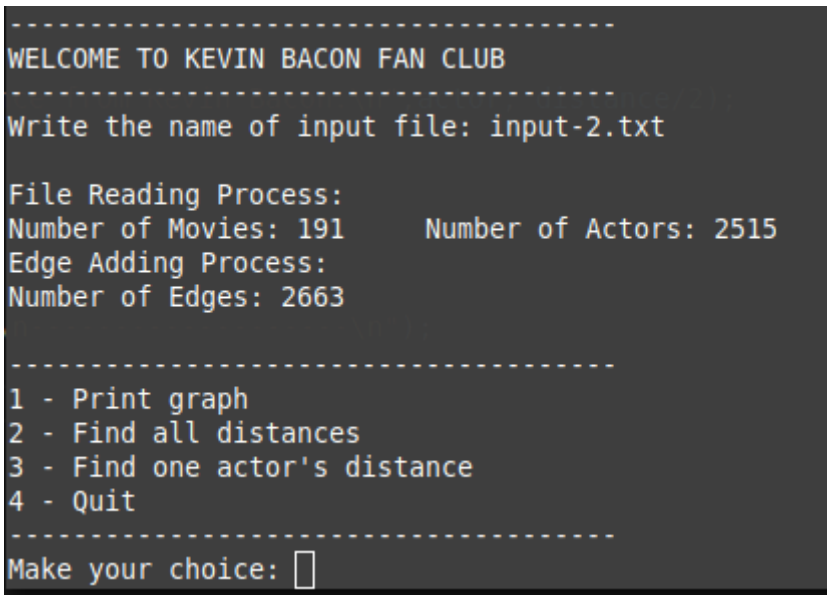
**Enes Sadi Uysal 17011041**

# 2020 Bahar Dönemi Veri Yapıları ve Algoritmalar Dersi Proje Raporu

## Algoritmanın Açıklanması

Kullanıcıdan alınan girdi dosyasının ismi ile birlikte dosya bir kere baştan sona okunur. Bu okuma sırasında '/' karakterine ve satır başlarına dikkat edilerek movies ve actors isminde iki tane dizi oluşturulur. Bu diziler oluşturulurken film olanların film dizisine, aktör olanların ise aktör dizisine koyulmasına ve ayrıca bu dizilerin küme mantığıyla doldurulmasına dikkat edilir (eleman tekrarı olmayacak).

Dosya okunduktan sonra toplam node sayısı bulunur ve graf oluşturulur. Ardından graf üzerinde node'lar arasındaki komşuluk bağlantıları dosya bir kere daha okunarak ilgili filme ilgili oyuncuyu bağlama ve ilgili oyuncuya da illgili filmi bağlama şeklinde kenar oluşturma fonksiyonu ile sağlanır. (Bu fonksiyonun ayrıntılı açıklaması kaynak kod üzerinde yorum satırlarıyla yapılmıştır.)




Kenar bağlantıları ile birlikte elimizde artık BFS algoritmasını kullanabileceğimiz bir grafımız oluşmuştur. Ardından bu grafı kullanarak kullanıcının istekleri ( Grafı yazdırma, girilen oyuncunun Kevin Bacon sayısını bulma ve gidilen yolu yazdırma, bütün oyuncular için frekans dizisi oluşturma) yerine getirilir. Bu işlemlerin de nasıl yapıldıkları kaynak kod üzerinde yorum satırları ile ayrıntılı açıklanmıştır.

## Frekans Dizilerinin Ekran Çıktıları

### input-1.txt

```
1       of actors have  0       distance from Kevin Bacon.
1494    of actors have  1       distance from Kevin Bacon.
3       of actors have infinite distance from Kevin Bacon.
```

### input-2.txt

```
1       of actors have  0       distance from Kevin Bacon.
61      of actors have  1       distance from Kevin Bacon.
25      of actors have  2       distance from Kevin Bacon.
36      of actors have  3       distance from Kevin Bacon.
120     of actors have  4       distance from Kevin Bacon.
182     of actors have  5       distance from Kevin Bacon.
248     of actors have  6       distance from Kevin Bacon.
76      of actors have  7       distance from Kevin Bacon.
302     of actors have  8       distance from Kevin Bacon.
237     of actors have  9       distance from Kevin Bacon.
43      of actors have  10      distance from Kevin Bacon.
6       of actors have  11      distance from Kevin Bacon.
1178    of actors have infinite distance from Kevin Bacon.
```

### input-3.txt

```
1       of actors have  0       distance from Kevin Bacon.
1372    of actors have  1       distance from Kevin Bacon.
93798   of actors have  2       distance from Kevin Bacon.
72979   of actors have  3       distance from Kevin Bacon.
1636    of actors have  4       distance from Kevin Bacon.
14      of actors have  5       distance from Kevin Bacon.
717     of actors have infinite distance from Kevin Bacon.
```

**Raporda İstenen Oyuncular İçin**
**input-3.txt Dosyasındaki Sonuçlar**

```
Name of actor: Streep, Meryl
[ Streep, Meryl ] played in
[ River Wild, The (1994) ] is the movie, that includes the actor named
[ Bacon, Kevin ].

So [ Streep, Meryl ] has [ 1 ] distance from Kevin Bacon.
```

```
Name of actor: Cage, Nicolas
[ Cage, Nicolas ] played in
[ Rumble Fish (1983) ] is the movie, that includes the actor named
[ Dillon, Matt ] that played in
[ Wild Things (1998) ] is the movie, that includes the actor named
[ Bacon, Kevin ].

So [ Cage, Nicolas ] has [ 2 ] distance from Kevin Bacon.
```

```
Name of actor: Samaha, Elie
[ Samaha, Elie ] played in
[ 20 Dates (1998) ] is the movie, that includes the actor named
[ Carrere, Tia ] that played in
[ Hostile Intentions (1994) ] is the movie, that includes the actor named
[ Pickens Jr., James ] that played in
[ Sleepers (1996) ] is the movie, that includes the actor named
[ Bacon, Kevin ].

So [ Samaha, Elie ] has [ 3 ] distance from Kevin Bacon.
```

```
Name of actor: Fanning, Dakota
[ Fanning, Dakota ] played in
[ I Am Sam (2001) ] is the movie, that includes the actor named
[ Dern, Laura ] that played in
[ Novocaine (2001) ] is the movie, that includes the actor named
[ Bacon, Kevin ].

So [ Fanning, Dakota ] has [ 2 ] distance from Kevin Bacon.
```

```
Name of actor: Naşit, Adile
[ Naşit, Adile ] played in
nothing or even he/she played some movies he/she has no connection with Kevin Bacon.
```

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

#define MAX1 200000         //Dosyayı okuyup aktörleri ve filmleri içine koyacağımız dizilerin maksimum boyutu
#define MAX2 200            //Film veya aktörün isminin maksimum uzunluğu

//Buradaki dizileri global tanımlama sebebim dizi boyutlarının çok büyük olması ve bu boyutun fonksiyon içinde tanımlama için fazla olmasıdır.
char movies[MAX1][MAX2];    //Filmleri tutacak olan dizimiz
char actors[MAX1][MAX2];    //Aktörleri tutacak olan dizimiz

//Derste yazdığımız Graf Fonksiyonları başlangıcı
struct node{
    int vertex;
    struct node *next;
};

struct Graph{
    int numVertices;
    struct node **adjLists;
};

struct node* createNode(int vertex){

    struct node *newNode = malloc(sizeof(struct node));
    newNode->vertex = vertex;
    newNode->next = NULL;
    return newNode;
}

struct Graph* createGraph(int vertices){

    int i;
    struct Graph *graph = malloc(sizeof(struct Graph));
    graph->numVertices = vertices;
    graph->adjLists = malloc(vertices * sizeof(struct node*));
    for(i = 0 ; i < vertices ; i++){
        graph->adjLists[i] = NULL;
    }
    return graph;
}

void addEdge(struct Graph* graph, int src, int dest){

    struct node *newNode = createNode(dest);
    newNode->next = graph->adjLists[src];
    graph->adjLists[src] = newNode;

    newNode = createNode(src);
    newNode->next = graph->adjLists[dest];
    graph->adjLists[dest] = newNode;
}
//Derste yazdıklarımızın sonu

//Graf yapımı yazdırdığım fonksiyon
void printGraph(struct Graph *myGraph, int movieCounter){

    struct Graph *g = myGraph;
```

```
59      //Yazdırırken indekse göre yapılan movie & actor ayrımı raporda ayrıntılı
    biçimde açıklanmıştır.
60      int i;
61      for(i = 0 ; i < g->numVertices ; i++){
62          if(i >= movieCounter){
63              printf("%s -->  ", actors[i-movieCounter]);
64          }
65          else{
66              printf("%s -->  ", movies[i]);
67          }
68
69          struct node *tmp = g->adjLists[i];
70          while(tmp){
71              if(tmp->vertex >= movieCounter){
72                  printf("%s ||", actors[(tmp->vertex)-movieCounter]);
73              }
74              else{
75                  printf("%s ||", movies[tmp->vertex]);
76              }
77              tmp = tmp->next;
78          }
79          printf("\n\n");
80      }
81  }
82
83  //Okunan kelimenin film olup olmadığını döndüren fonksiyon
84  int isMovie(char tmp[MAX2]){
85
86      int i = 0;
87      while(tmp[i] != '\0' && tmp[i] != '('){
88          i++;
89      }
90      if(tmp[i] == '('){  //Eğer '(' içeriyorsa filmdir.
91          return 1;
92      }
93      else{
94          return 0;
95      }
96  }
97
98  //Dosyadan okunan film veya aktörün ilgili dizide zaten olup olmadığı kontrol
    edilmektedir.
99  //Zaten var ise diziye koyulmaz çünkü graf yapısında aynı isimde iki node
    olmamalı.
100 int isThere(int movieOrActor, int i, char vertex[MAX2]){
101
102     if(movieOrActor == 1){
103         while(strcmp(movies[i], vertex) && i >= 0){
104             i--;
105         }
106         if(strcmp(movies[i], vertex) == 0){
107             return 1;
108         }
109         else{
110             return 0;
111         }
112     }
113     else{
114         while(strcmp(actors[i], vertex) && i >= 0){
115             i--;
116         }
```

```
        if(strcmp(actors[i], vertex) == 0){
            return 1;
        }
        else{
            return 0;
        }
    }
}

//Dosyanın okunup movies ve actors dizilerinin doldurulması işleminin yapıldığı fonksiyon
void readFile(int *movieCounter, int *actorCounter, char nameOfFile[50]){

    FILE *fp = fopen(nameOfFile, "r");

    if(fp == NULL){
        printf("File does not exist!");
        exit(1);
    }

    char tmp = getc(fp);
    int i;
    char tmpWord[MAX2];
    i = 0;
    //İlk okunan değerin film olduğu bilindiğinden film dizisine atılır.
    while(tmp != '/'){
        tmpWord[i] = tmp;
        i++;
        tmp = getc(fp);
    }
    tmpWord[i] = 0;
    strcpy(movies[0], tmpWord);
    (*movieCounter)++;

    tmp = getc(fp);
    //Okunan her kelimenin film veya aktör olduğuna karar verilir ve
    //İlgili dizide eğer yoksa o diziye atanır.
    while(!feof(fp)){
        i = 0;
        while(!feof(fp) && tmp != '/' && tmp != '\n'){
            tmpWord[i] = tmp;
            i++;
            tmp = getc(fp);
        }
        tmpWord[i] = 0;

        if(isMovie(tmpWord)){
            strcpy(movies[(*movieCounter)++], tmpWord);
        }
        else{
            if(!isThere(0, *actorCounter, tmpWord)){
                strcpy(actors[*actorCounter], tmpWord);
                (*actorCounter)++;
            }
        }
        printf("\rNumber of Movies: %d     Number of Actors: %d", *movieCounter, *actorCounter);
        tmp = getc(fp);
    }
    fclose(fp);
}
```

```
176
177 //Parametre olarak word'un gerekli dizideki indisini bulup döndürür.
178 int findIndex(int movieOrActor, char word[MAX2], int n){
179     int i = 0;
180     if(movieOrActor == 1){
181         while(i < n && strcmp(movies[i], word) != 0){
182             i++;
183         }
184         if(strcmp(movies[i], word) == 0){
185             return i;
186         }
187         else{
188             return -1;
189         }
190     }
191     else{
192         while(i < n && strcmp(actors[i], word) != 0){
193             i++;
194         }
195         if(strcmp(actors[i], word) == 0){
196             return i;
197         }
198         else{
199             return -1;
200         }
201     }
202 }
203
204 //Dosyayı tekrar okuyarak, oluşturulan grafa kenar bağlantılarının eklendiği
    fonksiyon.
205 struct Graph* createEdges(struct Graph *myGraph, int movieCounter, int
    actorCounter, char nameOfFile[50]){
206
207     //Öncelikle dosyadan okunan filmin movies dizisindeki indisi tutulur ve
    bir sonraki film okunana kadar
208     //okunan bütün aktörlerin actors dizisindeki indisleri ile film sayısı
    toplanarak oluşturulan sayı arasında
209     //bir kenar bağlantısı oluşturulur.
210
211     int movieIndex, actorIndex;
212     static int edgeCounter = 0;
213
214     FILE *fp = fopen(nameOfFile, "r");
215
216     if(fp == NULL){
217         printf("File does not exist!");
218         exit(1);
219     }
220
221     char tmp = getc(fp);
222     int i;
223     char tmpWord[MAX2];
224     i = 0;
225     while(tmp != '/'){
226         tmpWord[i] = tmp;
227         i++;
228         tmp = getc(fp);
229     }
230     tmpWord[i] = 0;
231     //İlk okunan kelimenin film olduğu bilindiğinden bu indis tutulur.
232     movieIndex = findIndex(1, tmpWord, movieCounter);
```

```
233
234     tmp = getc(fp);
235
236     while(!feof(fp)){
237         i = 0;
238         while(!feof(fp) && tmp != '/' && tmp != '\n'){
239             tmpWord[i] = tmp;
240             i++;
241             tmp = getc(fp);
242         }
243         tmpWord[i] = 0;
244         //Ardından okunan bütün kelimeler için film olup olmadığı kontrolü
    yapıldıktan sonra
245         if(isMovie(tmpWord)){
246             movieIndex = findIndex(1, tmpWord, movieCounter);
247         }
248         //aktör ise daha önce tutulan film indisi ile arasında kenar
    bağlantısı kurulur.
249         else{
250             actorIndex = findIndex(0, tmpWord, actorCounter);
251             edgeCounter++;
252             //Kenar eklenirken bulunan aktörün indisine film sayısının
    eklenme sebebi
253             //Graf yapısında ilk node'ların filmlerden, sonraki nodeların ise
    oyunculardan oluşmasıdır.
254             //Kullanıcı bu yapıyı grafı yazdır seçeneğini kullanarak daha net
    görebilir.
255             addEdge(myGraph, movieIndex, actorIndex + movieCounter);
256             printf("\rNumber of Edges: %d", edgeCounter);
257         }
258         tmp = getc(fp);
259     }
260     fclose(fp);
261
262     return myGraph;
263 }
264
265 //Derste yazdığımız Kuyruk Yapısı Fonksiyonları başlangıcı
266 struct QNode{
267     int value;
268     struct QNode *next;
269 };
270
271 struct Queue{
272     struct QNode *front, *rare;
273 };
274
275 struct Queue *createQueue(){
276
277     struct Queue *q;
278     q = (struct Queue*)malloc(sizeof(struct Queue));
279     if(q == NULL){
280         exit(0);
281     }
282     else{
283         q->front = q->rare = NULL;
284         return q;
285     }
286 }
287
288 void enqueue(struct Queue *q, int value){
```

```
289
290     struct QNode *tmp;
291     tmp = (struct QNode*)malloc(sizeof(struct QNode));
292     if(tmp == NULL){
293         exit(0);
294     }
295     tmp->value = value;
296     tmp->next = NULL;
297     if(q->front == NULL){
298         q->front = q->rare = tmp;
299     }
300     else{
301         q->rare->next = tmp;
302         q->rare = tmp;
303     }
304 }
305
306 int dequeue(struct Queue *q){
307
308     int value;
309     if(q->front == NULL){
310         printf("Queue is empty!\n");
311         return -1;
312     }
313     else{
314         value = q->front->value;
315         struct QNode *tmp;
316         tmp = q->front;
317         q->front = tmp->next;
318
319         if(q->front == NULL){
320             q->rare = NULL;
321         }
322         free(tmp);
323         return value;
324     }
325 }
326 //Derste yazdıklarımızın sonu
327
328 //Breath First Search algoritması defalarca çağırıldığı vakit
329 //Oluşan kirliliği gidermek adına yapılan bir temizleme fonksiyonudur.
330 void destroyQ(struct Queue *queue){
331     while(queue->front != NULL){
332         dequeue(queue);
333     }
334     free(queue);
335 }
336
337 //Kevin Bacon Sayısı İçin Breath First Search Algoritması
338 int BFS_FindKBacon(struct Graph *myGraph, int s, char findWord[MAX2], int
    movieCounter, int step){
339
340     struct Graph *G = myGraph;
341     //Aranan kişi zaten Kevin Bacon ise 0 döndürür.
342     if(!strcmp("Bacon, Kevin", findWord)){
343         return 0;
344     }
345     //Kaçıncı adımda olunduğunu ve ayrıca ziyaret edilip edilmeme durumunu
    tutan dizi
346     int *visited = (int*)malloc(G->numVertices * sizeof(int));
347     int i;
```

```
348     for(i = 0 ; i < G->numVertices ; i++){
349         visited[i] = -1;
350     }
351     //Algoritma gereği ilk eleman kuyruğa koyulur.
352     struct Queue *queue = createQueue();
353     enqueue(queue, s);
354     visited[s] = 0;
355 
356     int v;
357     while(queue->front != NULL){
358         v = dequeue(queue);
359         struct node *tmp = G->adjLists[v];
360         //Kuyruktan çekilen her değer için kuyruğa o değerin graftaki ziyaret
edilmemiş komşuları kuyruğa koyulur.
361         while(tmp){
362             if(visited[tmp->vertex] == -1){
363 
364                 enqueue(queue, tmp->vertex);
365                 visited[tmp->vertex] = visited[v] + 1;
366                 //Geçici olarak tutulan compare değişkeni sürekli ulaşılan
node ile
367                 //karşılaştırılır ve bulunup bulunmadığına karar verilir.
368                 char compare[MAX2];
369                 if(tmp->vertex >= movieCounter){
370                     strcpy(compare, actors[tmp->vertex - movieCounter]);
371                 }
372                 else{
373                     strcpy(compare, movies[tmp->vertex]);
374                 }
375                 if(!strcmp(compare, findWord)){
376                     //Bulunduğunda ise direkt olarak adım sayısı döndürülmez
377                     //Öncelikle recursive çağrılar yapılarak graf üzerinde
Kevin Bacon'dan
378                     //Aranan aktöre giden yol yazdırılır.
379                     if(v >= movieCounter){
380                         if(strcmp(actors[v - movieCounter], "Bacon, Kevin")
== 0){
381                             printf("[ %s ].\n", actors[v - movieCounter]);
382                         }
383                         else{
384                             printf("[ %s ] that played in\n", actors[v -
movieCounter]);
385                         }
386                     }
387                     else{
388                         printf("[ %s ] is the movie, that includes the actor
named\n", movies[v]);
389                     }
390                     step++;
391 
392                     destroyQ(queue);
393                     free(visited);
394                     if(v != s){
395                         char arr[MAX2];
396                         if(v >= movieCounter){
397                             strcpy(arr, actors[v- movieCounter]);
398                         }
399                         else{
400                             strcpy(arr, movies[v]);
401                         }
```

```
402                         //Bu recursive çağrı her bir adım sonunda gelinen adımdan istenen kişiye ulaşmak için
403                         //algoritmayı bir kere daha çağırır ve böylelikle adım adım gidilen yol yazdırılır.
404                         return BFS_FindKBacon(myGraph, s, arr, movieCounter, step);
405                     }
406                     else{
407                         return step;
408                     }
409                 }
410             }
411             tmp = tmp->next;
412         }
413     }
414     //Bütün oyuncular için bir frekans dizisi oluşturulup ekrana yazdırılmak istendiğinde
415     //fonksiyona manuel olarak 'x' gönderilir ve bu karakter oyuncu listesinde olmadığından bütün grafın gezilmesi
416     //sağlanır. Bütün graf gezildiğinde oluşan visited dizisi üzerinden ise frekanslar hesaplanır ve ekrana yazdırılır.
417     if(!strcmp("x",findWord)){
418         int distances[40] = {0};
419         int sum = 0;
420         for(i = 0 ; i < G->numVertices ; i++){
421             if(visited[i] != -1){
422                 distances[visited[i]]++;
423             }
424         }
425         printf("\n\n");
426         //ikişer atlama sebebi BFS adımlarının birinin filmlerden birinin aktörlerden
427         //Şeklinde örüntü ile ilerlemesidir.
428         for(i = 0 ; i < 40 ; i += 2){
429             if(distances[i] != 0){
430                 printf("%d\tof actors have\t%d\tdistance from Kevin Bacon.\n", distances[i], i/2);
431                 sum += distances[i];
432             }
433         }
434         printf("%d\tof actors have infinite distance from Kevin Bacon.\n", (G->numVertices - sum) - movieCounter);
435     }
436 
437     //Temizleme işlemlerinin ardından
438     //Bulunamadıysa -2 döndürülür.
439     destroyQ(queue);
440     free(visited);
441 
442     return -2;
443 }
444 
445 int main(){
446 
447     int movieCounter = 0, actorCounter = 0;
448 
449     //Kullanıcı için kolay ve anlaşılır bir arayüz oluşturulması
450     printf("\n---------------------------------------\nWELCOME TO KEVIN BACON FAN CLUB\n---------------------------------------\n");
451     char nameOfFile[50];
452     printf("Write the name of input file: ");
```

```
    scanf("%s", nameOfFile);

    printf("\nFile Reading Process: \n");
    readFile(&movieCounter, &actorCounter, nameOfFile);

    struct Graph *myGraph = createGraph(movieCounter + actorCounter);
    printf("\nEdge Adding Process: \n");
    myGraph = createEdges(myGraph, movieCounter, actorCounter, nameOfFile);

    int choice;
    do{
        printf("\n\n-------------------------------------\n1 - Print graph\n2 - Find all distances\n3 - Find one actor's distance\n4 - Quit\n-------------------------------\n");
        printf("Make your choice: ");
        scanf("%d", &choice);
        //Graf yazdırma fonksiyonu ile grafın konsola basıldığı fonksiyon
        if(choice == 1){
            printf("\nGraph:\n\n");
            printGraph(myGraph, movieCounter);
        }
        //BFS Fonksiyonuna 'x' gönderilerek frekans dizisinin oluşturulup yazdırılması
        //(Yukarıda fonksiyonun olduğu yerde daha net açıklanmıştır.)
        else if(choice == 2){
            printf("\nDistance Finding Process: \n");
            BFS_FindKBacon(myGraph,findIndex(0, "Bacon, Kevin", actorCounter) + movieCounter, "x", movieCounter, 0);
        }
        //Kullanıcıdan alınan aktör ismi ile birlikte mesafenin bulunup yolun yazırılması
        else if(choice == 3){
            int step = 0;
            char actor[MAX2];
            printf("\nName of actor: ");
            scanf(" %[^\n]s",actor);
            printf("[ %s ] played in\n", actor);
            int distance = BFS_FindKBacon(myGraph,findIndex(0, "Bacon, Kevin", actorCounter) + movieCounter, actor, movieCounter, step);
            if(distance == -2){
                printf("nothing or even he/she played some movies he/she has no connection with Kevin Bacon.");
            }
            else{
                printf("\nSo [ %s ] has [ %d ] distance from Kevin Bacon.\n",actor, distance/2);
            }
        }
        else{
            choice = 4;
        }

    }while(choice != 4);
    printf("\n------------------\ndeveloped by Sadi\n------------------\n");
    return 0;
}
```