Structuri de date – Tema 3

Seria CD

Deadline hard: 17.05.2020

Responsabili: Mihai Nan, Ștefania Ghiță

Facultatea de Automatică şi Calculatoare
Universitatea Politehnica din Bucureşti
Anul universitar 2019 - 2020

1 Obiective

În urma realizării acestei teme, studentul va fi capabil:

- să implementeze şi să utilizeze grafuri în rezolvarea unor probleme;
- să implementeze algoritmi de prelucrare a grafurilor;
- să transpună o problemă din viaţa reală într-o problemă care uzitează teoria grafurilor;
- să rezolve o problemă de algoritmică, folosind noţiunile şi algoritmii studiaţi 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 mişcări continue. Se spune despre film că este o formă importantă de divertisment, o foarte populară alegere pentru petrecerea timpului liber şi o puternică metodă de educaţie 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 şi grafurile.

Existând o gamă largă de filme şi, implicit, un număr foarte mare de actori care au contribuit la realizarea acestor filme, este greu de reţinut o colecţie vastă. În cadrul acestei teme, o să modelaţi o colecţie de filme, uzitând o structură de date avantajoasă, şi o să rezolvaţi o serie de cerinţe.

Pornind de la ideea unui joc celebru în lumea cinematografica: ***Cele şase grade ale lui Kevin Bacon***; o să definim algoritmul de construcţie a grafului ce va modela colecţia noastră. Actorul american Kevin Bacon este unul dintre cei mai apreciaţi actori de la Hollywood, având la activ zeci de filme celebre şi seriale. Graţie portofoliului său bogat şi carierei îndelungate, trei studenţi 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 lanţ ce include cel mult şase filme.

3 Algoritmul de creare a grafului

Se dă un fişier text care conţine informaţii despre o colecţie de filme. Fiecare actor reprezintă un nod din graful care modelează colecţia, iar doi actori sunt conectaţi printr-o muchie dacă au fost implicaţi în realizarea unui film comun. Pentru o mai bună înţelegere a algoritmului de creare a grafului, se va oferi un exemplu detaliat.

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

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

4 Detalierea reprezentării

4.1 Informaţii

Se porneşte de la următoarea colecţie:

Film1
- Actor1
- Actor2
- Actor3

Film2
- Actor1
- Actor3
- Actor4

Film3
- Actor2
- Actor4
- Actor5

Film4
- Actor6
- Actor7
- Actor8

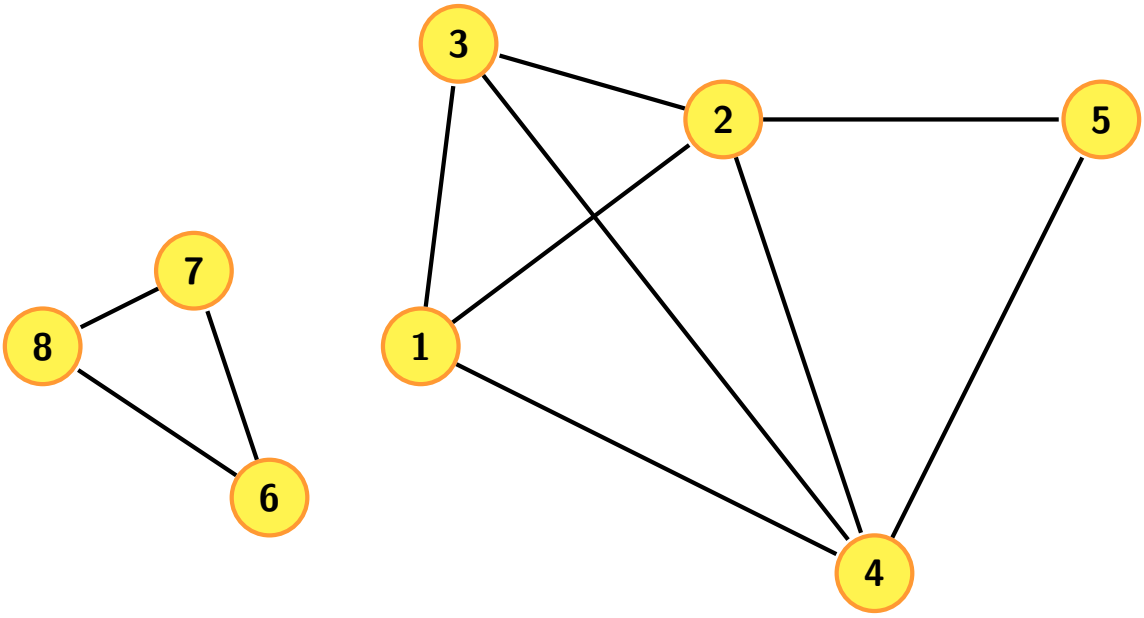
4.2 Explicaţii

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

După cum se poate deduce, în exemplul expus există două producţii. Prima producţie conţine primele 3 filme (*Film1*, *Film2* şi *Film3*), iar a doua producţie conţine doar ultimul film (*Film4*).

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

4.3 Reprezentare

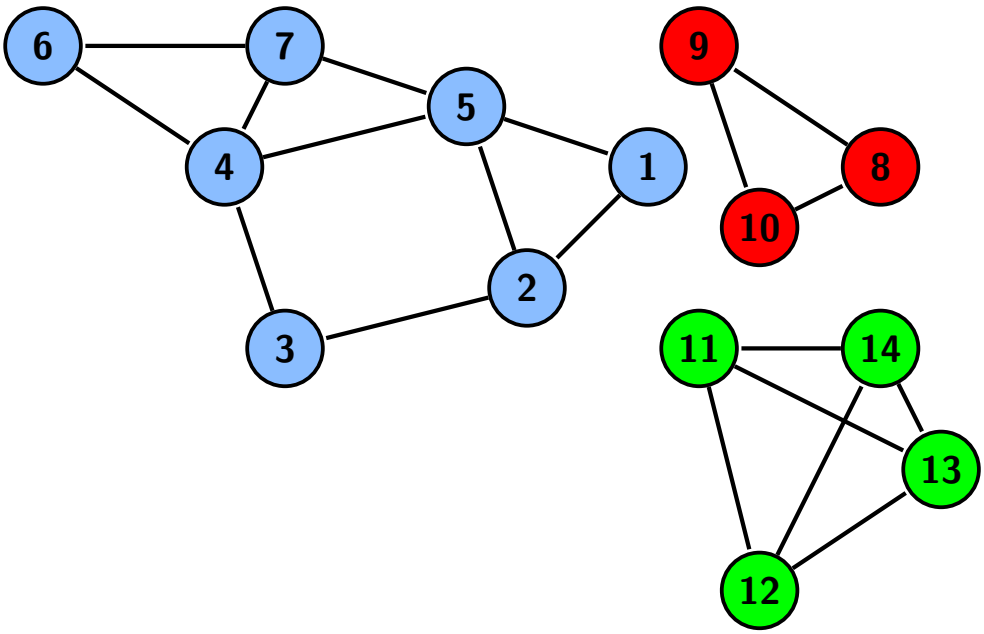


5 Cerinţe

5.1 Cerinţa 1

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

Pentru a înţelege mai clar ce trebuie realizat la această cerinţă, pornim de la analizarea următorului exemplu care conţine 3 producţii, actorii din fiecare producţie fiind coloraţi cu una dintre culorile *albastru*, *roşu*, *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.

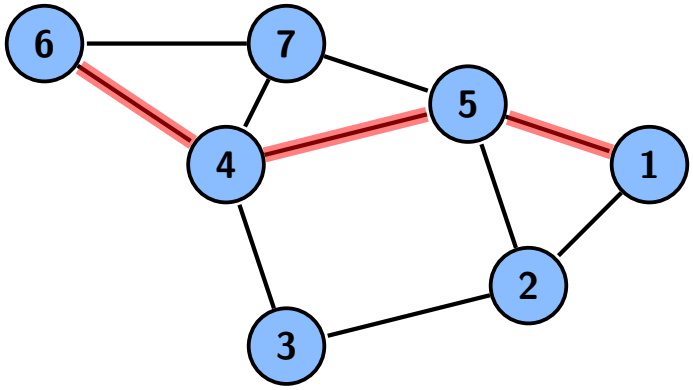
Important

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

5.2 Cerinţa 2

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

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



Important

Este **obligatoriu** ca rezolvarea aceste cerinţe să fie realizată folosind o structură de date de tip **graf** reprezentat uzitând liste de adiacenţă!

5.3 Cerinţa 3

Definiţii

1. Se numeşte **lanţ** într-un graf neorientat, o secvenţă de vârfuri $[x_1, x_2, \ldots, x_k]$, cu proprietatea că oricare două vârfuri consecutive din secvenţă sunt adiacente.
2. Se numește **ciclu** într-un graf un lanț $v_1, v_2, \ldots, v_k$ cu proprietatea că $v_1 = v_k$ și muchiile (v_1, v_2), $(v_2, v_3), \ldots, (v_{k-1}, v_k)$ sunt distincte două câte două.
3. Un graf se numeşte **conex** dacă pentru oricare două vârfuri x şi y diferite ale sale, există un lanţ 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 informaţie care ar putea fi interesantă pentru o persoană pasionată de filme şi grafuri este aceea legată de numele actorilor între care există o punte în graful construit pe baza raţionamentului prezentat anterior. De aceea, în cadrul acestei cerințe voi va trebui să determinați actorii între care există o puncte în graful distribuției. Pentru determinarea punților într-un graf neorientat se foloseşte o parcurgere în adâncime modificată, reţinându-se informaţii 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 notaţii:

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

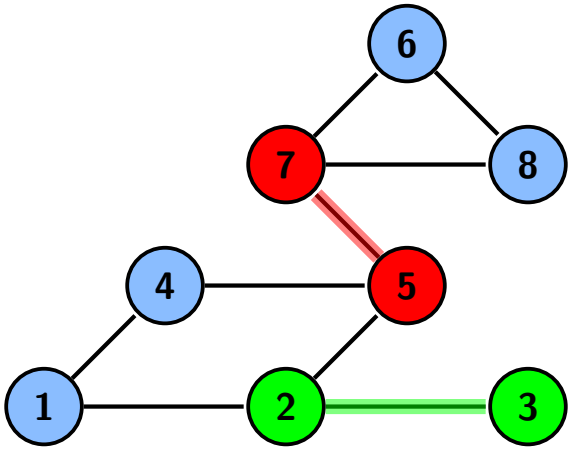
$$low[u] = \textbf{min}(\{idx[u]\} \cup \{idx[v] | (u, v) \text{ este o muchie înapoi }\} \cup \{low[vi] | vi \text{ copil al lui } u \text{ în arborele de adâncime }\})$$

Folosind aceste notaţii, condiţia pentru ca o muchie (v, u) să fie puncte 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 roşu există o punte și între cele colorate cu verde.



Actorii între care există o **punte** în graful producţiilor cinematografice vor fi afişaţi în ordine alfabetică.

Important

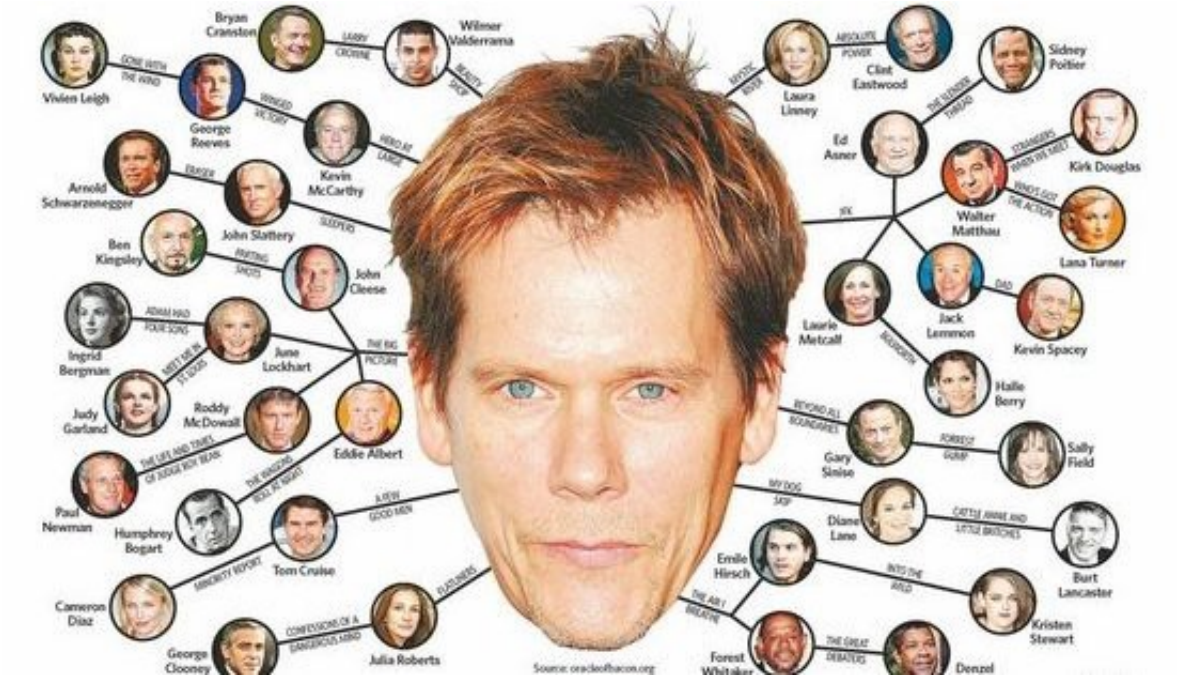
Este **obligatoriu** ca rezolvarea aceste cerinţe să fie realizată folosind o structură de date de tip **graf** reprezentat uzitând liste de adiacenţă!

Important

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

Algorithm 1 Punți

```
1:  procedure PUNȚI(G)
2:      (V, E) ← G
3:      timp ← 0
4:      for each v ∈ V do
5:          idx[v] ← −1
6:          low[v] ← ∞
7:          π[v] ← nill
8:      for each v ∈ V do
9:          if idx[v] == −1 then
10:             dfsB(G, v, timp, idx, low, π)
11: procedure DFSB(G, v, timp, idx, low, π)
12:     (V, E) ← G
13:     idx[v] ← timp
14:     low[v] ← timp
15:     timp ← timp + 1
16:     copii ← ∅
17:     for each (v, u) ∈ E do
18:         if u ≠ π[v] then
19:             if idx[u] == −1 then          ▷ înseamnă că nodul u este nedescoperit, deci alb
20:                 π[u] ← v
21:                 dfsB(G, u, timp, idx, low, π)
22:                 low[v] ← min(low[v], low[u])
23:                 if low[u] > low[v] then
24:                     (v, u) este punte
25:             else        ▷ înseamnă că nodul u este descoperit, deci gri, iar muchia v → u este muchie înapoi
26:                 low[v] ← min(low[v], idx[u])
```



6 Formatul fişierelor

Fişierul de intrare va conţine, pe prima linie, un număr natural **Nr**, reprezentând numărul de filme din colecţie. Pe următoarele linii se vor găsi informaţiile despre cele **Nr** filme. Pentru fiecare film se va specifica numele filmului, numărul de actori implicaţi în realizarea acelui film şi o listă a actorilor (câte unul pe rând).

Pentru **cerinţa 1**, acest fişier nu va conţine informaţii suplimentare, iar fişierul de ieşire va conţine pe prima linie un singur număr, reprezentând numărul de actori care fac parte din producţia maximală, iar pe următoarele linii se vor găsi numele acestor actori, afișați în ordine crescătoare și câte unul pe linie.

Pentru **cerinţa 2**, fişierul de intrare va conţine, la sfârşit două linii suplimentare. Pe penultima linie din fişier şi pe ultima se vor afla numele actorilor pentru care ne dorim să determinăm gradul de înrudire.

Pentru **cerinţa 3**, nu vom avea informaţii suplimentare în fişierul de intrare, iar în fişierul de ieşire se va afişa pe prima linie numărul de punți din graful construit, iar pe următoarele linii se vor afişa numele actorilor între care există o muchie a cărei eliminare duce la creşterea numărului de componente conexe în graful producţiei. Numele actorilor se vor afişa în ordine alfabetică.

7 Restricţii şi 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 conţine toate fişierele sursă alături de un **Makefile**, ce va fi folosit pentru compilare, şi un fişier **README**, în care se vor preciza detaliile implementării.

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

Programul vostru va primi, ca argumente în linia de comandă, numele fişierului de intrare şi a celui de ieşire, dar şi o opţiune, pentru fiecare cerinţă în parte, în felul următor:

./movies [-c1 | -c2 | -c3] [fisier_intrare] [fisier_iesire]

8 Punctaj

Cerinţa	Punctaj
Cerinţa 1	30 de puncte
Cerinţa 2	30 de puncte
Cerinţa 3	40 de puncte

Atentie!

Orice rezolvare care nu conţine structurile de date specificate nu este punctată.
Temele vor fi punctate doar pentru testele care sunt trecute pe vmchecker.
Nu lăsaţi warning-urile nerezolvate, deoarece veţi fi depunctaţi.
Dealocaţi toată memoria alocată pentru reţinerea informaţiilor, deoarece se vor depuncta pierderile de memorie.
Tema este individuală! Toate soluţiile trimise vor fi verificate, folosind o unealtă pentru detectarea plagiatului.