Ministère de l'Enseignement Supérieur
UNIVERSITÉ D'ORLÉANS

Institut Universitaire de Technologie IUT'O Orléans
DEPARTEMENT INFORMATIQUE

RAPPORT SAE

SAE n° 2.02

- A la conquête d'Hollywood

Élaboration d'une application pour modéliser des graphes.

BUT 1 Informatique

Elaboré par :

-MOINS Bastien

- COSME VINOU Ilona

2024/2025

Dans le cadre de cette SAE algo, nous avions pour mission d'étudier les relations entre acteurs de cinéma en utilisant les graphes. A la fin, nous obtiendrons des classes et implémentations qui permettent de visualiser comment les acteurs sont éloignés les uns des autres.

Pour commencer, nous nous sommes familiarisés avec la bibliothèque `JGraphT` que nous avions précédemment utilisée en TPs. Il a aussi fallu au début générer l'archétype du projet sous Maven… ce qui fut compliqué. Après nous être aidés des commandes utiles fournies dans les TPs, tout s'est bien déroulé. C'était très utile car on a eu le fichier `AppTest` généré. Cette première étape d'organisation de notre travail a permis de poser les bases. Dans la même lancée, on a créé notre dépôt GitHub.

La partie la plus longue fut l'échauffement : en effet, les données n'étaient pas en `JSON` classique... Il a fallu réfléchir et au final nous avons opté pour une classe statique `Film` qui reprend le nom des variables du fichier de données. Puis on parcourt les lignes grâce au `BufferedReader()`.

Nous avons mis en place des tests. Et on a également généré un fichier `DOT` puis `PDF` pour vérifier visuellement si tout allait bien à la conversion, même si cela n'était pas demandé.

-> Comment exprimez-vous cette notion (ensemble des collaborateurs en commun) en termes de théorie des graphes ?

Nous pouvons exprimer cette notion comme l'intersection des ensembles de voisins des deux sommets représentant les acteurs.

-> Pouvez-vous donner une borne inférieure sur le temps nécessaire à l'exécution de votre fonction ?

Si la représentation des voisins est un `HashSet`, l'intersection est en `O((d(u),d(v))` où `d(x)` est le degré.

-> Reconnaissez-vous l'algorithme classique en théorie des graphes qui est au cœur de ce programme ?

Oui, c'est la recherche en largeur pour explorer couche par couche comme vu en TP avec les couleurs.

-> Grâce à la fonction précédente, comment pouvez-vous déterminer si un acteur se trouve à distance k d'un autre acteur ?

Il faut faire un parcours par largeur depuis le premier acteur et vérifier à chaque niveau si l'autre acteur est présent dans le niveau atteint.

-> Est-ce que réutiliser la fonction précédente vous semble intéressant ?

Oui car la fonction de recherche à distance k renvoie l'ensemble des sommets à chaque niveau.

-> Donnez la complexité d'un tel algorithme.

La complexité asymptotique d'un tel algorithme est de O(V(acteur))

-> Quelle notion de théorie des graphes permet de modéliser cela ?

La notion sur laquelle nous nous penchons est l'excentricité qui cherche la distance maximale entre cet acteur et tout autre acteur dans le graphe.

-> Est-ce que ce nombre est bien inférieur ou égal à 6 pour le jeu de données fourni ?

```
// vérifie la théorie des 6 degrés
assertTrue(distanceMax <= 6);
```

Pour le jeu de données fourni il est difficile de vérifier les 6 degrés, mais la méthode fonctionne pour un graphe quelconque créé (en l'occurrence celui du test).

Au niveau des démonstrations de compétences, on pense avoir montré notre maîtrise de plusieurs compétences telles que :

-la gestion de projet collaboratif sous GitHub et en liveshare,

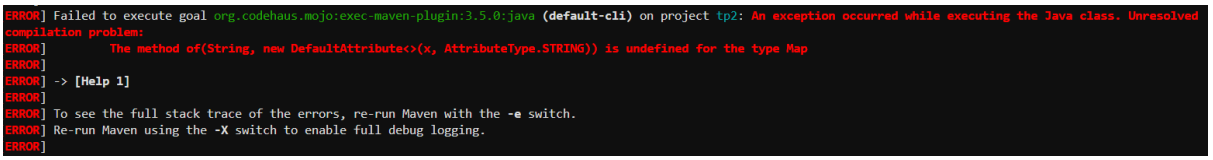
-la manipulation de fichiers avec `BufferedReader` et `Scanner`,

-analyser un problème avec méthode comme les intersections d'ensembles ou le parcours `BFS`,

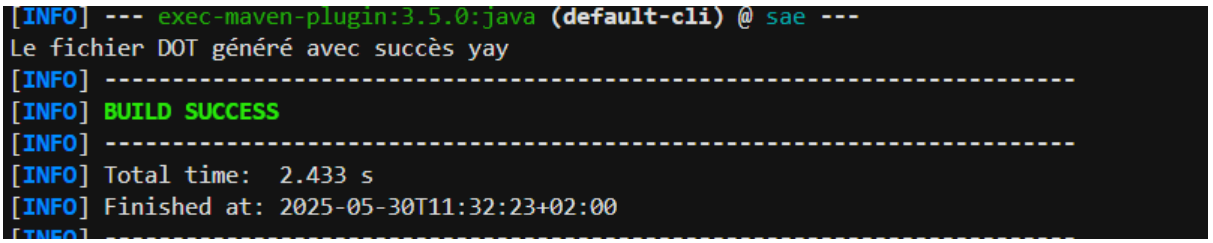
- l'ajout de notions par notre initiative comme les exceptions pour gérer les erreurs

- et enfin notre capacité à résoudre des problèmes totalement inattendus comme par exemple :

Pour resoudre l'erreur au niveau de la conversion `DOT`, il fallait rajouter une propriété maven dans le `pom.xml` pour mettre à jour le compilateur. Et il faut aussi penser à `clean compile`.



Ce n'est qu'un problème parmi tant d'autres que nous avons réussi à fixer. Voici ce que ca donne apres :



Sur les points à améliorer : notre code pourrait être plus rapide, et on aurait pu faire un menu dans le terminal avec plus de temps. On a aussi essayé la partie bonus mais elle n'est pas achevée. De plus, le fichier de données avec des milliers de lignes prendrait un temps conséquent à se convertir ce qui est dommage. La complexité de la méthode `jsonVersGraphe()` est $O(N^3)$, `getVoisins()` est en $O(N)$, `collaborateurs_proches` en $O(N^k)$. Le `main` est $O(N^3)$ et pour `acteursCentraux()` de la partie bonus, c'est pour l'instant en $O(N^2)$. La fonction `centraliteDUnActeur()`, et la fonction `centreDuGraphe()` sont de l'ordre de $O(N^2)$ puisqu'il parcourt une seconde fois les sommets.

Pour conclure, ce projet nous a permis de mettre en pratique la théorie des graphes, de collaborer comme il faut, d'utiliser une bibliothèque, et d'améliorer notre organisation de code. Les quelques points faibles rencontrés ont été des occasions pour progresser en autonomie ou grâce à notre entraide. En global, on est satisfaits d'avoir produit une bonne application et des classes solides avec des exceptions et des jolis tests.