Lucas THOMAS
Nicolas Nauche

Rapport collectif - SAE Graphe

Les différentes fonctions demandées dans le sujet on été réalisées (mise a part la fonction bonus) y compris la fonction “collaborateurs_proches” qui bien que donnée dans le sujet à été refaite.

La répartition des tâches à été la suivante (bien qu’au final chacun a contribué aux fonction en apportant des idées ou des optimisation):

- Lucas:
  - json_vers_nx
  - collaborateurs_proches
  - est_proche
  - distance
  - centralite
  - tests des cinques fonction ci dessus
- Nicolas:
  - collaborateurs_communs
  - distance_naive
  - centre_hollywood
  - eloignement_max
  - tests des quatres fonction ci dessus

A chaque fonction écrite des tests unitaire ont été mis en place avec l’outil “pytest” en créant des jeux de données plus petits (avec 2, et 5 films) et donc plus facilement testable.
et une fonction nommée “chrono” à également été implémentée afin de tester les performances en temps de chacune des fonctions.
Une attention particulière à été donnée au fait que le code respecte un maximum PEP8 et soit le plus lisible possible.

Les couts des différentes fonctions sont:

- json_vers_nx O(n)
- collaborateurs_communs O(n)
- collaborateurs_proches O(n)
- est_proche O(n)
- distance_naive $O(n^2)$
- distance O(n)
- centralite O(n)

- centre_hollywood $O(n^2)$
- eloignement_max $O(n^3)$

Les deux dernières fonctions ne sont clairement pas optimales et prennent beaucoup de temps à s'exécuter pour le fichier à 150k lignes. Mais nous n'avons pas su trouver des algorithmes plus optimisés.

Réponses aux questions:
6.2/

L'ensemble des collaborateurs en commun de deux acteurs est l'intersection de leurs ensembles de voisins dans le graphe représentant les collaborations.

6.3/

1) La fonction fournie qui prend un acteur et un entier k et renvoie la liste des acteurs à distance au plus k de cet acteur repose sur l'algorithme classique de **parcours en largeur (BFS)** en théorie des graphes.

2) Pour déterminer si un acteur se trouve à distance exacte k d'un autre acteur, on peut vérifier si cet acteur est présent dans la liste renvoyée par la fonction lorsqu'on l'appelle avec k, mais cela n'est pas optimal car cela nécessiterait des appels répétitifs à la fonction pour chaque distance k

3) Il suffit de faire un parcours en largeur, et de stocker notre "niveau" actuel et de retourner ce dernier quand l'acteur est trouvé.

6.4 /

Cette notion est l'excentricité.