

# Structuri de date – Tema 3

Seria CC

**Deadline hard: 24.05.2021**

**Responsabili:** Veronica Radu, Alexandru Rotaru

**Autor:** [Mihai Nan](#)



Facultatea de Automatică și Calculatoare  
Universitatea Politehnica din București  
Anul universitar 2020 - 2021

# 1 Obiective

În urma realizării acestei teme, studentul va fi capabil:

- să implementeze și să utilizeze grafuri în rezolvarea unor probleme;
- să implementeze algoritmi de prelucrare a grafurilor;
- să transpună o problemă din viața reală într-o problemă care uzitează teoria grafurilor;
- să rezolve o problemă de algoritmică, folosind noțiunile și algoritmi studiați la curs pentru grafuri.

# 2 Descriere generală



Filmul este o formă de artă care prezintă o poveste printr-o succesiune de imagini, lasând iluzia unei mișcări continue. Se spune despre film că este o formă importantă de divertisment, o foarte populară alegere pentru petrecerea timpului liber și o puternică metodă de educație sau îndoctrinare. Trăind într-o lume a imaginii, suntem, fără îndoială, mari cinefili. Astfel, această temă are scopul de a vă introduce în vasta lume a filmului, îmbinând, într-un mod plăcut, două pasiuni ale voastre: filmul și grafurile.

Existând o gamă largă de filme și, implicit, un număr foarte mare de actori care au contribuit la realizarea acestor filme, este greu de reținut o colecție vastă. În cadrul acestei teme, o să modelați o colecție de filme, uzitând o structură de date avantajoasă, și o să rezolvați o serie de cerințe.

Pornind de la ideea unui joc celebru în lumea cinematografică: *Cele șase grade ale lui Kevin Bacon*; o să definim algoritmul de construcție a grafului ce va modela colecția noastră. Actorul american Kevin Bacon este unul dintre cei mai apreciați actori de la Hollywood, având la activ zeci de filme celebre și seriale. Grație portofoliului său bogat și carierei îndelungate, trei studenți americani au elaborat în anul 1994 ipoteza potrivit căreia prolificul actor Kevin Bacon ar putea fi *înrudit*, din punct de vedere cinematografic, cu orice actor din lume, prin intermediul unui lanț ce include cel mult șase filme.

# 3 Algoritmul de creare a grafului

Se dă un fișier text care conține informații despre o colecție de filme. Fiecare actor reprezintă un nod din graful care modelează colecția, iar doi actori sunt conectați printr-o muchie dacă au fost implicați în realizarea unui film comun. Pentru o mai bună înțelegere a algoritmului de creare a grafului, se va oferi un exemplu detaliat.

*Gradul de înrudire* a doi actori este egal cu distanța minimă dintre cele două noduri ale grafului, calculată ca număr de muchii parcurse. Dacă doi actori nu sunt conectați printr-o serie de filme, aceștia au *gradul de înrudire* egal cu 0.

Definim o *producție* ca fiind o serie de filme în realizarea cărora au contribuit numai actori înrudiți (cu gradul de înrudire mai mare ca 0).

## 4 Detalierea reprezentării

### 4.1 Informații

Se pornește de la următoarea colecție:

Film1	Film2	Film3	Film4
<ul style="list-style-type: none"><li>• Actor1</li><li>• Actor2</li><li>• Actor3</li></ul>	<ul style="list-style-type: none"><li>• Actor1</li><li>• Actor3</li><li>• Actor4</li></ul>	<ul style="list-style-type: none"><li>• Actor2</li><li>• Actor4</li><li>• Actor5</li></ul>	<ul style="list-style-type: none"><li>• Actor6</li><li>• Actor7</li><li>• Actor8</li></ul>

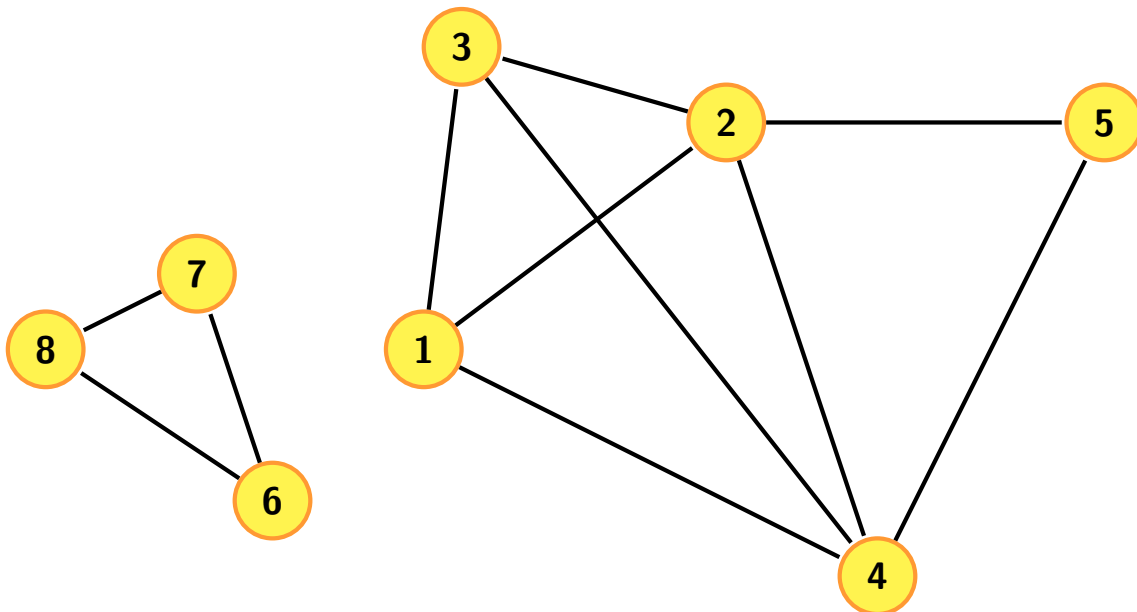
### 4.2 Explicații

Pentru simplitate și o mai bună reprezentare, vom considera că fiecare nod este reprezentat printr-un nod ce are ca etichetă un număr (numărul existent și în numele acestuia).

După cum se poate deduce, în exemplul expus există două producții. Prima producție conține primele 3 filme (*Film1*, *Film2* și *Film3*), iar a doua producție conține doar ultimul film (*Film4*).

Gradul de înrudire pentru actorii *Actor1* și *Actor5* este 2, în timp ce pentru actorii *Actor1* și *Actor7* este 0.

### 4.3 Reprezentare

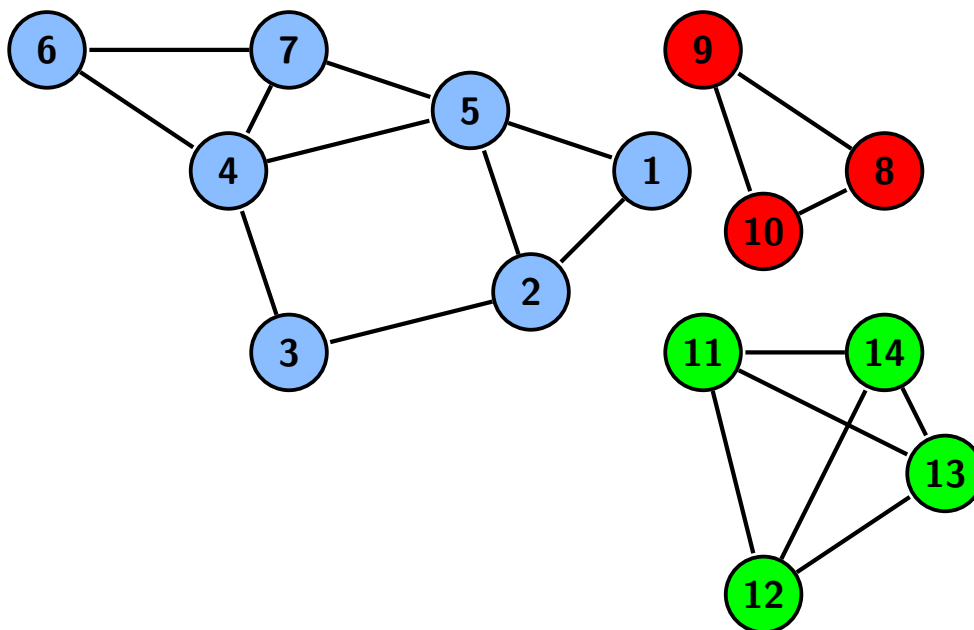


## 5 Cerințe

### 5.1 Cerința 1

Dorim să aflăm care este distribuția cu numărul maxim de actori existentă într-o colecție de filme pusă la dispoziție.

Pentru a înțelege mai clar ce trebuie realizat la această cerință, pornim de la analiza următorului exemplu care conține 3 producții, actorii din fiecare producție fiind colorați cu una dintre culorile *albastru*, *roșu*, *verde*.



Pentru acest exemplu, producția maximă este cea care conține nodurile colorate cu **albastru**.

Rezultatul pentru această cerință o să cuprindă numărul de actori care fac parte din producția cea mai mare și numele actorilor care fac parte din această producție, în ordine crescătoare.

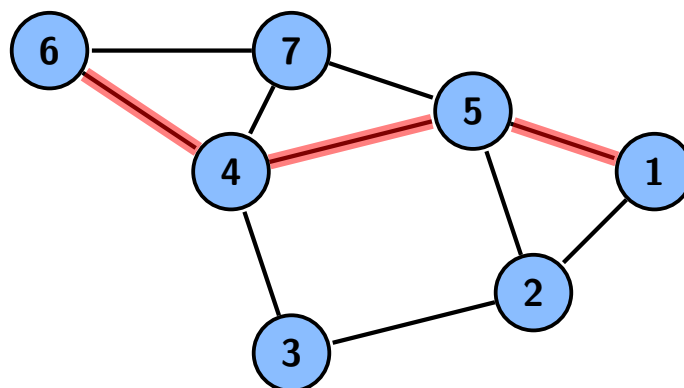
#### Important

Este **obligatoriu** ca rezolvarea acestei cerințe să fie realizată folosind o structură de date de tip **graf** reprezentat uzitând liste de adiacență!

## 5.2 Cerința 2

Pentru această cerință, dorim să determinăm gradul de înrudire dintre doi actori, pe baza jocului *Cele șase grade ale lui Kevin Bacon*. Dacă actorul  $X$  a jucat într-un film împreună cu actorul  $Y$ , atunci gradul de înrudire dintre  $X$  și  $Y$  este 1. Dacă actorul  $X$  nu a jucat împreună cu actorul  $Y$  într-un film, dar actorul  $X$  a făcut parte din distribuția unui film împreună cu  $Z$ , iar  $Y$  a jucat la rândul lui, într-un alt film, cu  $Z$ , atunci gradul de înrudire pentru  $X$  și  $Y$  este 2.

Considerând drept exemplul graful de mai jos, gradul de înrudire dintre Actorul1 și Actorul6 este 3.



#### Important

Este **obligatoriu** ca rezolvarea acestei cerințe să fie realizată folosind o structură de date de tip **graf** reprezentat uzitând liste de adiacență!

### 5.3 Cerința 3

#### Definiții

1. Se numește **lanț** într-un graf neorientat, o secvență de vârfuri  $[x_1, x_2, \dots, x_k]$ , cu proprietatea că oricare două vârfuri consecutive din secvență sunt adiacente.
2. Se numește **ciclu** într-un graf un lanț  $v_1, v_2, \dots, v_k$  cu proprietatea că  $v_1 = v_k$  și muchiile  $(v_1, v_2)$ ,  $(v_2, v_3), \dots, (v_{k-1}, v_k)$  sunt distincte două câte două.
3. Un graf se numește **conex** dacă pentru oricare două vârfuri  $x$  și  $y$  diferite ale sale, există un lanț care le leagă.
4. Se numește **componentă conexă** a unui graf  $G = (V, E)$  un subgraf  $G_1 = (V_1, E_1)$  conex al lui  $G$  care are proprietatea că nu există niciun lanț în  $G$  care să lege un vârf din  $V_1$  cu un vârf din  $V \setminus V_1$ .
5. O **punte** sau o **muchie critică** (*bridge*) este o muchie a unui graf a cărei ștergere ar crește numărul de componente conexe ale grafului rezultat prin eliminarea sa.

O informație care ar putea fi interesantă pentru o persoană pasionată de filme și grafuri este aceea legată de numele actorilor între care există o punte în graful construit pe baza raționamentului prezentat anterior. De aceea, în cadrul acestei cerințe voi va trebui să determinați actorii între care există o punte în graful distribuției. Pentru determinarea punților într-un graf neorientat se folosește o parcurgere în adâncime modificată, reținându-se informații suplimentare pentru fiecare nod și folosind următoarea observație: **punțile sunt muchiile care nu apar în niciun ciclu**. Acest algoritm a fost identificat de către **Tarjan**.

#### Algoritmul lui Tarjan

Vom porni de la următoarele notații:

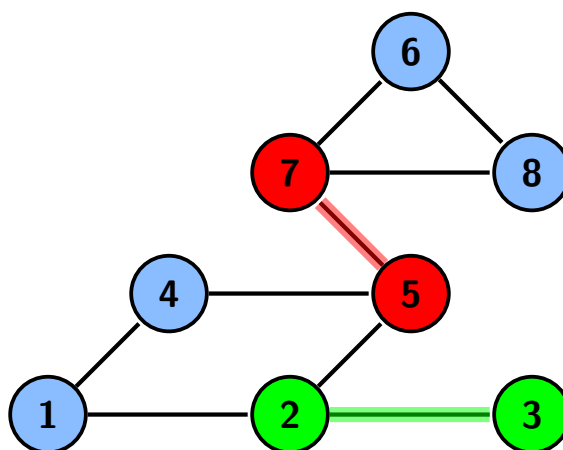
$$idx[u] = \text{timpul de descoperire a nodului } u$$

$$low[u] = \min(\{idx[u]\} \cup \{idx[v] \mid (u, v) \text{ este o muchie înapoi}\} \cup \{low[v] \mid v \text{ copil al lui } u \text{ în arborele de adâncime}\})$$

Folosind aceste notații, condiția pentru ca o muchie  $(v, u)$  să fie punte în graf se poate scrie:

$$(v, u) \text{ este punte} \Leftrightarrow low[u] > idx[v], \text{ iar } u \text{ nu este părintele lui } v \text{ în arborele rezultat în urma parcurgerii în adâncime}$$

În graful din figura de mai jos, între nodurile colorate cu roșu există o punte și între cele colorate cu verde.



Actorii între care există o **punte** în graful producțiilor cinematografice vor fi afișați în ordine alfabetică.

#### Important

Este **obligatoriu** ca rezolvarea acestei cerințe să fie realizată folosind o structură de date de tip **graf** reprezentat uzitând liste de adiacență!

**Important**

O structură de date avantajoasă pentru menținerea datelor în ordine, căutarea, inserarea și eliminarea unei valori este **arborele binar de căutare**.

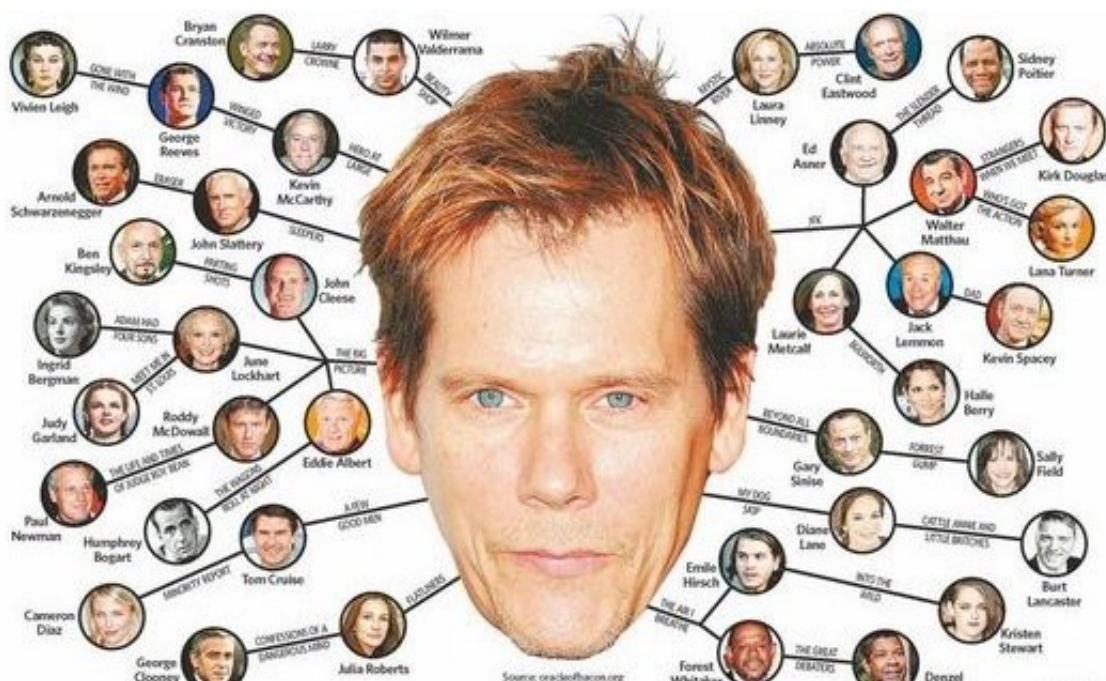
---

**Algorithm 1** Puncti

```

1: procedure PUNȚI( $G$ )
2:    $(V, E) \leftarrow G$ 
3:    $timp \leftarrow 0$ 
4:   for each  $v \in V$  do
5:      $idx[v] \leftarrow -1$ 
6:      $low[v] \leftarrow \infty$ 
7:      $\pi[v] \leftarrow \text{nil}$  ▷  $\pi[v]$  părintele nodului  $v$  în arborele rezultat în urma parcurgerii DFS
8:   for each  $v \in V$  do
9:     if  $idx[v] == -1$  then
10:       dfsB( $G, v, timp, idx, low, \pi$ )
11: procedure DFSB( $G, v, timp, idx, low, \pi$ )
12:    $(V, E) \leftarrow G$ 
13:    $idx[v] \leftarrow timp$ 
14:    $low[v] \leftarrow timp$ 
15:    $timp \leftarrow timp + 1$ 
16:    $copii \leftarrow \emptyset$ 
17:   for each  $(v, u) \in E$  do
18:     if  $u \neq \pi[v]$  then
19:       if  $idx[u] == -1$  then ▷ înseamnă că nodul  $u$  este nedescoperit, deci alb
20:          $\pi[u] \leftarrow v$ 
21:         dfsB( $G, u, timp, idx, low, \pi$ )
22:          $low[v] \leftarrow \min(low[v], low[u])$ 
23:         if  $low[u] > idx[v]$  then
24:           ( $v, u$ ) este punte
25:       else ▷ înseamnă că nodul  $u$  este descoperit, deci gri, iar muchia  $v \rightarrow u$  este muchie înapoi
26:          $low[v] \leftarrow \min(low[v], idx[u])$ 

```



## 6 Formatul fișierelor

Fișierul de intrare va conține, pe prima linie, un număr natural **Nr**, reprezentând numărul de filme din colecție. Pe următoarele linii se vor găsi informațiile despre cele **Nr** filme. Pentru fiecare film se va specifica numele filmului, numărul de actori implicați în realizarea acelui film și o listă a actorilor (câte unul pe rând).

Pentru **cerința 1**, acest fișier nu va conține informații suplimentare, iar fișierul de ieșire va conține pe prima linie un singur număr, reprezentând numărul de actori care fac parte din producția maximală, iar pe următoarele linii se vor găsi numele acestor actori, afișați în ordine crescătoare și câte unul pe linie.

Pentru **cerința 2**, fișierul de intrare va conține, la sfârșit două linii suplimentare. Pe penultima linie din fișier și pe ultima se vor afla numele actorilor pentru care ne dorim să determinăm gradul de înrudire.

Pentru **cerința 3**, nu vom avea informații suplimentare în fișierul de intrare, iar în fișierul de ieșire se va afișa pe prima linie numărul de punți din graful construit, iar pe următoarele linii se vor afișa numele actorilor între care există o muchie a cărei eliminare duce la creșterea numărului de componente conexe în graful producției. Numele actorilor se vor afișa în ordine alfabetică.

## 7 Restricții și precizări

Temele trebuie să fie încărcate pe [vmchecker](#). **NU** se acceptă teme trimise pe e-mail sau altfel decât prin intermediul vmchecker-ului.

O rezolvare constă într-o arhivă de tip **zip** care conține toate fișierele sursă alături de un **Makefile**, ce va fi folosit pentru compilare, și un fișier **README**, în care se vor preciza detaliile implementării.

**Makefile**-ul trebuie să aibă obligatoriu regulile pentru **build** și **clean**. Regula build trebuie să aibă ca efect compilarea surselor și crearea binarului **movies**.

Programul vostru va primi, ca argumente în linia de comandă, numele fișierului de intrare și a celui de ieșire, dar și o opțiune, pentru fiecare cerință în parte, în felul următor:

```
./movies [-c1 | -c2 | -c3] [fișier_intrare] [fișier_iesire]
```

## 8 Punctaj

Cerința	Punctaj
<b>Cerința 1</b>	30 de puncte
<b>Cerința 2</b>	30 de puncte
<b>Cerința 3</b>	40 de puncte

### Atentie!

Orice rezolvare care nu conține structurile de date specificate nu este punctată.  
Temele vor fi punctate doar pentru testele care sunt trecute pe vmchecker.  
Nu lăsați warning-urile nerezolvate, deoarece veți fi depunțați.  
Dealocați toată memoria alocată pentru reținerea informațiilor, deoarece se vor depuncta pierderile de memorie.

**Tema este individuală! Toate soluțiile trimise vor fi verificate, folosind o unealtă pentru detectarea plagiatului.**