IUT Laval
Le Mans Université

Département Informatique

Sébastien George et Yann Walkowiak

SAE 1.02 Comparaison d'approches algorithmiques

Exploration de données cinématographiques

SUJET DU PROJET

Votre entreprise travaille dans le domaine des données ouvertes (*open data*) qui désigne des données numériques dont l'accès et l'usage sont libres. Dans ce cadre, votre entreprise propose des applications informatiques permettant d'effectuer des opérations telles que des recherches, des filtres ou des tris sur des données existantes et accessibles librement. Les données sont le plus souvent fournies dans des fichiers texte structurés.
Un nouveau client vous demande une application afin de pouvoir explorer des données cinématographiques. Il souhaite pouvoir effectuer des requêtes sur les films produits entre 1944 et 2020.

ASPECTS TECHNIQUES DU PROJET

Les données que vous devez utiliser se trouvent dans un fichier de type TSV (*Tab-Separated Values*). Il s'agit d'un format texte ouvert représentant des données sous forme de valeurs séparées par des tabulations (alors qu'un fichier CSV possède des données séparées par des virgules). Chaque ligne correspond à une donnée et les champs d'une même ligne sont séparés par une tabulation. La première ligne du fichier indique les types de chaque champ.
Le fichier complet se nomme `IMDbmoviesFULL.tsv` et contient 80 944 films.
À noter que ce fichier est sous licence *Creative Commons Universal (CC01)*.

> *Remarques : pour les besoins de vos tests, vous pourrez travailler sur des données partielles avec des fichiers contenant 100, 1 000, 10 000 ou 40 000 films (ex : `IMDbmoviesCUT1000.tsv`)*

FONCTIONNALITÉS DEMANDÉES

Le client souhaite une application permettant d'effectuer les opérations suivantes :

- Charger les données (choix du fichier).
- Afficher les données.
- Trier les données (par titre, par année, par genre, par pays, par note, …).
- Filtrer les données pour ne conserver que les films correspondant à une année, un réalisateur, un acteur, un genre, …
- Rechercher des données en affichant les films correspondant à un titre précis.
- Sauvegarder une requête dans un fichier pour la réutiliser plus tard (*optionnel*).

Le programme proposera un menu pour réaliser ces opérations (avec des sous-menus lorsque nécessaire, par exemple pour choisir un critère de tri). Les opérations doivent pouvoir s'enchainer en cascade (ex. appliquer un filtre puis un tri). Il sera possible de recharger une base à tout moment.

ASPECTS TECHNIQUES POUR LA RÉALISATION DU PROJET

Vous devez écrire un programme java qui s'exécutera en mode console. Les données seront lues à partir des fichiers et stockées en mémoire dans des structures de données (plusieurs types de structures seront étudiées afin de comparer les performances, voir ci-dessous).
Vous devez définir une classe java `Film` afin de stocker les informations sur un film, en particulier :

- `le titre du film,`
- `l'année de réalisation,`
- `le genre du film (il peut y en avoir plusieurs),`
- `la durée du film,`
- `le pays de production,`
- `la langue,`
- `le réalisateur,`
- `le scénariste,`
- `la liste des acteurs,`

- `la description,`
- `le nombre de votes de spectateurs,`
- `la moyenne des votes.`

Concernant la structure de données, les films seront mis dans une liste. Deux implémentations seront comparées en termes de performance : `ArrayList` et `LinkedList`. Toutes deux implémentent l'interface List de java et vous utiliserez ainsi les mêmes fonctions d'ajouts, de recherche, de parcours, etc.. Le passage d'une implémentation à l'autre pourra donc se faire facilement en début de programme.

COMPARAISON DES APPROCHES ET MESURES DE PERFORMANCES

L'objectif du projet est de comparer les performances de différents algorithmes selon l'implémentation choisie. Des temps d'exécution seront mesurés afin de les comparer aux complexités théoriques des algorithmes. Vous veillerez à faire vos relevés sur la même machine.

Un tableur sera construit, avec au minimum ces informations :

	Complexité	évolution théorique					temps mesuré ArrayList					temps mesuré LinkedList				
	n	100	1000	10000	40000	80944	100	1000	10000	40000	80944	100	1000	10000	40000	80944
Filtre manuel																
Filtre Java																
Tri Sélection																
Tri Fusion																
Tri Java																
Recherche Linéaire																
Recherche Dichotomique																
Suppression Un à Un																

Des courbes seront tracées pour représenter visuellement les données. Vous choisirez un type de repère adapté en fonction du type de croissance de la complexité attendue.
Le détail des fonctionnalités et algorithmes attendus est donné en annexe.

ANALYSE DE L'OCCUPATION MÉMOIRE ET DE LA CHARGE CPU

Java fournit des outils pour analyser l'exécution des programmes. En particulier, Java Visual VM est un outil graphique qui permet de faire du monitoring sur des applications Java. On peut ainsi analyser la consommation CPU, l'utilisation de mémoire (*Heap*, ...), les instances de classes (*Heap Dump*), etc. Voici le chemin pour exécuter Java Visual VM :

C:\programme\java\jdk1.8.0_181\bin\jvisualvm.exe

OPTIONNEL : SAUVEGARDE DES RÉSULTATS DES REQUÊTES SUR DISQUES

À tout moment l'utilisateur pourra décider de sauvegarder une requête dans un fichier. Deux types de sauvegarde seront possibles :

- dans un fichier texte tabulaire (format TSV),
- dans un fichier binaire.

Les temps de traitement et la taille du fichier de sauvegarde seront analysés. Les avantages et inconvénients de chaque solution seront discutés.

LIVRABLES

Vous produirez un rapport décrivant précisément tous les constats effectués pendant ce projet. Vous discuterez notamment des temps de traitement (complexité temporelle) selon les types d'implémentation de la liste et selon les algorithmes. L'occupation en mémoire (complexité spatiale) sera aussi analysée.
Le rapport sera illustré avec des tableaux et graphiques provenant du tableur. L'intégralité du tableur sera également fournie en annexe, ainsi que le code source de l'application.

Date limite de remise : mercredi 20 décembre à 12h00 (sur UMTICE)

IUT Laval Le Mans Université
Département Informatique
Sébastien George et Yann Walkowiak

Détail des fonctionnalités et de leur implémentation

Chargement des données

L'utilisateur pourra choisir le fichier de données à charger :

- Base complète (80 944 films) : `IMDbmoviesFULL.tsv`
- Base partielle (40 000 films) : `IMDbmoviesCUT40000.tsv`
- Base partielle (10 000 films) : `IMDbmoviesCUT10000.tsv`
- Base partielle (1 000 films) : `IMDbmoviesCUT1000.tsv`
- Base partielle (100 films) : `IMDbmoviesCUT100.tsv`

Pour la lecture du fichier, le plus simple est de lire chaque ligne du fichier avec :

```
BufferedReader tsvReader = new BufferedReader(new FileReader(nomFichier));
String row=new String();
row=tsvReader.readLine();
```

Les champs d'une ligne peuvent ensuite être séparés par (le séparateur étant la tabulation) :

```
String[] data = row.split("\t");
```

Affichage des données

Chaque Film sera affiché sur une ligne dans un format que vous choisirez. La liste sera numérotée.

Dans Eclipse, il sera utile de paramétrer la console pour pouvoir afficher plus d'informations :
dans le menu Window > Preferences, allez dans Run/Debug > Console puis enlever la Limit console output.

Pour les grands volumes de données, vous pouvez choisir de n'afficher qu'un extrait (par exemple un film sur 100 ou 1000).

Pour l'affichage des données, vous pourrez utiliser le formatage des chaines de caractères en java :
Exemple : s.format("%-60s", titre); *// affiche le titre en ajoutant des espaces pour prendre 60 caractères*

%s	Permet d'injecter une chaîne de caractères.
%10s	Permet d'injecter une chaîne de caractères en utilisant 10 caractères dans l'affichage. La chaîne de caractères sera alignée par la droite. Si la taille de la chaîne est plus petite que la taille spécifiée, des blancs seront ajoutés à gauche. Si la taille de la chaîne est plus grande que la taille spécifiée, la taille du format sera ignorée.
%-10s	Permet d'injecter une chaîne de caractères en utilisant 10 caractères dans l'affichage. La chaîne de caractères sera alignée par la gauche. Si la taille de la chaîne est plus petite que la taille spécifiée, des blancs seront ajoutés par la droite. Si la taille de la chaîne est plus grande que la taille spécifiée, la taille du format sera ignorée.

Filtre (selon un critère)

- Filtre linéaire :

Vous écrirez un algorithme qui parcourt les éléments un par un et supprime de la liste chaque film qui ne correspond pas au critère souhaité (année, genre, acteur, …). Au final, la liste ne contiendra plus que les films correspondant au critère demandé.

- Filtre java :

Pour les collections java, il est possible d'appeler une fonction particulière pour retirer un élément ne correspondant pas à un critère. La fonction se nomme `removeIf` et prend un argument particulier : un prédicat (possible depuis java 8 avec les expressions lambda). Voici un exemple :

```
listeFilms.removeIf(f -> !f.getGenre().contains("Comedy"));
// supprime tous les films qui ne contiennent pas « Comedy » dans le genre
```

IUT Laval Le Mans Université
Département Informatique
Sébastien George et Yann Walkowiak

Tri (selon un critère)

- *Tri Sélection :*
  L'algorithme du tri sélection sera implémenté pour ordonner les éléments de la liste ***selon un seul attribut particulier d'un film (ex. année)***. Vous veillerez à privilégier au maximum l'utilisation d'itérateurs (`ListIterator`) plutôt que des accès directs (`listeFilms.get(i)`).
- *Tri Fusion :*
  Cet algorithme sera codé de la façon vue en TP. L'utilisation d'un tableau temporaire sera donc nécessaire pour ce tri (ce type de tri ne s'effectuant pas en place). ***Là encore le tri sera effectué selon un seul attribut particulier d'un film (ex. année).***
- *Tri Java :*
  L'algorithme de tri proposé par Java pour les collections sera utilisé (`Collections.sort(liste, fonction_comparaison)`). L'algorithme utilisé est le TimSort, vous vous documenterez sur ce type de tri fusion hybride. ***Pour le tri java, l'utilisateur pourra choisir différents critères de tri (année, note, titre, pays, …).***

Recherche (un titre de film)

- *Recherche linéaire :*
  L'algorithme parcourt les éléments un par un jusqu'à trouver un titre correspondant (même partiellement). Si on arrive à la fin de la liste, l'élément cherché n'est pas présent.
- *Recherche dichotomique :*
  L'algorithme s'exécute uniquement sur une liste triée (par titre de film ici). La recherche peut alors être optimisée en regardant l'élément situé au milieu de la liste afin de réduire de moitié l'espace de recherche. On recommence ainsi la recherche jusqu'à trouver ou non.

Suppression un à un

- L'algorithme supprime le premier élément de la liste et boucle jusqu'à tant qu'il n'y ait plus de film.

Sauvegarde dans un fichier *(optionnel)*

- *fichier en mode texte :*
  L'objectif est de sauvegarder les données de la liste actuelle dans un fichier au format texte tabulaire (TSV). La première ligne d'en-tête contiendra la liste des types de champs enregistrés.
- *fichier en mode binaire :*
  La sauvegarde de la liste s'effectue en mode binaire par sérialisation de l'objet java. Pour charger un fichier binaire, on veillera à vérifier que le type de la sauvegarde correspond à l'implémentation choisie (`ArrayList` ou `LinkedList`)