VERİ YAPILARI VE ALGORİTMALAR

- DÖNEM ÖDEVİ -

KONU : KEVIN BACON SAYISI

GRUP 1

İrem ATILGAN

17061036

19.05.2020

GİRİŞ

Ödevde main fonksiyonu hariç toplam 20 fonksiyon kullanıldı. Bunlar:

- **QUEUE* createQueue(int vertices) :** Kuyruk oluşturma fonksiyonudur. Parametre olarak düğüm sayısını alır. Oluşturulan QUEUE* yu döner.
- **int isEmpty(QUEUE* q) :** Kuyruğun boş olup olmadığını kontrol eden fonksiyondur. Parametre olarak QUEUE* alır; boşsa 1, doluysa 0 değerini döner.
- **void enqueue(QUEUE* q, int val) :** Kuyruğa eleman ekleyen fonksiyondur. Parametre olarak QUEUE* ve eklenecek değeri alır.
- **int dequeue(QUEUE* q) :** Kuyrukta bekleyen sıradaki elemanı çıkartan fonksiyondur. Parametre olarak QUEUE* alır, kuyruk boş değilse çıkarılan elemanı döner.
- **NODE* createNode(char name[], int type) :** NODE oluşturma fonksiyonudur. Parametre olarak aktör/film ismini ve tipini (Film/Aktör bilgisi. filmse 0, aktörse 1'dir) alır, NODE* döner.
- **Graph* initGraph(int numVertices) :** Grafın başlangıçta oluşturulmasını sağlayan fonksiyondur. Parametre olarak düğüm sayısını alır, Graph* döner.
- **int* BFS(int start,Graph* graph) :** Breadth-First Search algoritmasının uygulandığı fonksiyondur. Başlangıç elemanını ve grafı parametre olarak alır. Düğümler arasındaki bağlantıların tutulduğu int dizisini döner.
- **void printGraph(Graph* graph) :** Graf yazdırma fonksiyonudur. Bu fonksiyon inceleyen kişinin grafı görebilmesini sağlamak amacıyla ekstra olarak eklenmiştir. İstenirse koda eklenerek fonksiyon çalıştırılabilir.
- **unsigned long hash(char* name) :** Hashing işleminin yapıldığı fonksiyondur. Bu fonksiyonda string parametre olarak alınır ve kendisine karşılık gelen değer hesaplanır. Hashleme kısmında djb2 hash fonksiyonundan yararlanıldı. Dönüş değeri üretilen unsigned long sayıdır.
- **int search(char name[],unsigned long* hashtable, int length) :** Gelen film veya aktörün grafta olup olmadığını bulan fonksiyondur. Her string özel bir sayıya karşılık geldiği için aranan isme karşılık düşen sayı bulunur. Hashtable dizisinde key ile aynı değere sahip bir değer bulunursa bulunduğu indis yani node'un ID'si , bulunamazsa -1 dönülür. Parametre olarak aranan string, hash fonksiyonundan çıkan sayıların tutulduğu hashtable dizisi ve toplam düğüm sayısı alınır.
- **Graph* newAllocation(Graph* graph) :** Eğer graf için ayrılan node sayısı yetersiz kalırsa grafı genişletmek için newAllocation fonksiyonundan yararlanılır, 10,000 Düğümlük ek yer ayırır. Parametre olarak graf alınır ve graf dönülür.
- **Graph* addEdge(Graph* graph,char src[],char dst[]) :** Düğümlerin bağlandığı, kenar (edge) oluşturma fonksiyonudur. Parametre olarak graf, film ve aktör ismi alınır. Düzenlenen graf dönülür.
- **NODE* findByID(int ID,Graph* graph) :** Verilen ID'den ID'nin ait olduğu node'u dönen fonksiyondur. Node bulunamazsa NULL dönülür.
- **int* buildPathGeneral(int start,int* prev,Graph* graph) :** Her Kevin Bacon Sayısına sahip kaç aktörün olduğunu bulan fonksiyondur. Parametre olarak başlangıç ID'si (Kevin Bacon'un bulunduğu node'un ID'si), tüm bağlantıların tutulduğu prev dizisi ve graf alınır. Hangi Kevin Bacon Sayısı'na kaç aktörün sahip olduğunu gösteren int dizisi dönülür.
- **int* buildPathActor(int start, int end, int* prev,Graph* graph) :** Spesifik olarak bir aktörün Kevin Bacon Sayısı hesaplanmak istendiğinde buildPathActor fonksiyonu çalıştırılır. Parametre olarak başlangıç ID'si, bağlantısı aranan ID, tüm bağlantıların tutulduğu prev dizisi, ve graf alınır. Bağlantı yolunun tutulduğu int dizisi dönülür.

- **FILE* openFile() :** Dosya açma fonksiyonudur. Açılan dosya dönülür.
- **char* editName(char* name) :** Dosyadaki isimlerin düzenlenmesini sağlayan fonksiyon (örneğin Pitt, Brad -> Brad Pitt'e dönüşür). Parametre olarak aktör ismi alınır, ismin son hali dönülür.
- **Graph* cleanData(char** text, int rows) :** Parametre olarak alınan text'in düzenlenerek grafın oluşturulduğu fonksiyondur. "rows" değişkeni ise text'teki toplam satır sayısını tutar. Oluşturulan graf dönülür.
- **char** readFile(FILE* fp,int* row) :** Dosyadaki bilgilerin **text matrisine aktarıldığı fonksiyondur. Parametre olarak dosya ve toplam satır sayısı alınır, **text matrisi dönülür.
- **char* getName() :** Kullanıcıdan aktör ismi okuyan fonksiyondur. Okunan ismi döner.

Fonksiyonlar kodda yer alan yorum satırlarında daha detaylı açıklanmıştır. İşlem adımlarını genel olarak sıralarsak:

1. main fonksiyonunda ilk olarak dosya açılır, veriler okunur ve matrise yazılır. Ardından cleanData fonksiyonu çağrılarak grafın oluşturulmasında ilk adım atılır.
2. cleanData fonksiyonunda alınan isimler düzenlenir ve addEdge fonksiyonu çağrılarak grafa eklenir.
3. addEdge fonksiyonunda daha önce o isimde aktör/film olup olmadığına search fonksiyonu ile bakılır ve buna bağlı olarak oluşturulan node'lara uygun ID'ler atanır. Bu ID'ler adjList'in kullanılmasını kolaylaştırmak ve node'lara daha hızlı erişmek için kullanılır.
4. search fonkiyonunda ID aranırken *hashtable dizisinden yararlanılır. Hashleme işlemi ile her string özel değerler alarak bu diziye yerleştirilir. Arama yapılırken de bu dizide lineer arama yapılarak tüm grafın gezilmesi gerekliliği ortadan kalkar. Böylece kod daha hızlı ve efektif çalışır.
5. Eğer hashtable dizisinde aranan değer bulunamazsa hash fonksiyonu ile string'e karşılık gelen sayısal değer diziye eklenir. Hash fonksiyonunda, djb2 hashleme fonksiyonundan yararlanılır.
6. Tüm graf oluşturulduktan sonra her Kevin Bacon sayısında kaç aktör olduğunu bulmak için buildPathGeneral fonksiyonu çağrılır.
7. buildPathGeneral'da bir baconCounter dizisi oluşturularak her Kevin Bacon sayısında kaç aktör olduğu bulunur. Sayaç dizisi dönülür.
8. Kullanıcıya tüm bilgiler sunularak spesifik olarak başka bir aktörün Kevin Bacon bilgisini öğrenmek isteyip istemediği sorulur. Kullanıcı isterse aktör ismini aratarak bağlantı yolunu görebilecektir.
9. Bunun için **distances matrisinden ve buildPathActor fonksiyonundan yararlanılır. Eğer aranan aktör, daha önce de aranmışsa distances matrisinden bilgileri çekilerek kolayca kullanıcıya gösterilebilir. İlk defa sorulduysa buildPathActor fonksiyonu çağrılır ve dönülen yol dizisi distances matrisine eklenerek birdahaki sefere sorulduğundan bilginin oradan çekilmesini sağlar.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
typedef struct Node{
    char name[100];     //Film veya aktörün ismi
    int ID;             //Film veya aktörün ID'si (Bu ID her film ve aktör için özeldir)
    int type;           //Film veya aktör olup olmadığının bilgisi (Filmse 0 Aktörse 1'e eşittir)
    struct Node* next;  //Node'un gösterdiği kendinden bir sonraki node
}NODE;

typedef struct queue{
    int* items;         //Kuyruktaki elemanların tutulduğu dizi
    int front,rear;     //Kuyruğun ilk ve en sonundaki elemanı gösteren değişkenler
}QUEUE;


typedef struct Graph{

    int* visited;               //Breadth-First Searh yapılırken grafta gezilen node'ların işaretlenmesini sağlayan dizi

    NODE** adjList;             //Grafta her film ve aktörün linkli listesini tutan matris
    int numOfVertices;          //Grafın düğüm sayısı
    unsigned long *hashtable;   //Aktör veya filmin daha önce grafa eklenip eklenmediğini daha kısa sürede ve kolayca bulmak için
                                //Hashtable kullanıldı. Her aktörün ve filmin ID'si hashtable içerisinde string için üretilen özel bir
                                //sayıyla saklanıyor.
}Graph;
```

1. main

```c
int main()
{
    FILE* fp;
    fp = openFile();    //Dosya açılır

    char key;                               //Kullanıcının aktör aramak isteyip istemediğinin bilgisini tutan değişken

    int rowCount = 0;                       //Textteki satır sayısının tutulduğu değişken
    char** text = readFile(fp,&rowCount);   //Açılan dosyadaki bilgiler okunur


    Graph* graph = cleanData(text,rowCount);//Graf oluşturulur
    int kevinID = search("Kevin Bacon",graph->hashtable,graph->numOfVertices);  //Kevin Bacon'un ID'si bulunur
    int* prev = BFS(kevinID,graph);                                             //BFS uygulanır
    int* kevinNumbers = buildPathGeneral(kevinID,prev,graph);                   //Her K.B. Sayısında toplam kaç aktör olduğu bulunur

    int i;
    int j;

        if(actorID == -1)   //Aktör mevcut değilse
            printf("\nThis actor does not exist in the list.\n");
        else{
            //İlk olarak yolların tutulduğu distances matrisine bakılır, aktörün ID'sinin bulunduğu hücre boşsa
            //Henüz yolu hesaplanmamış demektir, buildPathActor fonksiyonu ile hesaplanır ve tabloya yazılır
            //Boş değilse fonksiyon çağrılmadan tablodan bilgisi alınır
            if(distances[actorID] == NULL)
            {
                path = buildPathActor(kevinID,actorID,prev,graph);
                distances[actorID] = path;
            }
            else
                path = distances[actorID];

            //Bir yol mevcutsa
            if(path)
            {
                for(i = 0; path[i] != -1; i++); //Yoldaki toplam düğüm sayısını bulunur (i) ve length değişkenine atanır
                length = i;
                printf("%s's Kevin Bacon number is %d\n",findByID(path[0],graph)->name,length/2);   //Kevin bacon sayısını bulmak için yol uzunluğu 2'ye bölünür
                                                                                                    //(2 Aktör 1 Film örüntüsü ile ilerlediği için)
                for(i = 0; i < length-2; i+=2)
                    printf("%s - %s  : %s\n",findByID(path[i],graph)->name, findByID(path[i+2],graph)->name,findByID(path[i+1],graph)->name);

                printf("\n");
            }
```

```
654             else
655             {
656                 printf("%s's Kevin Bacon number is infinite\n",findByID(path[0],graph)->name);
657             }
658
659
660         }
661         printf("Would you like to search another Actor's Kevin Bacon number? [y/n] : ");
662         scanf("%c",&key);
663     }
664
665
666     return 0;
667 }
```

2. openFile

```
437 //Dosya açma fonksiyonu
438 FILE* openFile()
439 {
440     FILE* fp = fopen("movies/input-1.txt","r");
441     if(fp == NULL)  //Dosya açılamazsa
442     {
443         printf("File Error!..");
444         exit(0);
445     }
446
447 }
```

3. readFile

```
530 //Dosya okuma fonksiyonu
531 char** readFile(FILE* fp,int* row)
532 {
533     char** text;                            //Dosyanın tamamını yerleştireceğimiz text matrisi
534                                             //Böylece tek seferde text'i alıp işleme sokabileceğiz
535
536     text = malloc(sizeof(char*)*15000);
537     char line[5000];
538
539     while(!feof(fp))
540     {
541         fgets(line,sizeof(line),fp);        //Okunan her satır line değişkenine aktarılır
542         if(line[strlen(line)-1] == '\n')
543             line[strlen(line)-1] = '\0';
544         text[*row] = malloc(sizeof(char)*(strlen(line)+1)); //textte gereksiz sütun sayısının oluşmaması için her satırda
545                                                             //okunan line uzunluğu kadar hücre oluşturulur
546         strcpy(text[*row],line);    //line oluşturulan bölgeye kopyalanır ve yerleştirilir
547
548         (*row)++;
549     }
550
551     fclose(fp);
552     return text;
553 }
```

4. cleanData

```
//Text'in düzenlenerek grafın oluşturulduğu fonksiyon
Graph* cleanData(char** text,int rows)
{
    printf("\nProgram is started. Construction of graph and calculation of Kevin Bacon Numbers may take a while. Please wait...\n");

    int i;      //Satır indislerini tutan değişken

    Graph* graph = initGraph(15000);//Graf hazırda 15,000 düğümle başlatılır


    char* movie;        //Film ismi
    char* name;         //Düzenlenmemiş aktör ismi
    char *input;        //strtok fonksiyonu ile ayıracağımız kelimelerin bellek adresinin tutulduğu değişken
    char* actorName;    //Düzenlenmiş aktör ismi


    for(i = 0; i < rows; i++)
    {

        input = strtok(text[i],"/");    //strtok fonksiyonu ile '/' içeren stringteki kelimeler ayrılır
        movie = strdup(input);          //alınan ilk string film adı olduğundan string duplicate edilerek movie char dizisine yerleştirilir
        while((input = strtok(NULL,"/")) != NULL)   //Tüm satır bitene dek aktör isimleri alınıp grafa eklenmeye devam eder
        {

            //Aktör isimlerine de filme uygulanan işlemler uygulanır
            name = strdup(input);
            actorName = editName(name);

            graph = addEdge(graph,movie,actorName); //Film-Aktör düğümleri edge olarak eklenmek üzere addEdge fonksiyonu çağrılır

        }


        printf("Loading (Building Graph) : %d\r",(i*100/rows)); //Büyük verilerde sürecin görülebilmesi için yazıldı

    }

    printf("\nGraph is constructed!..Please wait for the calculation of total number of actors at each Kevin Bacon Number\n");

    return graph;

}
```

5. addEdge

```
//Düğümlerin bağlandığı, kenar (edge) oluşturma fonksiyonu
Graph* addEdge(Graph* graph,char src[],char dst[])
{
    //static değişkeni newID, gelen her farklı aktör/filme ID atanmasını sağlar
    //Her çağrıldığında kaldığı yerden ID ataması için static yapılmıştır
    static int newID = 0;

    //Eğer toplam düğüm sayısı (newID aynı zamandason düğümün ID'sini tutar) grafın alabileceği
    //düğüm sayısından fazlaysa yeni yer açılması için newAllocation fonksiyonu çağrılır
    if(graph->numOfVertices <= newID)
        graph = newAllocation(graph);


    //Yeni gelen aktör/filmin grafta daha önce olup olmadığını bulabilmek için search fonksiyonu çağrılır
    int ID = search(src,graph->hashtable,newID);

    //Film için node oluşturulur
    NODE* node = createNode(src,0);


    if(ID == -1)    //Grafta bu film yoksa
    {
        graph->hashtable[newID] = hash(src);    //hashtable'ın ID (yeni node'a ait ID) indisli hücresine hash fonksiyonu ile üretilen sayı yerleştirilir
        node->ID = newID++;
    }
```

```
251         else
252             node->ID = ID;  //Grafa film daha önce eklenmişse ID'si alınır ve oluşturulan node'un ID'sine atanır
253 
254 
255         //Aynı işlemler aktör için de uygulanır
256         NODE* node2 = createNode(dst,1);
257         ID = search(dst,graph->hashtable,newID);
258 
259         if(ID == -1)
260         {
261             graph->hashtable[newID] = hash(dst);
262             node2->ID = newID++;
263         }
264         else
265             node2->ID = ID;
268         //  adjList
269         // ----------
270         //    ID    ->  İşaret ettiği node
271         //-----------  --------------------
272         // Aktör ID -> Film
273         // Film  ID -> Aktör olacak şekilde graf düzenlenir
274 
275         node->next = graph->adjList[node2->ID];
276         graph->adjList[node2->ID] = node;
277 
278 
279         node2->next = graph->adjList[node->ID];
280         graph->adjList[node->ID] = node2;
281 
282 
283         return graph;
284 
285 
286     }
```

6. initGraph

```
136 //Grafın initialize edilmesini sağlayan fonksiyon
137 Graph* initGraph(int numVertices)
138 {
139 
140     Graph* graph = (Graph*)malloc(sizeof(Graph));
141     graph->numOfVertices = numVertices;
142     graph->visited = malloc(sizeof(int)*numVertices);
143     graph->adjList = malloc(sizeof(NODE*)*numVertices);
144     graph->hashtable = malloc(sizeof(unsigned long)*numVertices);
145 
146     //Graftaki diziler sıfırlanır, henüz edge eklenmediği için adjList'teki her satır NULL yapılır
147     int i;
148     for(i = 0; i < numVertices; i++)
149     {
150         graph->adjList[i] = NULL;
151         graph->hashtable[i] = 0;
152         graph->visited[i] = 0;
153     }
154 
155     return graph;
156 }
```

7. search

```
174  //Gelen film veya aktörün grafta olup olmadığını bulan fonksiyon
175  int search(char name[],unsigned long *hashtable, int length)
176  {
177
178      int i;
179
180      unsigned long key = hash(name); //Her string özel bir sayıya karşılık geldiği için aranan isme karşılık düşen sayı bulunur
181
182      //Hashtable dizisinde key ile aynı değere sahip bir değer bulunursa bulunduğu indis yani node'un ID'si dönülür
183      for(i = 0; i < length;i++)
184      {
185          if(key == hashtable[i])
186              return i;
187
188      }
189
190      //Bulunamazsa -1 dönülür
191      return -1;
192
193
194  }
```

8. hash

```
158  //Hash fonksiyonu djb2
159  unsigned long hash(char* name)
160  {
161      //Her string için özel bir sayı üretilir. Böylece string arandığında ID'sine çok daha efektif ve hızlı bir şekilde ulaşılabilir
162      unsigned long hash = 5381;
163      int c;
164
165      while (c = *name++){
166          hash = ((hash << 5) + hash) + c; /*hash * 33 + c*/
167      }
168
169      return hash;
170  }
```

9. createNode

```
100  //Node oluşturma fonksiyonu
101  NODE* createNode(char name[],int type)
102  {
103      NODE* node = malloc(sizeof(NODE));
104
105      //Node'un ismi, tipi (film/aktör) atanır ve göstereceği bir sonraki node NULL yapılır
106      strcpy(node->name,name);
107      node->type = type;
108      node->next = NULL;
109
110
111      return node;
112  }
```

10. BFS

```c
//Breadth First Search algoritmasının uygulandığı fonksiyon
int* BFS(int start,Graph* graph)
{

    int* prev; //Gezilen her node'a hangi node'dan ulaşıldığını gösteren prev dizisi
    int v;                      //Kuyruktan çıkarılan elemanların tutulduğu değişken
    int startVertex = start;    //BFS'nin aramaya başlayacağı düğüm (Kevin Bacon)


    QUEUE* q = createQueue(graph->numOfVertices);       //Kuyruk oluşturulur

    prev = malloc(sizeof(int)*(graph->numOfVertices));

    int i;
    for(i = 0; i < graph->numOfVertices; i++)           //Henüz graf gezilmediği için tüm elemanlar -1 yapılır
        prev[i] = -1;


    enqueue(q,startVertex);             //Başlangıç ID'si kuyruğa eklenir ve visited dizisinde gezilmiş olarak işaretlenir
    graph->visited[startVertex] = 1;
    while(!isEmpty(q))                  //Kuyruk boş olmadığı sürece
    {

        v = dequeue(q);                 //Kuyrukta bekleyen sıradaki eleman çıkarılır
        NODE* tmp = graph->adjList[v];

        while(tmp != NULL)
        {

            if(graph->visited[tmp->ID] == 0){   //Node daha önce gezilmemişse kuyruğa eklenir ve gezilmiş olarak işaretlenir

                enqueue(q,tmp->ID);
                graph->visited[tmp->ID] = 1;
                prev[tmp->ID] = v;
            }
            tmp = tmp->next;    //Komşu node'lar gezilir
        }

    }

    free(q);        //BFS bittiğinde kuyruk (QUEUE) bellek alanından temizlenir


    return prev;    //Tüm yollara ulaşmamızı sağlayan prev dizisi return edilir
}
```

11. enqueue & dequeue

```
75  //Kuyruğa yeni film/aktör ekleme fonksiyonu. Kuyrukta film/aktörün ID'si tutulur
76  void enqueue(QUEUE* q,int val)
77  {
78      //Kuyruk boşsa ilk elemanı gösteren değişken 0 yapılır
79      if(isEmpty(q))
80          q->front = 0;
81      //Sıradaki son elemanı gösteren değişken bir arttırılır
82      q->rear++;
83      q->items[q->rear] = val;//Son elemanın olduğu göze yeni gelen ID yerleştirilir
84
85  }
86
87
88  //Kuyruktan eleman çıkarma fonksiyonu
89  int dequeue(QUEUE* q)
90  {
91      //Kuyruk boşsa eleman çıkarılamaz, bu yüzden -1 dönülür
92      if(isEmpty(q))
93          return -1;
94
95      return q->items[q->front++];    //Kuyruk boş değilse sırada bekleyen ilk eleman dönülür, front kuyrukta gelen
96                                      //sıradaki elemanı göstermesi için ayarlanır
97  }
98
```

12. createQueue & isEmpty

```
52  //Kuyruk oluşturma fonksiyonu
53  QUEUE* createQueue(int vertices)
54  {
55      QUEUE* q = malloc(sizeof(QUEUE));
56
57      q->items = malloc(sizeof(int)*vertices);    //Düğüm sayısı kadar kuyruk dizisinde yer ayırılır
58      q->front = -1;                              //Henüz kuyruk boş olduğundan, bunu belli etmek amacıyla
59      q->rear = -1;                               //ilk ve son elemanı gösteren değişkenler (front,rear) -1 yapılır
60
61      return q;
62  }
63
64  //Queue'nun boş olup olmadığını öğrenmek için oluşturulan fonksiyon
65  int isEmpty(QUEUE* q)
66  {
67      if(q->front == -1 || (q->front > q->rear))  //Kuyruk hiç oluşturulmamışsa sıradaki elemanı gösteren değişken (front) -1'dir
68          return 1;                               //Kuyruk oluşturulmuş, kuyruktaki tüm elemanlar gezilmişse front kuyruktaki son elemanı gösteren
69                                                  //değişkenden (rear) büyük olur. Bu iki durum da kuyruğun boş olduğunu gösterir
70
71      return 0;   //Kuyruk boş değilse
72  }
73
```

13. buildPathGeneral

```
313  //Her Kevin Bacon Sayısına sahip kaç aktörün olduğunu bulan fonksiyon
314  int* buildPathGeneral(int start, int* prev,Graph* graph)
315  {
316      int index;
317      int i;
318      int j;
319      NODE* tmp;
320
321      //Kevin Bacon Sayılarındaki aktörleri saymak için baconCounter dizisi oluşturulur ve sıfırlanır
322      int* baconCounter = malloc(sizeof(int)*(graph->numOfVertices));
323      for(i = 0; i < graph->numOfVertices; i++)
324          baconCounter[i] = 0;
325
326
327      for(j = 0; graph->adjList[j] != NULL; j++)
328      {
329          tmp = findByID(j,graph);    //Aktör sayılarını hesaplayacağımız için elimizdeki node'un film/aktör bilgisine ulaşmalıyız
330                                      //Bu yüzden elimizdeki ID'den ait olduğu node'u bulur ve type'ını kontrol ederiz
331
332          if(tmp->type)   //Aktörse (type = 1 aktör, type = 0 film)
333          {
334              //Aktörden Kevin Bacon'a ulaşmak istiyoruz, bu yüzden başlangıcı aktörün ID'si yaparız
335              //prev dizisi o ID'ye hangi ID'den ulaştığımızı bulmamızda yardımcı olur. Kevin Bacon'un ID'sine veya bağlantının bittiği (-1)
336              //bir noktaya ulaşıncaya dek geriye gidilir -> index = prev[index]
337              for(index = j; index != -1 && start != index; index = prev[index])
339
340             //Kevin Bacon'ın ID'sine ulaşılmışsa sayacın, bulunan kevin bacon numaralı hücresi arttırılır
341             //Sayacın 0. gözü sonsuz Kevin Bacon Sayısı'na sahip aktörler için ayrılmıştır
342             if(index == start){
343                 baconCounter[1+(i/2)]++;    //i yolun uzunluğunu gösterir ancak bu uzunluğun içerisinde filmlerin ID'leri de katılmıştır.
344                                             //İki aktörü bağlayan bir film olduğundan toplam uzunluğun 2'ye bölünmesiyle K.B. Sayısı elde edilebilir
345             }
346             else
347                 baconCounter[0]++;          //Kevin Bacon'a giden bir yol bulunamamışsa sonsuz sayıların toplandığı 0. göz arttırılır
348
349
350         }
351
352
353         i=0;
354
355     }
356     printf("The total number of actors at each Kevin Bacon Number is calculated!..\n\n");
357
358     return baconCounter;
359 }
```

14. findByID

```
289  //Verilen ID'den ID'nin node'unu dönen fonksiyon
290  NODE* findByID(int ID,Graph* graph)
291  {
292
293      NODE* tmp;
294
295      //ID'nin gösterdiği ilk node'un ID'sini kullanarak adjList teki gözüne gidebilir ve aradığımız node'a ulaşabiliriz
296      //Detaylı Anlatım :
297      //Bağlantılar çift yönlü (bipartite) olduğu için Aktör->Film ise Film->Aktör olacaktır
298      //Bu durumda Aktör'ün node'unu bulmak istiyorsak gösterdiği filmin linkli listesine gidebilir ve
299      //orada aradığımız ID'nin node'unu bulabiliriz
300      tmp = graph->adjList[graph->adjList[ID]->ID];
301      while(tmp != NULL && tmp->ID != ID)
302          tmp = tmp->next;
303
304      //Linkli liste bitmemişse aradığımız node'u bulmuşuz demektir
305      if(tmp != NULL)
306          return tmp;
307
308      //Bitmişse node bulunamamış demektir, NULL dönülür
309      return NULL;
310  }
```

15. getName

```c
//Kullanıcıdan aktör ismi okuyan fonksiyon
char* getName()
{
    char* name = malloc(sizeof(char)*100);

    printf("\nActor Name : \n");
    getchar();          //Önceki işlemden gelen '\n' karakterini yakalamak için getchar() ektra çağrılır

    char c = getchar();
    int i = 0;
    while(c != '\n')    //ENTER'a basılana dek isim alınır
    {
        name[i++] = c;
        c = getchar();
    }
    name[i] = '\0';

    return name;
}
```

16. buildPathActor

```c
//Spesifik olarak bir aktörün K.B. Sayısı hesaplanmak istendiğinde buildPathActor fonksiyonu çalıştırılır
int* buildPathActor(int start, int end, int* prev, Graph* graph)
{
    int index;
    int i;
    //Yol bilgilerini (yol üzerindeki ID'leri) de tutmamız gerektiğinden path isimli bir int dizi oluşturulur
    int* path = malloc(sizeof(int)*(graph->numOfVertices));

    //Henüz yol oluşturulmadığından dizideki her eleman -1 yapılır
    for(i = 0; i < graph->numOfVertices; i++)
        path[i] = -1;

    i = 0;

    //buildPathGeneral'da bahsi geçen mantıkla Kevin Bacon bulunana veya yol bitene kadar geriye gidilir
    for(index = end; index != -1 && start != index; index = prev[index])
        path[i++] = index;

    //Kevin Bacon'a ulaşıldıysa
    if(index == start)
    {
        path[i++] = index;
        return path;
    }
    free(path); //Ulaşılamadıysa path'in kapladığı bellek alanı free edilir ve NULL dönülür
    return NULL;
}
```

17. newAllocation

```
198  //Eğer graf için ayrılan node sayısı yetersiz kalırsa grafı genişletmek için newAllocation fonksiyonundan yararlanılır
199  Graph* newAllocation(Graph* graph)
200  {
201      //10,000 düğümlük yer açılır
202      int newsize = (graph->numOfVertices)+10000;
203      //adjList e eklenecek yeni linkli listeler için realloc fonksiyonu ile yeni yer eklenir
204      graph->adjList = realloc(graph->adjList,newsize*sizeof(NODE*));
205      //Ziyaret edilecek node sayısı da arttığından visited dizisine yeni yer eklenir
206      graph->visited = realloc(graph->visited,newsize*sizeof(int));
207
208      //Hashtable dizisi de farklı stringler geleceği için genişletilir
209      graph->hashtable = realloc(graph->hashtable,newsize*sizeof(unsigned long));
210
211      int i;
212      //Grafa eklenen yeni yerler sıfırlanır
213      for(i = graph->numOfVertices; i < newsize;i++)
214      {
215          graph->adjList[i] = NULL;
216          graph->hashtable[i] = 0;
217          graph->visited[i] = 0;
218      }
219
220      //Grafın yeni boyutu düğüm sayısı değişkenine atanır
221      graph->numOfVertices = newsize;
222      return graph;
223  }
```

18. editName

```
449  //Dosyadaki isimlerin düzenlenmesini sağlayan fonksiyon (Pitt, Brad -> Brad Pitt)
450  char* editName(char* name)
451  {
452
453      int length = strlen(name)+1; // +1 -> '\0' için ismin uzunluğuna eklendi
454      int i;
455
456      char* lastname = malloc(sizeof(char)*length);
457      char* firstname = malloc(sizeof(char)*length);
458      char* actorName = malloc(sizeof(char)*length);
459
460      for(i = 0; i < length && name[i] != ',';i++)    //Text dosyasında ilk aktörün soyadı yazdığından, virgül ile karşılaşıncaya kadar
461          lastname[i] = name[i];                      //veya isim bitene dek (Sadece ismi yazan aktörler olduğundan) lastname değişkenine isim atanır
462
463      lastname[i] = '\0';
464
465      i+= 2;  //virgül ve boşluk atlanır
466
467      int j= 0;
468      while(i < length)
469          firstname[j++] = name[i++]; //Aktör ismi, hala harf kaldıysa oluşturulur
470
471      //j sıfır olarak initialize edildiğinden, eğer ilk isime bir şey atanamadıysa değeri sıfır olarak kalacaktır
472      if(j)   //isim ve soyisim bilgilerine sahipsek
473      {
474          strcpy(actorName,firstname);    //Aktörün tam adının yazacağı actorName değişkenine ilk ismi kopyalanır
475          strcat(actorName," ");          //Boşluk eklenir
476          strcat(actorName,lastname);     //Son olarak soyismi eklenir
477      }
478      else    //sadece ismi veya soyismi varsa
479          strcpy(actorName,lastname);     //actorName'e kopyalanır
480
481
482      return actorName;
483  }
```

19. printGraph

```
114 //Graf yazdırma fonksiyonu. Bu fonksiyon inceleyen kişinin grafı görebilmesini sağlamak amacıyla ekstra olarak eklenmiştir
115 //İstenirse koda eklenerek fonksiyon çalıştırılabilir
116 void printGraph(Graph* graph)
117 {
118     int i;
119     printf("\n");
120     NODE* temp;
121     for(i = 0; graph->adjList[i] != NULL; i++)
122     {
123         printf("%d - > ",i);
124         temp = graph->adjList[i];
125         while(temp != NULL)
126         {
127             printf("%s/",temp->name);
128             temp = temp->next;
129         }
130         printf("\n");
131     }
132 }
```

SONUÇ

```
Program is started. Construction of graph and calculation of Kevin Bacon Numbers may take a while. Please wait...
Loading (Building Graph) : 97
Graph is constructed!..Please wait for the calculation of total number of actors at each Kevin Bacon Number
The total number of actors at each Kevin Bacon Number is calculated!..

Total number of actors whose Kevin Bacon Number is 0 -> 1       Actors
Total number of actors whose Kevin Bacon Number is 1 -> 1494    Actors
Total number of actors whose Kevin Bacon Number is Infinite -> 3        Actors
Other Kevin Bacon numbers does not exist.


Would you like to search another Actor's Kevin Bacon number? [y/n] : _
```

Şekil 1: input-1.txt Sonuçları

```
Program is started. Construction of graph and calculation of Kevin Bacon Numbers may take a while. Please wait...
Loading (Building Graph) : 99
Graph is constructed!..Please wait for the calculation of total number of actors at each Kevin Bacon Number
The total number of actors at each Kevin Bacon Number is calculated!..

Total number of actors whose Kevin Bacon Number is 0 -> 1       Actors
Total number of actors whose Kevin Bacon Number is 1 -> 61      Actors
Total number of actors whose Kevin Bacon Number is 2 -> 25      Actors
Total number of actors whose Kevin Bacon Number is 3 -> 36      Actors
Total number of actors whose Kevin Bacon Number is 4 -> 120     Actors
Total number of actors whose Kevin Bacon Number is 5 -> 182     Actors
Total number of actors whose Kevin Bacon Number is 6 -> 248     Actors
Total number of actors whose Kevin Bacon Number is 7 -> 76      Actors
Total number of actors whose Kevin Bacon Number is 8 -> 302     Actors
Total number of actors whose Kevin Bacon Number is 9 -> 237     Actors
Total number of actors whose Kevin Bacon Number is 10 -> 43     Actors
Total number of actors whose Kevin Bacon Number is 11 -> 6      Actors
Total number of actors whose Kevin Bacon Number is Infinite -> 1176     Actors
Other Kevin Bacon numbers does not exist.


Would you like to search another Actor's Kevin Bacon number? [y/n] :
```

Şekil 2: input-2.txt Sonuçları

```
Program is started. Construction of graph and calculation of Kevin Bacon Numbers may take a while. Please wait...
Loading (Building Graph) : 99
Graph is constructed!..Please wait for the calculation of total number of actors at each Kevin Bacon Number
The total number of actors at each Kevin Bacon Number is calculated!..

Total number of actors whose Kevin Bacon Number is 0 -> 1       Actors
Total number of actors whose Kevin Bacon Number is 1 -> 1372    Actors
Total number of actors whose Kevin Bacon Number is 2 -> 93794   Actors
Total number of actors whose Kevin Bacon Number is 3 -> 72974   Actors
Total number of actors whose Kevin Bacon Number is 4 -> 1636    Actors
Total number of actors whose Kevin Bacon Number is 5 -> 14      Actors
Total number of actors whose Kevin Bacon Number is Infinite -> 717      Actors
Other Kevin Bacon numbers does not exist.


Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Meryl Streep
Meryl Streep's Kevin Bacon number is 1
Meryl Streep - Kevin Bacon  : River Wild, The (1994)

Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Nicolas Cage
Nicolas Cage's Kevin Bacon number is 2
Nicolas Cage - Matt Dillon  : Rumble Fish (1983)
Matt Dillon - Kevin Bacon  : Wild Things (1998)

Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Elie Samaha
Elie Samaha's Kevin Bacon number is 3
Elie Samaha - Tia Carrere  : 20 Dates (1998)
Tia Carrere - James Pickens Jr.  : Hostile Intentions (1994)
James Pickens Jr. - Kevin Bacon  : Sleepers (1996)

Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Dakota Fanning
Dakota Fanning's Kevin Bacon number is 2
Dakota Fanning - Laura Dern  : I Am Sam (2001)
Laura Dern - Kevin Bacon  : Novocaine (2001)

Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Adile Naşit

This actor does not exist in the list.
Would you like to search another Actor's Kevin Bacon number? [y/n] :
```

Şekil 3: input-3.txt Sonuçları

input-2.txt dosyasındaki aktörlerin isimleri aratılarak Kevin Bacon sayıları ve bağlantı yolları çıktı olarak alınmıştır:

```
Actor Name :
Mel Blanc
Mel Blanc's Kevin Bacon number is infinite
Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Lauren Graham
Lauren Graham's Kevin Bacon number is infinite
Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Mark Wahlberg
Mark Wahlberg's Kevin Bacon number is 6
Mark Wahlberg - Danny DeVito  : America: A Tribute to Heroes (2001)
Danny DeVito - Arnold Schwarzenegger  : Last Action Hero (1993)
Arnold Schwarzenegger - James Coburn  : Arnold Schwarzenegger: Hollywood Hero (1999)
James Coburn - Harry Guardino  : Hell Is for Heroes (1962)
Harry Guardino - Kevin McCarthy  : Hell with Heroes, The (1968)
Kevin McCarthy - Kevin Bacon  : Hero at Large (1980)

Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Morgan Lofting
Morgan Lofting's Kevin Bacon number is 6
Morgan Lofting - Frank Welker  : G.I. Joe: A Real American Hero (1983)
Frank Welker - Keith David  : Gargoyles: The Heroes Awaken (1994)
Keith David - James Coburn  : Arnold Schwarzenegger: Hollywood Hero (1999)
James Coburn - Harry Guardino  : Hell Is for Heroes (1962)
Harry Guardino - Kevin McCarthy  : Hell with Heroes, The (1968)
Kevin McCarthy - Kevin Bacon  : Hero at Large (1980)

Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Larna Scott

This actor does not exist in the list.
Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Lorna Scott
Lorna Scott's Kevin Bacon number is infinite
Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Hugh Trevor
Hugh Trevor's Kevin Bacon number is infinite
Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Mika Yamada
Mika Yamada's Kevin Bacon number is infinite
```

```
Actor Name :
Ben Hiller
Ben Hiller's Kevin Bacon number is infinite
Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Clancy Brown
Clancy Brown's Kevin Bacon number is 5
Clancy Brown - Keith David  : Gargoyles: The Heroes Awaken (1994)
Keith David - James Coburn  : Arnold Schwarzenegger: Hollywood Hero (1999)
James Coburn - Harry Guardino  : Hell Is for Heroes (1962)
Harry Guardino - Kevin McCarthy  : Hell with Heroes, The (1968)
Kevin McCarthy - Kevin Bacon  : Hero at Large (1980)

Would you like to search another Actor's Kevin Bacon number? [y/n] : y

Actor Name :
Pete Duel
Pete Duel's Kevin Bacon number is 2
Pete Duel - Kevin McCarthy  : Hell with Heroes, The (1968)
Kevin McCarthy - Kevin Bacon  : Hero at Large (1980)
```