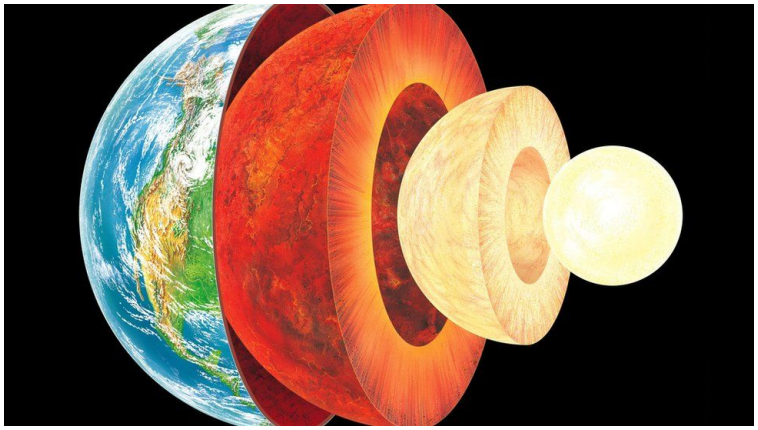
Algorithmique 2 (INFO-F203)
Dégénérescence, Coloration et Cœurs

Jean Cardinal

Université libre de Bruxelles (ULB)
Mars 2022



(Science Photo Library.)

1 Dégénérescence

Dans ce qui suit, on considérera des graphes non-dirigés, simples et sans boucle. Un *sous-graphe* d'un tel graphe G est un graphe obtenu en supprimant certains sommets et certaines arêtes de G. Un graphe G est dit *k-dégénéré* si pour tout sous-graphe de G, y compris G lui-même, il existe un sommet de degré au plus k. La dégénérescence d'un graphe est le plus petit nombre k tel que le graphe est k-dégénéré. Un exemple de graphe 3-dégénéré est donné en Figure 1.

Dans une première partie du projet, nous vous demandons de :

1. décrire un algorithme efficace pour calculer la dégénérescence d'un graphe G donné en entrée,
2. comparer la complexité de votre algorithme à une borne inférieure, et démontrer éventuellement l'optimalité de votre algorithme,

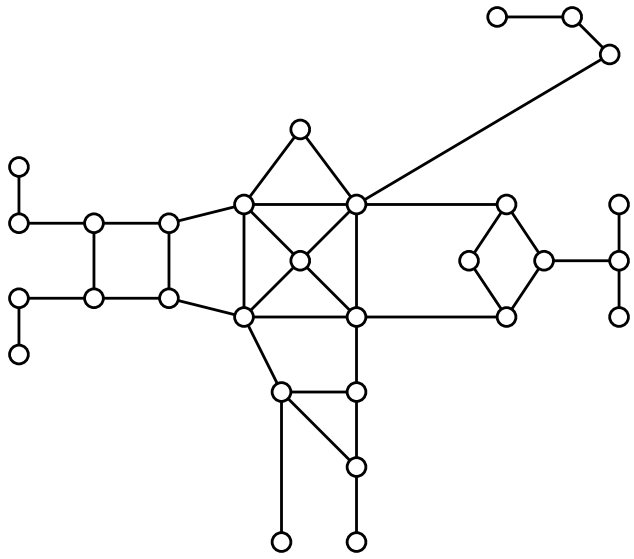


Figure 1: Exemple de graphe 3-dégénéré.

3. proposer une mise en œuvre de cet algorithme en Java.

Ce paramètre est un classique de la théorie des graphes, avec de nombreuses applications. Nous allons dans les parties suivantes de ce projet explorer deux applications de la notion de dégénérescence. La première concerne le lien entre la dégénérescence et le *nombre chromatique* d'un graphe. La seconde est l'exploration de la structure d'un graphe via la notion de *cœur*.

2 Coloration

Il est toujours possible de colorer les pays d'une carte géographique avec au plus quatre couleurs, de façon que deux pays ayant une frontière commune n'aient pas la même couleur. Ce théorème, dit "Théorème des 4 couleurs" n'a été complètement démontré que relativement récemment. Ce n'est en effet qu'en 1976 qu'une démonstration a été produite par Kenneth Appel et Wolfgang Haken à l'aide d'un ordinateur. Celle-ci a été simplifiée, puis complètement vérifiée respectivement en 1997 et en 2005.

Étant donné un graphe G (simple, non-dirigé), une *coloration propre* de G est une assignation de couleurs à ses sommets, telle que deux sommets adjacents n'ont jamais la même couleur. Le nombre minimum de couleurs dans une coloration propre de G est appelé *nombre chromatique* de G, et noté $\chi(G)$. Le Théorème des 4 couleurs est un résultat de théorie des graphes, qui dit que tout graphe *planaire* a un nombre chromatique d'au plus 4. Un exemple de coloration du graphe de la Figure 1 est donné en Figure 2.

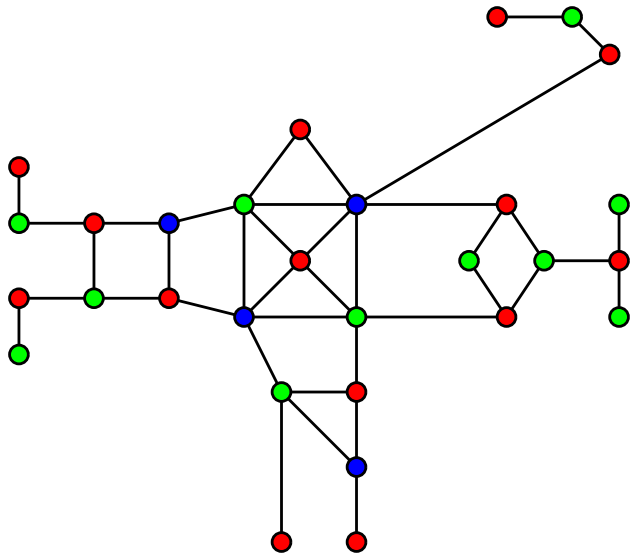


Figure 2: 3-coloration du graphe de la Figure 1.

Le problème de coloration de graphe, qui consiste à décider si un graphe donné peut être coloré avec un certain nombre k de couleurs, est un problème NP-complet, pour lequel on ne connaît pas d'algorithme efficace. Cependant, le lemme suivant donne une borne supérieure sur le nombre chromatique d'un graphe en fonction de sa dégénérescence :

Lemme 1. *Le nombre chromatique d'un graphe k-dégénéré est inférieur ou égal à $k + 1$.*

Dans cette seconde partie de votre travail, nous vous demandons de

1. donner une démonstration du Lemme 1,
2. décrire un algorithme efficace renvoyant une coloration propre d'un graphe G donné en entrée avec (au plus) $k + 1$ couleurs, où k est la dégénérescence de G,
3. comparer la complexité de votre algorithme à une borne inférieure, et démontrer éventuellement l'optimalité de votre algorithme,
4. décrire une mise en œuvre de cet algorithme en Java.

3 Dégénérescence et Cœurs

Les *k-cœurs* d'un graphe G sont les composantes connexes du graphe obtenu après avoir supprimé itérativement les sommets de degré inférieur à k. De manière équivalente, il s'agit d'un sous-graphe connexe de G pour lequel tous les sommets sont de degré au moins k et qui est maximal pour l'inclusion, donc auquel on ne peut ajouter aucun autre sommet sans perdre cette propriété. La dégénérescence d'un graphe est le plus grand nombre k tel

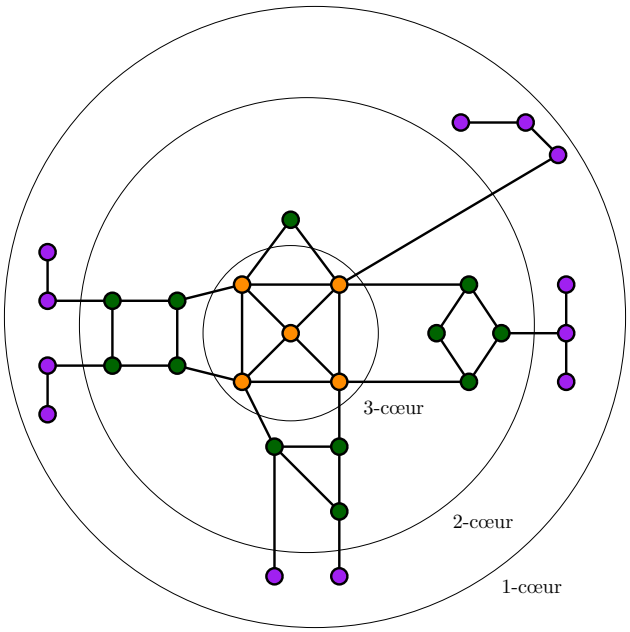


Figure 3: Cœurs d'un graphe.

que le graphe a un k-cœur.

Pour un sommet v de G, on peut définir sa *profondeur* $c(v)$ comme le plus grand nombre k tel que v appartient à un k-cœur. Un exemple est donné en Figure 3.

Dans cette troisième partie de votre travail, nous vous demandons de

1. décrire un algorithme efficace calculant, pour un graphe G donné en entrée, la profondeur $c(v)$ de chaque sommet de v,
2. comparer la complexité de votre algorithme à une borne inférieure, et démontrer éventuellement l'optimalité de votre algorithme,
3. décrire une mise en œuvre de cet algorithme en Java.

4 Expériences sur des graphes issus de données réelles

Pour tester les algorithmes ci-dessus et vérifier expérimentalement leurs complexités, nous vous proposons d'utiliser des graphes issus de données réelles recueillies par des chercheurs.

Le site web du projet SNAP de l'Université de Stanford propose une collection de grands graphes (de plusieurs milliers à plusieurs millions de sommets) sur lesquels vous

pouvez tester vos algorithmes :

`https://snap.stanford.edu/data/`.

Dans une quatrième partie du travail, nous vous demandons de concevoir une ou plusieurs expériences utilisant vos programmes sur ces données, et de commenter les résultats obtenus.

Nous vous laissons, dans cette partie plus ouverte du travail, une latitude quant à la teneur et au plan exact de vos expériences. Nous attendons cependant des conclusions sur les deux aspects suivants :

1. Les algorithmes sont-ils efficaces sur des grands graphes ? Quelle est la taille maximale des graphes que vous avez pu traiter ?
2. Quels sont les caractéristiques des différents graphes que vous avez testé, en terme de dégénérescence, de colorabilité, de cœurs? Quels sont les différents types de graphes qui se distinguent ?

Consignes

- Votre travail sera composé d'un rapport scientifique, d'un programme en Java, et d'un fichier `README`.
- Le fichier `README` est un fichier texte devant contenir les instructions permettant au correcteur de compiler, exécuter et tester le code.
- Le rapport sera composé de quatre parties correspondant au quatre sections précédentes, chacune faisant l'objet d'une évaluation.
- Pour les trois premières parties, on demande les réponses aux questions posées, une description détaillée des algorithmes mis en œuvre, et une description synthétique de l'organisation de vos programmes Java, avec un commentaire détaillé des parties les plus importantes.
- Le rapport devra être au format `pdf`, rédigé en français correct, et mis en page avec le logiciel LaTeX.
- Le rapport comprendra également la liste des références bibliographiques utilisées.
- On s'attend pour ce travail à un rapport dont la longueur est d'environ une dizaine de pages.
- Les fichiers sont à remettre sur l'Université Virtuelle (UV).

Recommandations et ressources externes

Rédaction scientifique. Des conseils généraux sur la rédaction d'un rapport scientifique se trouvent dans le document "Éléments de rédaction scientifique en informatique", rédigé par Hadrien Mélot (UMons) et disponible à l'adresse suivante :

`http://informatique.umons.ac.be/algo/redacSci.pdf`.

Ressources bibliographiques. Vous êtes encouragé·e·s, pour répondre aux questions, à consulter la littérature scientifique. Les documents utilisés doivent être clairement mentionnés dans le texte et figurer dans la liste de références bibliographiques. Nous vous encourageons à utiliser le logiciel BibTeX :

`https://www.bibtex.com/`.

Code. Les programmes doivent être dûment commentés et structurés, et respecter les bonnes pratiques enseignées dans les cours de programmation orientée objet. Vous êtes autorisé·e·s à réutiliser, en le mentionnant clairement, les classes de la bibliothèque Java `algs4.jar`, disponible à l'adresse suivante :

`https://algs4.cs.princeton.edu/code/`.

Travail en binôme, échanges et plagiat

Vous êtes autorisé·e·s à rendre le travail en collaboration avec un·e autre étudiant·e, auquel cas la même note sera attribuée au deux membres du binôme. Vous pouvez également rendre le travail à titre individuel.

Vous êtes autorisé·e·s à discuter des problèmes et de vos solutions éventuelles avec d'autres étudiant·e·s et d'autres binômes. Si un point de votre rapport est le fruit de ces discussions, vous êtes invité·e·s à le mentionner explicitement. La rédaction du rapport et des programmes doit cependant être propre à chaque étudiant·e / binôme. Tout emprunt non mentionné explicitement est un plagiat, et constitue une fraude. Les étudiant·e·s reconnu·e·s coupables de fraude ou de tentative de fraude s'exposent à des sanctions disciplinaires comprenant l'annulation de la session d'examens et l'interdiction de s'inscrire à la session d'examens suivante.

`https://bib.ulb.be/fr/support/boite-a-outils/evitez-le-plagiat`

Date de remise

29 avril 2022, avant minuit.