.                            int found=0;
205.                            int tmp_id; //komsu aktor id si
206.                            a2_id=backtrack[count-i-1]; //queueden cekilen id ler
207.                            a2_kb=graph->visited[a2_id];
208.                            struct node *temp = graph-
    >adjLists[a2_id]; //komsu aktorun  komsuluklari temp node un da tutulur
209.                            if(a2_kb ==(a1_kb-1) || a2_kb==-
    1){ //eger komsu aktorun kb nosu kucukse bulunan aktorun kb nosundan 1 kucukse ve
210.                                while(temp && !found){
211.                                    tmp_id = temp->a_id; //id lere bak
212.                                    if(tmp_id == a1_id){
213.                                        printf("%s-
    %s : %s \n",actors[a1_id],actors[a2_id],films[temp->f_id]);
214.                                        found=1;
215.                                        a1_id=a2_id;
216.                                        a1_kb=a2_kb;
217.                                    }
218.                                    temp=temp->next;
219.                                }
220.                            }
221.                        }
222.                        system("PAUSE");
223.                    }
224.                    //end of backtracking
225.                    iter=iter->next;
226.                }
```

```
227.            }
228.            if(not_found){
229.                printf("\nSonsuz\n");
230.                system("PAUSE");
231.            }
232.            system("CLS");
233.            printf("RESULTS");
234.            int frequency[13];
235.        //    if(graph->visited[searchIndex]==0){
236.        //        printf("\nsonsuz");
237.        //        system("PAUSE");
238.        //    }
239.        //    for(i=0;i<asize;i++){
240.        //        printf("\n%-5d.%-30s KB:%d",i,actors[i],graph->visited[i]);
241.        //    }
242.            for(i=0;i<13;i++)
243.                frequency[i]=0;
244.            for(i=0;i<13;i++){
245.                for(j=0;j<asize;j++){
246.                    if(graph->visited[j]==i)
247.                        frequency[i]++;
248.                }
249.            }
250.            printf("\nSonsuz olanlarin sayisi %d ",frequency[0]);
251.            for(i=1;i<13;i++){
252.                printf("\nKB = %d olanlarin sayisi %d",i,frequency[i]);
253.            }
254.        }
255.
256.
257.
258.        //aktor ve filmleri dosyaya yazan ve okuyan fonksiyonlar
259.        int create_Actors(){
260.            FILE *fp,*fpgr;
261.            char *line=calloc(LINE_BUF,sizeof(char));
262.            char *tok;
263.            int fsize=0;
264.            int i,k=0;
265.            int lc; //line counter cevrim degiskeni
266.            actors=(char**)malloc(sizeof(char*));
267.            actors[0]=(char*)malloc(sizeof(char)*NAME_BUF);
268.            fpgr=fopen(F_GRAPH,"w");
269.            if(!fpgr){
270.                printf("\nFile not opened. Quitting...");
271.                exit(0);
272.            }
273.            fp=fopen(file1,"rb");
274.            if(!fp){
275.                printf("\nFile not opened. Quitting...");
276.                exit(0);
277.            }
278.        //    while(fgets(line, sizeof(char)*LINE_BUF,fp)!=NULL){
279.        //        printf("%s",line);
280.        //    }
281.            for(lc=0;lc<Film_Num;lc++){ //dosyanin sonuna gelene kadar
282.                fgets(line, sizeof(char)*LINE_BUF,fp); //satir oku
283.                line[strlen(line)-1]='\0';
284.                //printf("%s\n",line);
285.                //system("PAUSE");
286.                tok=strtok(line,delim); //ilk elemani (filmadi) / a gore parcala
287.                //printf("\nFilm adi : %s",tok);
```

```
288.                fprintf(fpgr,"%d",k++);
289.                strcpy(films[fsize++],tok); //ilk elemani filmler dizisine yerlestir.
290.                tok = strtok(NULL ,delim);
291.                while (tok != NULL) { //oyuncu adlari icin satir sonuna kadar / a gore parcalam
    aya devam et
292.                    // printf("\nOyuncu adi : %s",tok);
293.                    i=0;
294.                    while(i<asize && strncmp(actors[i],tok,NAME_BUF)!=0) //actor onceden eklenm
    is mi
295.                        i++;
296.                    if(i==asize){ //eklenmemisse
297.                        asize++; //actor dizisinin boyutunu artir
298.                        actors=realloc(actors,asize*sizeof(char*)); //bellekte realloc ile yer
    ayir
299.                        actors[asize-
    1]=(char*)malloc(sizeof(char)*NAME_BUF); //isim icinde yer ac
300.                        strcpy(actors[asize-1],tok); //son goze actoru ekle
301.                        // printf("\nActors %d : %s",asize,actors[asize-1]);
302.                        fprintf(fpgr," %d",i);
303.
304.                    }
305.                    else{
306.                        fprintf(fpgr," %d",i);
307.                    }
308.                    tok = strtok(NULL ,delim); //oyuncu adini tok a al
309.                }
310.                fputc('\n',fpgr);
311.            }
312.            //printf("\nLine num %d",lc);
313.            fclose(fp);
314.            fclose(fpgr);
315.        }
316.
317.        int create_files(){
318.            int i,j;
319.            FILE *fp1,*fp2;
320.            fp1=fopen(F_FILMS,"w");
321.            if(!fp1){
322.                printf("File not opened to write!!! Quitting...");
323.                exit(0);
324.            }
325.            for(i=0;i<Film_Num;i++) {
326.                //printf("\nFilm %d. : %s\n",i+1, films[i]);
327.                fprintf(fp1,"%d-%s\n",i,films[i]);
328.            }
329.            fclose(fp1);
330.            fp2=fopen(F_ACTORS,"w");
331.            if(!fp2){
332.                printf("File not opened to write!!! Quitting...");
333.                exit(0);
334.            }
335.            fprintf(fp2,"%d\n",asize);
336.            for(j=0;j<asize;j++){
337.                //printf("\t%d. %s\n",j+1,actors[j]);
338.                fprintf(fp2,"%d-%s\n",j,actors[j]);
339.            }
340.            fclose(fp2);
341.            return 0;
342.        }
343.
344.        //dosyadan okuma yapar aktor ve filmleri kendi olusturdugum dosyadan
```

```
345.        int read_from_files(){
346.            FILE *fp1,*fp2,*fp3;
347.            char *line=calloc(LINE_BUF,sizeof(char));
348.            char *tok;
349.            int i=0,j=0;
350.            int k;
351.            //aktorler okunuyor
352.            fp2=fopen(F_ACTORS,"r");
353.            if(!fp2){
354.                printf("File error!!! Quitting");
355.                exit(0);
356.            }
357.            fscanf(fp2,"%d",&asize);
358.            fgets(line,sizeof(char)*LINE_BUF,fp2);
359.            actors=(char**)(char**)malloc(sizeof(char*)*asize);
360.            for(i=0;i<asize;i++){
361.                //fgets BURADAN DEVAM ET DOSYAYA YAZIM DEGISECEK YADA TUMUYLE KAYIT ETME
362.                fgets(line,sizeof(char)*LINE_BUF,fp2);
363.                line[strlen(line)-1]='\0';
364.                tok=strtok(line,"-");
365.                tok=strtok(NULL,"-");
366.                while(tok!=NULL){
367.                    actors[i]=(char*)malloc(sizeof(char)*NAME_BUF);
368.                    strcpy(actors[i],tok);
369.                    //printf("\n %d. actor >  %s ",i,actors[i]);
370.                    tok=strtok(NULL,"-");
371.                }
372.            }
373.            fclose(fp2);
374.            //filmler okunuyor
375.            fp3=fopen(F_FILMS,"r");
376.            if(!fp3){
377.                printf("File error!!! Quitting");
378.                exit(0);
379.            }
380.            for(i=0;i<Film_Num;i++){
381.                fgets(line,sizeof(char)*LINE_BUF,fp3);
382.                line[strlen(line)-1]='\0';
383.                tok=strtok(line,"-");
384.                tok=strtok(NULL,"-");
385.                while(tok!=NULL){
386.                    strcpy(films[i],tok);
387.                    //printf("\n %d. film >  %s ",i,films[i]);
388.                    tok=strtok(NULL,"-");
389.                }
390.            }
391.            fclose(fp3);
392.            //grafin inputtaki hali okunuyor
393.            fp1=fopen(F_GRAPH,"r");
394.            if(!fp1){
395.                printf("File error!!! Quitting");
396.                exit(0);
397.            }
398.            for(i=0;i<Film_Num;i++){
399.                fgets(line,sizeof(char)*LINE_BUF,fp1);
400.                line[strlen(line)-1]='\0';
401.                tok=strtok(line," ");
402.                playing[i].f_id=atoi(tok);
403.                //printf("\nfilm %d",playing[i].f_id);
404.                //printf("\nFilm id : %d Film Name : %s",playing[i].f_id,films[playing[i].f_id]
    );
```

```
405.                tok=strtok(NULL," ");
406.                playing[i].actors=(int*)calloc(1,sizeof(int));
407.                j=0;
408.                while(tok!=NULL){
409.                    playing[i].actors[j]=atoi(tok);
410.                    //printf("\n\tActor id > %d Actor Name > %s",playing[i].actors[j],actors[pl
    aying[i].actors[j]]);
411.                    j++;
412.                    playing[i].actors=(int*)realloc(playing[i].actors,sizeof(int)*(j+1));
413.                    tok=strtok(NULL," ");
414.                    //printf(" %d ",playing[i].actors[j-1]);
415.                }
416.
417.                playing[i].actors=realloc(playing[i].actors,sizeof(int)*j);
418.                playing[i].actor_num=j; //filmde oynayan aktor sayisi saklaniyor
419.                //printf("\nTotal actor %d ",playing[i].actor_num);
420.                //system("PAUSE");
421.            }
422.            fclose(fp1);
423.
424.            //her bir oyuncunun oynadigi filmler actor film yapisi icerisinde olusturuluyor
425.            actor_film=(PLAYA*)malloc(sizeof(PLAYA)*asize);
426.            for(i=0;i<asize;i++){
427.                actor_film[i].film_num=0;
428.                actor_film[i].a_id=i;
429.                actor_film[i].films=(int*)malloc(sizeof(int));
430.            }
431.            for(i=0;i<asize;i++){ //her bir aktor icin
432.                for(j=0;j<Film_Num;j++){//her bir filme bak
433.                    for(k=0;k<playing[j].actor_num;k++){ //filmde oynayan aktorler arasinda
434.                        if(playing[j].actors[k]==i){//bakilan aktor var mi varsa
435.                            actor_film[i].films=realloc(actor_film[i].films,sizeof(int)*(actor_
    film[i].film_num+1));//actorun oynadigi filmleri realloc yap
436.                            actor_film[i].films[actor_film[i].film_num]=j; //ilgili aktore ilgi
    li filmin indisini ekle
437.                            actor_film[i].film_num++; //aktorun oynadigi film sayisini artir
438.                            //printf("\n\t%d. actor %d. film--
    res %d Total Films are %d k is %d in %d",i,j,playing[j].actors[k],actor_film[i].film_num,k,pla
    ying[j].actor_num);
439.                            //system("PAUSE");
440.                        }
441.                    }
442.                }
443.                //kontrol amacli yazdirmalar
444.                //printf("\nActor id %d Name > %s ",i,actors[i]);
445.                //printf("\n\tTotal Films are %d ",actor_film[i].film_num);
446.                //system("PAUSE");
447.            }
448.            //kontrol amacli yazdirmalar
449.        //    for(i=0;i<asize;i++){
450.        //        printf("\nActor id %d Name > %s ",i,actors[i]);
451.        //        for(j=0;j<actor_film[i].film_num;j++){
452.        //            printf("\n\tFilm id %d Film name > %s ",actor_film[i].films[j],films[acto
    r_film[i].films[j]]);
453.        //        }
454.        //    }
455.
456.        }
457.
458.        //kevin bacon un indexini bulan fonksiyon
459.        int find_kbacon(){
```

```
460.            int i=0;
461.            while(i<asize && strcmp("Bacon, Kevin",actors[i])!=0)
462.                i++;
463.            if(i==asize)
464.                return -1;
465.            else
466.                return i;
467.        }
468.
469.        //adi verilen aktorun indisini bulan ve donduren fonksiyon
470.        int find_actor(char *name){
471.            int i=0;
472.            printf("\nName is :%s",name);
473.            while(i<asize && strcmp(name,actors[i])!=0)
474.                i++;
475.            if(i==asize){
476.                printf("\nAktor bulunamadi\n");
477.                return -1;
478.            }
479.            else{
480.                return i;
481.            }
482.        }
483.
484.        //dosyadan okuma fonksiyonlarini cagirir okuma bittikten sonra graf olusturulur, sonras
    inda ise input alarak bfs yapip ekrana istenilen sonuclari dondurur.
485.        int main(){
486.            int i,j,k,index,act_index;
487.            struct node *iter;
488.            int control;
489.            int choice=1;
490.            char *name=malloc(sizeof(char)*NAME_BUF);
491.            int name_index;
492.            struct Graph *graph;
493.            FILE *fp1,*fp2;
494.            //dataset ayristirip dosyaya yazan fonksiyonlar
495.            create_Actors();
496.            create_files();
497.            //dosyadan okuma yaparak ilgili veri yapilarina degerleri aktaran fonksiyon
498.            read_from_files();
499.            printf("\nBaslangic icin 1 cikis icin 0 giriniz \n");
500.            scanf("%d",&choice);
501.            while(choice){
502.                graph=createGraph(asize);
503.                //komsuluklar olusturuluyor yani graf olusturuluyor
504.                for(i=0;i<asize;i++){ //ilk aktorun id si
505.                    for(j=0;j<actor_film[i].film_num;j++){ //oynadigi filmlerdeki
506.                        index=actor_film[i].films[j]; //film id si
507.                        for(k=0;k<playing[index].actor_num;k++){ //bu filmde oynayan diger oyun
    cu sayisi kadar don
508.                            act_index=playing[index].actors[k]; //bu filmde oynayan diger oyunc
    u indexi
509.                            if(act_index!=i) {//ilk oyuncunun indexine esit degilse
510.                                iter = graph->adjLists[i]; //komsuluk adresini iter e at
511.                                control=0;
512.                                while (iter != NULL && !control){ //eger iter null degilse ve o
    nceden eklenmemisse
513.                                    if(iter-
    >a_id==act_index){ //aktor indisi komsuluklarda yoksa
514.                                        control=1; //donguden cikis yap
515.                                    }
```

```
516.                                    iter=iter->next;
517.                                }
518.                                if(!control){//eger controllu bir cikis yapilmissa yani komuslu
    klarin sonuna gelinmemisse
519.                                    addEdge(graph, i, index, act_index, index); //grafa ekleme
    yap
520.                                    //printf("Actor index > %d Actor > %s\nActor index >%d Acto
    r > %s\nFilm index > %d Film > %s\n",i,actors[i],act_index,actors[act_index],index,films[index
    ]);
521.                                    // system("PAUSE");
522.                                }
523.                            }
524.                        }
525.                    }
526.                }
527.                system("CLS");
528.                fgets(name,NAME_BUF,stdin);
529.                printf("Oyuncu adini giriniz : ");
530.                scanf ("%[^\n]%*c", name);
531.                name_index=find_actor(name);
532.                printf("\nIndex of actor is %d",name_index);
533.                bfSearch(graph,find_kbacon(),name_index);
534.                printf("\nDevam icin 1 cikis icin 0 giriniz \n");
535.                scanf("%d",&choice);
536.            }
537.
538.
539.            printf("\nKBacon's index is %d\n",find_kbacon());
540.            for (i = 0; i < asize; i++) {
541.                free(actors[i]);
542.            }
543.            free(actors);
544.            system("PAUSE");
545.            return 0;
546.        }
```