Projet de théorie des graphes 2026-2027

Préambule :

Avec le développement d'IA génératives toujours plus performantes, il n'est plus envisageable de proposer des projets centrés sur la production de code. L'accent est dès lors placé sur la compréhension des concepts et le développement de vos capacités de communication et d'analyse.

Extrait de l'engagement pédagogique : "… l'évaluation portera essentiellement sur la compréhension des concepts de théorie des graphes utilisés, des principes algorithmiques mis en oeuvre et sur l'implication de l'étudiant.e dans le projet, plutôt que sur la seule production de code."

Description du projet :

Le degré d'un sommet ne suffit pas à caractériser "son importance" au sein d'un graphe. En effet, un sommet peut avoir de nombreux voisins tout en appartenant à une partie du graphe ayant peu de "cohésion". Ce projet étudie la décomposition en *k-cœurs* (*k-core)*. Cette notion permet d'identifier des sous-ensembles de sommets où chacun possède au moins *k* voisins dans l'ensemble. Elle révèle ainsi une organisation du graphe en niveaux emboîtés.

Vous développerez un code informatique (Python ou C) permettant de calculer cette décomposition et de l'utiliser pour analyser les graphes. Votre code peut aussi inclure un outil d'analyse de graphes fournissant des éléments pertinents (de votre choix). Le but est d'obtenir des informations sur la structure d'un graphe.

Le projet devra répondre à trois questions :

- Qu'apporte la décomposition en *k*-cœurs par rapport à la seule étude des degrés des sommets ?
- Quelles différences structurelles permet-elle de mettre en évidence entre plusieurs graphes ?
- Comment une observation incomplète du graphe (données partielles) affecte-t-elle les conclusions ?

À l'issue du projet, vous devrez être capable de

- définir et exploiter les notions rencontrées : degré, sous-graphe induit, degré interne, élagage récursif, *k*-coeur, *shell index*, *k-shell*, cluster, dégénérescence, composante connexe, distribution et corrélation des degrés, coefficient de clustering, autosimilarité ;
- justifier votre algorithme de décomposition et son implémentation ;
- analyser les résultats fournis par votre code.

L'utilisateur fourni un graphe non orienté au format GraphML, votre programme devra conserver les identifiants des sommets, détecter les boucles et les arêtes répétées, et documenter le traitement appliqué.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns">
  <graph id="G" edgedefault="undirected">
    <node id="A"/>
    <node id="B"/>
    <node id="C"/>
    <edge id="e1" source="A" target="B"/>
    <edge id="e2" source="A" target="B"/>
    <edge id="e3" source="B" target="C"/>
  </graph>
</graphml>
```

Un article présentant les notions essentielles se trouve ici : https://arxiv.org/abs/cs/0511007

Ne vous limitez pas à une seule référence, investiguez d'autres sources. Vous implémenterez l'élagage et la décomposition complète. Les bibliothèques de graphes comme NetworkX sont autorisées pour gérer les entrées-sorties, la visualisation et la vérification des résultats. Les fonctions de NetworkX calculant directement les cœurs ne devront pas remplacer votre implémentation.

Quelques questions qui peuvent guider votre réflexion :

- Pourquoi supprimer une seule fois les sommets de degré inférieur à *k* ne suffit-il pas ?
- Quelle différence y a-t-il entre un cœur et une coquille (shell) ?
- Un cœur peut-il comporter plusieurs composantes connexes ?

Pour aller plus loin, l'article suivant discute l'utilisation des coeurs dans l'analyse de la propagation d'une épidémie (la notion semble mieux adaptée que le degré pour déterminer qui sont les principaux agents contaminants) : https://arxiv.org/abs/1001.5285

Consignes :

Le code sera accompagné d'un ***rapport*** (max. 6 pages de texte) au format pdf. Le rapport doit décrire la stratégie utilisée, les choix opérés, les grandes étapes qui ont guidé votre réflexion. Il pourra aussi présenter les difficultés/challenges rencontrés en cours d'élaboration ou reprendre certains résultats expérimentaux (benchmarking sur des exemples types ou générés aléatoirement). On peut espérer que votre code n'est pas le résultat produit immédiatement par un prompt unique mais qu'il s'est construit progressivement, dans un travail régulier. Votre rapport doit faire état de ces échanges et des décisions prises (exemple : nous avons exploré *telle* piste, mais il s'est avéré que *telle autre* solution était préférable car …).

Le rapport doit aussi servir de manuel d'installation et de mode d'emploi succint. Citez vos sources (construire une petite bibliographie/sitographie) et détaillez votre contribution par rapport à ces références. Le rapport doit permettre d'identifier quand et comment une source a été utilisée. N'hésitez pas à consulter et comparer plusieurs sources pour sélectionner les plus pertinentes. Il est dommage de voir que des étudiants se contentent souvent d'une unique référence Wikipedia ou des sources proposées. Faites preuve d'esprit critique à toutes les étapes du projet.

Groupe de maximum 2 étudiants (le travail en solo est autorisé). Pas de groupe de 3 étudiants. Si un accord entre les étudiants n'est pas trouvé, le titulaire procédera à un tirage au sort des groupes.

Le plagiat est interdit : il est interdit d'échanger des solutions complètes ou partielles. Néanmoins, vous êtes autorisés à discuter entre groupes.

Timing :

- Lundi 5 octobre 2026 : choix individuel des modalités d'examen, répartition en groupes et choix du sujet. Les délégués créeront un fichier partagé reprenant sur une ligne les binômes.
- Dimanche 6 décembre 2026 (minuit) : dépôt du code et du rapport sous forme d'une archive envoyée par mail au titulaire du cours.
- Lundi 7 décembre 2026 : interrogation écrite, individuelle et obligatoire ; pendant le créneau du cours théorique, il faut présenter l'interrogation pour pouvoir défendre le projet. Cette interrogation a pour but de vérifier que vous avez compris les concepts utilisés dans le projet. Il permettra également de préciser l'implication de chacun dans la solution envoyée. Les résultats à cette interrogation interviennent dans la note globale du projet. Une insuffisance grave à l'interrogation entraîne l'échec au projet.
- Lundi 14 décembre 2026 - ordre de passage à déterminer : présentation orale.
- Janvier 2027 : examen écrit (commun pour tous).

Défense orale :

Pour la ***présentation orale***, il est conseillé que les étudiants utilisent un ordinateur pour faire tourner leur programme, pour présenter leur code, pour s'aider d'un support type "power point". Un projecteur vidéo est à disposition. Le but de cette présentation est de synthétiser et d'expliquer les concepts utilisés en maximum 10 minutes.

- Présenter, de façon pédagogique, la théorie nécessaire et le problème posé puis discuter de l'implémenation de votre solution.
- Soignez votre présentation, ne supposez pas que l'auditeur connaisse le problème (imaginez faire la présentation à un "client"). Vous serez évalué.e sur votre capacité à présenter un concept technique de façon claire et conçise.
- Le titulaire pourra fournir au moment du passage, un fichier test que les étudiants devront discuter et analyser en direct.
- Respectez le timing.

La présentation sera suivie par une ***séance de questions***. Le but de ces questions est de déterminer la contribution et l'implication de chacun. Ces questions pourront être "pointues" : rôle d'une variable dans le code, principe de fonctionnement d'une fonction particulière, détails d'implémentation, *etc.*