Tema 3

- la disciplina Structuri de date -

Ana-Maria Micu şi George Bukkosi
Grupa 311CA, Anul I, Facultatea de Automatică şi Calculatoare,
Universitatea "Politehnica" Bucureşti

23 mai 2017

Scurtă descriere Acest fişier reprezintă documentul README al celei de-a treia temă de la disciplina Structuri de date rezolvată de studenţii Micu Ana-Maria şi Bukkosi George. Aceasta presupune implementarea unei simulări pentru modul de funcţionare al site-ului IMDb.

Prin urmare, în următoarele pagini se vor regăsi explicaţii în legătură cu modul în care a fost abordată problema şi în legătura cu modul în care am ales să o rezolvăm.

De asemenea, documentul conţine şi un exemplu în care am luat unul dintre teste şi l-am detaliat în funcţie de paşii urmaţi în implementarea soluţiei.

Cuprins

1 Prezentare generala. Github

Aceasta a fost o temă ce necesita rezolvarea în echipe de două persoane. Pentru a dovedi faptul că amândoi am participat la conceperea şi rezolvarea temei, am ales să folosim un site de versionare, şi anume Github.

Pe acest site am creat un nou repository, T3 [2], în care fiecare încărca ceea ce lucra. De obicei, făceam fiecare câte două-trei funcţii, urmând ca apoi să le încărcăm pe site. Sistemul de versionare poate dovedi acest lucru.

2 Structuri folosite

Pentru a facilita modul de stocare al datelor, am ales sa implementam câteva structuri care conţin date despre diferite lucruri. Fiecare structură are propriile metode care au scopul de a modulariza codul. Aceste metode au drept comentarii în cod scopurile lor în cod şi ceea ce realizează, motiv pentru care nu mai este necesar să fie explicate în acest document.

2.1 Structura "movie"

Această structură conţine detaliile despre un anumit film de pe site. Pe lângă detaliile precizate deja in enunţul temei [3], am mai adăugat câteva variabile:

1. number_rating: memorează numărul total de rating-uri primite de un film de la utilizatori

2. total_rating: memorează suma tuturor rating-urilor primite de un film de la utilizatori

Scopul acestor noi variabile va fi vizibil ulterior, adică la calcularea rating-ului mediu al unui film.

Printre metodele acestei structuri se regasesc: constructorul, adăugarea unui rating, eliminarea unui rating, calcularea rating-ului mediu (sub formă de string, aproximat la două zecimale si sub formă de double, neaproximat), calcularea anului apariţiei filmului folosind operaţii cu timestamp-ul acestuia.

2.2 Structura "actor"

Această structură conţine detaliile despre un anumit actor de pe site. Pe lângă atributele prezentate deja în enunţul temei [3], am mai adăugat câteva:

1. struct movie *first_movie: este un pointer la structura primului film din cariera actorului respectiv

2. struct movie *last_movie: este un pointer la structura ultimului film din cariera actorului respectiv

Scopul acestor noi variabile va fi vizibil ulterior, adica la calcularea duratei carierei unui actor.

Printre metodele acestei structuri se regăsesc: constructorul, adaugarea unui film la cariera actorului, calcularea duratei carierei actorului.

2.3 Structura "user"

Această structură conţine detaliile despre un anumit utilizator de pe site. Pe lângă atributele prezentate deja în enunţul temei [3], am mai adăugat câteva:

1. unordered_map<string, int> ratings: este un Hashtable ce are ca scop memorarea tuturor perechilor de tipul (id_movie, rating) date de un utilizator

Scopul acestei noi variabile va fi vizibil ulterior, adica la adaugarea, actualizarea sau eliminarea unui rating.

Printre metodele acestei structuri se regăsesc: constructorul, adaugarea unui rating şi eliminarea unui rating pentru un anumit film.

2.4 Structura "director"

Această structură conţine detaliile despre un anumit regizor de pe site. Pe lângă atributele prezentate deja în enunţul temei [3], am mai adăugat câteva:

1. number_actors: reprezintă numărul total de actori care au jucat in filme conduse de un anumit regizor

Scopul acestei noi variabile va fi vizibil ulterior, adică la găsirea celui mai influent regizor.

Printre metodele acestei structuri se regăşeşte doar constructorul.

2.5 Structura "ratings"

Această structură conţine detaliile despre rating-urile înregistrate pentru un anumit film sau pentru o anumită perioadă. Variabilele conţinute de structură sunt:

1. number_ratings: memorează numărul total de rating-uri specifice
2. total_rating: memorează suma tuturor rating-urilor specifice

Scopul acestor variabile va fi vizibil ulterior, adică la găsirea mediei rating-urilor pentru o anumită perioadă sau pentru o anumită categorie.

Printre metodele acestei structuri se regăsesc: constructorul, adaugarea unui rating şi calcularea mediei rating-urilor.

2.6 Structura "actor_pair"

Această structură conţine detaliile despre o pereche de actori care au jucat împreună în cel puţin un film. Variabilele conţinute de structură sunt:

1. actor_id1: este id-ul primului actor ce formează perechea; este id-ul mai mic
2. actor_id2: este id-ul celui de-al doilea actor ce formează perechea; este id-ul mai mare
3. number_movies: reprezintă numărul de filme în care au jucat împreună cei doi actori cu id-urile respective

Scopul acestor variabile va fi vizibil ulterior, adică la găsirea perechilor de actori care au jucat împreuna în cele mai multe filme.

Printre metodele acestei structuri se regăsesc: constructorul, o funcţie ce returnează informaţiile din structură într-un mod convenabil.

3 Funcţii implementate

Această parte a documentului va conţine explicaţii cu privire la modul de rezolvare al cerinţelor, adică modul de implementare al funcţiilor al caror header ne-a fost dat.

Pentru a fi posibilă implementarea lor, a fost necesar să definim câteva noi variabile:

1. unordered_map<string, struct movie> movies: este un Hashtable ce ajută la memorarea perechilor de forma (id film, structură film); accesul la un anumit film este facilitat

2. unordered_map<string, struct user> users: este un Hashtable ce ajută la memorarea perechilor de forma (id utilizator, structură utilizator); accesul la un anumit utilizator este mai facil

3. unordered_map<string, struct actor> actors: este un Hashtable ce ajută la memorarea perechilor de forma (id actor, structură actor); accesul la un anumit actor este mai facil

4. unordered_map<string, struct director> directors: este un Hashtable ce ajută la memorarea perechilor de forma (id regizor, structură regizor); accesul la un anumit regizor este mai facil

5. unordered_map<string, vector<struct movie *>> categories: este un Hashtable ce ajută la memorarea perechilor de forma (categorie, vector de pointeri la filmele din categorie); accesul la filmele dintr-o anumită categorie este mai facil

6. Graph<string> colleagues: este un graf implementat cu ajutorul unei structuri proprii de graf neorientat cu cost; are drept noduri id-urile actorilor, iar ca muchii legăturile dintre ei; între doi actori există o muchie dacă au jucat în cel puţin un film; costul este reprezentat de numărul de filme în care au jucat cei doi actori; este util pentru mai multe dintre funcţiile ce se cer a fi implementate, precum: aflarea actorilor de ordin doi, top-ul partenerilor unui actor

7. struct director *most_influential_director: este un pointer la structura regizorului cel mai influent; este definit în clasa IMDb, deoarece are nevoie de actualizare constantă

8. struct actor *longest_career_actor: este un pointer la structura actorului cu cea mai lunga cariera; este definit în clasa IMDb, deoarece are nevoie de actualizare constantă

9. Heap<struct movie *> popular_movies: este un max Heap ce ajută la alcatuirea unui top al celor mai populare filme

10. bool top_popularity_has_changed: este o variabilă care este false dacă top-ul celor mai populare filme nu s-a schimbat, adică nu s-a mai adaugat, şters sau actualizat niciun rating sau film şi este true în caz contrar

11. Heap<struct movie *> recent_movies: este un max Heap ce ajută la alcatuirea unui top al celor mai recente filme

12. bool top_recent_movies_has_changed: este o variabilă care este true dacă top-ul celor mai recente filme s-a schimbat, adică s-a mai adăugat un nou film şi este false în caz contrar

3.1 Funcţia IMDb

Această funcţie reprezintă constructorul clasei şi are rolul de a iniţializa pointerii şi Heap-urile.

3.2 Funcţia add_movie

Această funcţie este cea mai complexă, deoarece realizează toate actualizările structurilor de date ce se petrec la adăugarea unui film. Acestea sunt următoarele:

1. crearea structurii noului film şi adăugarea ei în Hashtable-ul filmelor

2. actualizarea grafului de colegi prin adaugarea noilor relaţii dintre actori

3. adăugarea regizorului în Hashtable-ul regizorilor

4. adgarea noului film în Hashtable-urile categoriilor în care se încadrează

3.3 Funcţia add_user

Această funcţie este responsabilă cu adăugarea unui utilizator în Hashtable-ul utilizatorilor, prin apelarea metodei specifice.

3.4 Funcţia add_actor

Această funcţie este responsabilă cu adăugarea unui actor în Hashtable-ul actorilor, prin apelarea metodei specifice.

3.5 Funcţia add_director

Această funţie este apelată în momentul adăugarii unui film şi are scopul de a actualiza datele despre un regizor. Dacă acesta nu există în Hashtable-ul regizorului, este adaugat, iar dacă există, îi este actualizat numărul de actori pe care i-a condus. În ambele situaţii, se verifică dacă după modificarea datelor, regizorul nu a devenit cel mai influent, caz în care se actualizeaza pointer-ul la cel mai influent regizor.

3.6 Funcţia add_rating

Funcţia realizează actualizarea datelor la adăugarea unui nou rating, în felul următor:

1. se adaugă noul rating în cadrul Hashtable-ului de rating-uri date de utilizatorul respectiv
2. se adaugă noul rating la filmul respectiv
3. se semnalează modificarea top-ului filmelor populare, deoarece numărul de rating-uri al unui film a crescut

3.7 Funcţia update_rating

Funcţia realizează actualizarea unui rating dat de un utilizator unui film, urmând aceiaşi paşi ca la ştergerea rating-ului vechi şi adăugarea celui nou.

3.8 Funcţia remove_rating

Funcţia realizează actualizarea datelor la ştergerea unui rating, în felul următor:

1. se şterge rating-ul din cadrul Hashtable-ului de rating-uri date de utilizatorul respectiv
2. se şterge rating-ul la filmul respectiv
3. se semnalează modificarea top-ului filmelor populare, deoarece numărul de rating-uri al unui film a scăzut

3.9 Funcţia get_rating

Această funcţie are rolul de a apela metoda structurii movie care calculează media rating-urilor pe care le are un film.

3.10 Funcţia get_longest_career_actor

Având memorat un pointer la structura actorului cu cea mai lungă cariera pe care îl actualizăm constant la adăugarea unui film, este uşor ca aceasta funcţie sa reuturneze id-ul actorului a cărui structură se găseşte la această adresă.

3.11 Funcţia get_most_influential_director

Şi în cadrul acestei funcţii avem deja obţinut un pointer la structura regizorului cel mai influent. Prin urmare, se returnează dor id-ul directorului ce este memorat la această adresă.

3.12 Funcţia get_best_year_for_category

Funcţia returnează anul în care s-a înregistrat cel mai bun rating mediu pentru filmele ce fac parte din categoria respectivă. Implementarea acesteia este construită din mai mulţi paşi, după cum urmează:

1. găsirea vectorului de filme din categoria respectivă prin intermediul Hashtable-ului de categorii

2. parcurgerea vectorului de filme şi crearea unui Hashtable cu anii din categoria respectivă, Hashtable care conţine perechi de forma (an, structură ratings)

3. găsirea anului cu rating-ul cel mai bun prin comparaţii repetate

3.13 Funcţia get_2nd_degree_colleagues

Funcţia are rolul de a găsi colegii de gradul doi ai unui actor. Acest lucru se realizează prin intermediul grafului creat anterior, urmărind următorii paşi:

1. se caută colegii de gradul întâi, adică vecinii nodului din graf

2. se parcurg vecinii de gradul întâi ai vecinilor de gradul întâi, deoarece ei sunt posibili colegi de gradul doi

3. se verifică dacă nu cumva sunt vecini de gradul întâi, moment în care se sare peste vecinul respectiv; dacă nu sunt colegi de gradul întâi, se adaugă întru-n vector

4. se sortează vectorul astfel obţinut pentru a avea toţi colegii de gradul doi ordonaţi crescator după id

3.14 Funcţia get_top_k_most_recent_movies

Acestă funcţie preia Heap-ul creat pe parcurs ce conţine datele despre filmele cele mai recente. Fiind un max Heap, extragerea radacinei va determina construirea top-ului. Funcţia de comparare a elementelor din Heap compară doar timestamp-ul, deoarece acesta este unic şi este suficient doar un criteriu de comparaţie.

Este de remarcat faptul că dacă top-ul se schimbă, structura de Heap trebuie actualizată prin parcurgerea tuturor filmelor. Motivul pentru care am folosit acea variabilă care memorează daca top-ul s-a schimbat este că fisierul de input poate conţine multe comenzi de interogare fară a face modificări asupra top-ului.

3.15 Funcţia get_top_k_actor_pairs

Acestă funcţie creează un nou Heap în care se adaugă structuri actor_pair. Pentru a compara aceste elemente din Heap, am folosit o funcţie de comparare care compară mai întai numărul de filme, iar apoi id-urile actorilor.

De asemenea, folosim o matrice se adiacenţa, ce conţine 1 dacă perechea formată din actorul cu index-ul liniei şi actorul cu index-ul coloanei a fost introdusă deja în Heap sau 0 în caz contrar. Scopul ei este de a nu introduce în Heap de două ori aceeaşi pereche.

Odată construit Heap-ul, este uşor să se extragă primele k elemente ordonate descrescător.

3.16 Funcţia get_top_k_partners_for_actor

Pentru această funcţie ce generează top-ul colegilor de ordinul întâi se foloseşte tot graful cu legăturile dintre actori, deoarece este suficient să se preia vecinii nodului actorului respectiv.

Pentru a sorta aceşti parteneri, am folosit tot un max Heap, deoarece extragerea repetată a rădacinei determină formarea top-ului. Pentru acest Heap am folosit o altă funcţie de comparaţie care mai întâi compară numărul de filme, iar în caz de egalitate compară is-urile.

3.17 Funcţia get_top_k_most_popular_movies

Acestă funcţie preia Heap-ul creat pe parcurs ce conţine datele despre filmele cele mai populare. Fiind un max Heap, extragerea radacinei va determina construirea top-ului. Funcţia de comparare a elementelor din Heap compară atât numărul de rating-uri, cât şi id-ul.

Este de remarcat faptul că dacă top-ul se schimbă, structura de Heap trebuie actualizată prin parcurgerea tuturor filmelor. Motivul pentru care am folosit acea variabilă care memorează daca top-ul s-a schimbat este că fisierul de input poate conţine multe comenzi de interogare fară a face modificări asupra top-ului.

3.18 Funcţia get_avg_rating_in_range

Aceasta funcţie nu este implementată eficient, dar este implementată corect.

Pentru a calcula media rating-urilor medii dintr-un anumit interval, se realizează parcurgerea Hashtable-ului de filme şi, daca timestamp-ul acestuia se regăseşte în intervalul cerut, se adaugă rating-ul mediu la suma totală şi se incrementează numărul de filme din perioadă respectivă.

4 Exemplu

Pentru a arăta mai bine modul de folosire al grafului, am ales să construim un exemplu cu datele de intrare de la testul 0, identic cu cel din enunţul temei [3].

Înainte de primirea celei de-a 11-a interogări, cea pentru aflarea colegilor de gradul doi pentru un actor, graful colegilor este cel din Fig. **??**.

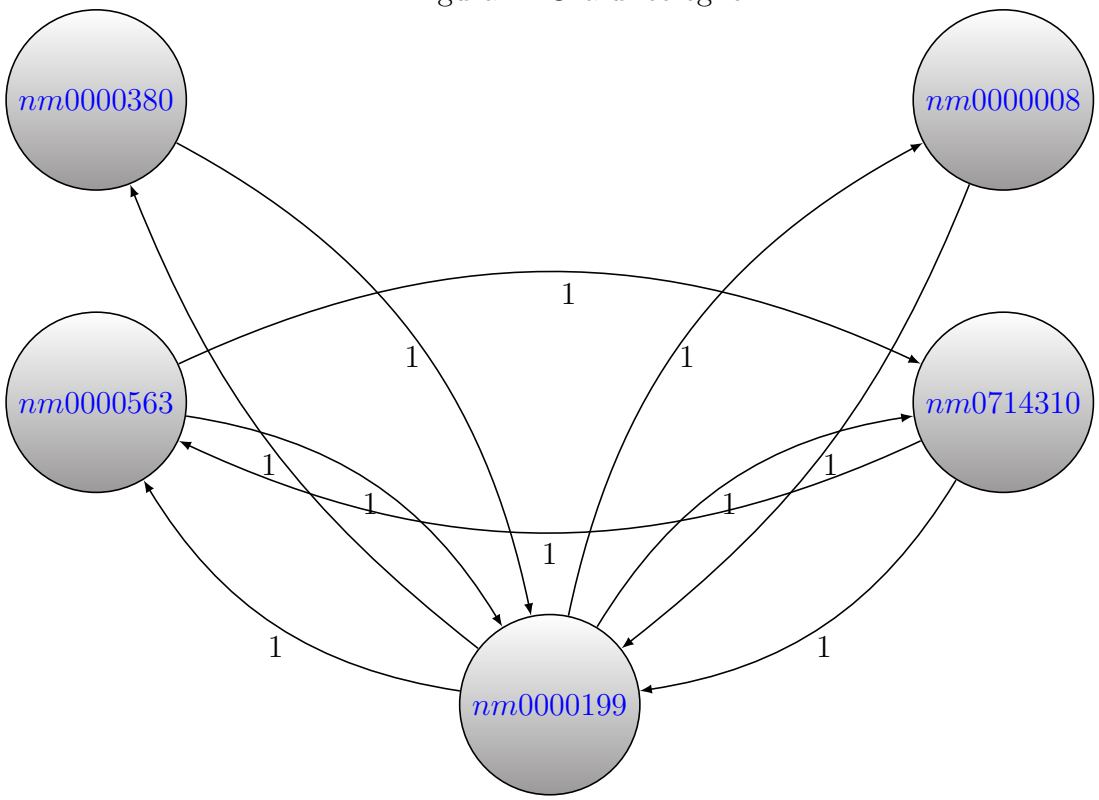
Având acest graf, este uşor să se găsească colegii de gradul doi ca fiind colegi de gradul întai ai colegilor de gradul întai, adică vecini ai vecinilor, fară sa fie vecin al nodului respectiv.

De asemenea, tot acest graf este folosit pentru a determina top-ul perechilor de actori şi topul colegilor.

5 Concluzie

Deşi nu este cea mai eficientă implementare pe care o poate avea această temă(considerăm că scopul acestei materii nu este eficienţa programelor), am încercat să folosim noţiunile noi învăţate pentru a le aprofunda(graf, hashtable, lista, Heap).

Figura 1: Graful colegilor.



Bibliografie

[1] *Laborator 9 - ABC si Heap*. Link.

[2] Micu Ana-Maria Bukkosi George. *Repository-ul temei*. Link.

[3] Armand Nicolicioiu Andrei Petre. *Tema 3 - IMDb*. Link.