Progetto Movida

Gianpaolo Roversi, Mattia Maranzana

Mappature implementate:

* AVL
* Hash Indirizzamento Aperto

Algoritmi di ordinamento implementati:

* BubbleSort
* QuickSort

Idee implementative:

* Dato che le strutture dati assegnateci non permettono riordinamenti, poiché devono esserne rispettate le proprietà, abbiamo deciso di utilizzare vari Array e ArrayList supplementari *(mArray, actors, directors, collab)* per permettere gli ordinamenti e facilitare le ricerche.
* La funzione *loadFromFile* con uno scanner recupera le informazioni dal file e inizializza sia la struttura dati principale (AVL o tabellaHash) sia le strutture ausiliarie.
* Alla creazione di una istanza di *MovidaCore* per default sono settati AVL come mappatura e QuickSort come algoritmo di ordinamento.
* Cambiare mappatura tramite la funzione *setMap* non modifica le strutture dati, quindi per avere un aggiornamento completo si deve chiamare anche *loadFromFile.*
* La funzione *deleteMovieByTitle* cancella il Movie in questione sia dalla struttura dati principale che da quelle ausiliarie.
* Abbiamo aggiunto un parametro alla classe *Person* chiamato *frequency* che indica il numero di ingaggi: il parametro è usato per alcune ricerche e aggiornato nella funzione *getAllPeople.*

Abbiamo deciso di usare due chiavi diverse per AVL e tabelle hash per poterle sfruttare in ricerche diverse. Gli AVL usano i voti dei film, mentre le tabelle hash l’anno di pubblicazione.

* La funzione *searchMoviesByTitle* esegue la ricerca sulle strutture dati ma non ne sfrutta le proprietà.
* La funzione *searchMoviesInYear* sfrutta l’implementazione della tabella hash che usa come chiave l’anno dei film, oppure nel caso in cui la mappatura sia settata con gli AVL, esegue una ricerca sulle strutture dati ausiliarie usando un algoritmo di ordinamento.
* La funzione *searchMostVotedMovies* sfrutta le proprietà degli AVL per la ricerca degli N film più votati, oppure nel caso in cui la mappatura sia settata con la tabella hash, esegue un ordinamento in base al voto dei film.
* Per la funzione *maximizeCollaborationsInTheTeamOf* abbiamo deciso di implementare la classe *UnionFind* per poter utilizzare l’algoritmo di Kruskal, e di creare una classe *collab\_comparator* usata nella coda con priorità, che permette di confrontare due *Collaborations* in base al loro punteggio.
* Abbiamo aggiunto un parametro alla classe *Collaboration* chiamato *mark* che nella funzione *maximizeCollaborationsInTheTeamOf* permette di tenere traccia delle *Collaborations* già aggiunte.

Nella cartella del progetto abbiamo anche aggiunto un file chiamato *MAIN* in cui abbiamo simulato una esecuzione del progetto, mostrando l’output di alcune funzioni cambiando sia la mappatura che l’ordinamento.