

SAé 2.02 Exploitation d'un système algorithmique

Table of Contents

Travail réalisé.....	2
Acquis mobilisés et obtenus.....	2
Démonstration de compétences.....	2

Travail réalisé

Au cours de cette SAé, plusieurs travaux ont été réalisés. En effet, nous avons produit personnellement différentes fonctions notamment `collaborateurs_communs`, `est_proche`, `centralité`, ou encore `éloignement_max`. Pareillement, nous avons réalisé l'application de manière à visualiser les graphes et de pouvoir proposer une interface plus intuitive dans la manipulation des données. Enfin, nous avons réalisé les mesures de temps.

Toutefois diverses difficultés ont pu apparaître lors du développement de celles-ci. Effectivement, l'inquiétude de la complexité est apparue. Nous avons dû être particulièrement attentif à produire des fonctions dont la complexité restait raisonnable de manière à obtenir un résultat dans des délais corrects.

Pareillement, des difficultés d'organisation sont apparues. En effet, la première fonction à développer pour l'échauffement a été chronophage. Nous avons donc mis en place diverses solutions comme une répartition plus efficace du travail ou encore modéliser des problèmes sous différents points de vue.

Acquis mobilisés et obtenus

Tout au long de notre travail, nous avons pu mobiliser des connaissances acquises particulièrement en Python, en théorie des graphes mais aussi complexité algorithmique. En effet, de manière à pouvoir calculer les différentes complexités asymptotiques du sujet, il a été nécessaire de s'appuyer sur des apprentissages réalisés en méthodes numériques. Également, l'ensemble des fonctions et l'application se sont produits avec le langage de programmation Python. Ainsi, nous mobilisé nos connaissances mais aussi appris. Effectivement, des nouvelles notions en Python ont été utilisées. Par exemple, l'utilisation du « `match-case` » ou encore de nombreuses fonctions Networkx comme « `nx.shortest_path` » ou « `nx.degree_centrality` » afin de vérifier nos résultats.

De plus, nous avons appris à formaliser mathématiquement les demandes afin de produire des résultats et les interpréter.

Démonstration de compétences

Tout d'abord, nous avons appris à analyser un problème en utilisant une méthode. De façon à produire une application pour visualiser les résultats, nous avons choisi de découper le problème en sous-problèmes. Ainsi, il a été plus simple de produire le programme principal.

```
# fonctions élémentaires utiles au fonctionnement du programme principal
> def chemin():--
> def demandeDistance():--
> def chgmtActeurU(acteurU):--
> def choix_programme(json_vers_nx, acteurU):--
|
# programme principal permettant de lancer l'application
> def programme_principal():--
```

Par exemple, pour obtenir un graphe, nous devons demander à l'utilisateur le nom du fichier source. Pour cela, une fonction a été dédiée. Ces fonctions concernent notamment les interactions avec l'utilisateur

Pareillement, nous avons analysé les problèmes en utilisant différentes structures de données. Par exemple, la fonction 'distance' utilise un dictionnaire prenant en clé un nœud et en valeur la distance avec le point de départ. Pourtant, celle-ci utilisait, en premier lieu, un ensemble d'acteur. Toutefois, la complexité fut trop importante, il a donc été nécessaire de changer la structure de données pour répondre au mieux au problème.

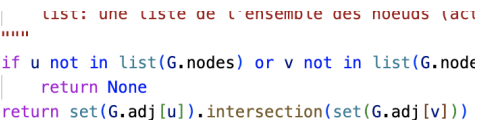
De ce fait, nous avons acquis la compétence, en premier temps, en analysant un problème à l'aide d'une méthode.

De plus, nous avons comparé des algorithmes pour des problèmes classiques. En effet, certains algorithmes, bien que corrects, ont pu être optimisés en réduisant la complexité. Afin de comparer au mieux, nous avons mis en place un tableau comportant les structures de données possibles en fonction des distances avec leur complexité.

	dictionnaire	set
distance	$O(n^2)$	$O(n^3)$
centre_hollywood	$O(n^3)$	$O(n^4)$
eloignement_max	$O(n^4)$	$O(n^4)$

Dans de nombreux cas, le dictionnaire nous permettait de garder en mémoire les distances avec leur noeud et de ne pas les recalculer à chaque tour. L'ensemble semble moins optimal que le dictionnaire.

De la même manière, la comparaison entre des algorithmes pour trouver des plus courts chemins ce sont fait. Il a été nécessaire d'utiliser des parcours plus simples pouvant réduire le temps d'exécution. Par exemple, nous avons comparé les deux méthodes BFS et DFS. Ainsi, nous avons assimilé la compétence 2 en comparant des algorithmes pour des problèmes classiques.

Enfin, nous avons su formaliser et mettre en œuvre des outils mathématiques pour l'informatique. Pour produire les différents résultats, nous avons utilisé les notions de complexité des algorithmes. Effectivement, en fonction de la taille des structures de données, il est important de calculer le temps d'exécution pour déterminer quelle méthode choisir. De même, nous avons pu utiliser la théorie des ensembles pour développer plusieurs fonctions comme 'collaborateurs_communs()'.


Sur cette capture, nous pouvons retrouver une notion de théorie des ensembles. On retourne un ensemble des voisins de $A \cap$ voisins de B .

De cette manière, nous avons acquis la compétence 2 en formalisant et mettant en œuvre des outils mathématiques pour l'informatique.

Ainsi, ces trois apprentissages critiques nous ont permis d'assimiler et d'intégrer la compétence 2, appréhender et construire des algorithmes simples.