

MATH0499 - Projet

Graphe k -dégénéré

Groupe s180498-s170220: Martin RANDAXHE, Cyril RUSSE

Table des matières

| | | |
|----------|---|----------|
| 1 | Introduction | 3 |
| 2 | Algorithmes importants | 3 |
| 2.1 | Est-k-dégénéré | 3 |
| 2.2 | Génération de graphes k-dégénérés maximaux | 3 |
| 3 | Difficultés | 3 |
| 3.1 | Performance algorithme k-dégénéré | 3 |
| 3.2 | Cas générique pour graphes k-dégénérés maximaux | 4 |
| 4 | Utilisation des programmes | 4 |
| 4.1 | k-degenere.py | 4 |
| 4.2 | k-degenere_max.py | 5 |
| 4.3 | affichage_graphe.py | 5 |

1 Introduction

Ce projet consiste en la création de deux algorithmes sur le sujet de la k -dégénérescence de graphes.

Le premier permet tout simplement de savoir si un graphe est k -dégénéré et de fournir, dans le cas positif, une suite de noeud à supprimer le prouvant.

En second algorithme, une implémentation créant un graphe dit de k -dégénéré maximal.

2 Algorithmes importants

2.1 Est- k -dégénéré

Comme introduit dans la section précédente, cet algorithme va permettre de définir si le graphe, donné par l'utilisateur du programme, est ou non k -dégénéré.

Notre algorithme va d'abords créer une copie de la liste des noeuds du graphe. Ensuite, nous les parcourons et vérifions leur degré si celui-ci est inférieur ou égal à la valeur de k , alors celui-ci est supprimé et enregistré à la suite d'une liste qui répertorie tous ces noeuds afin de pouvoir les enregistrer par après si le graphe est k -dégéré. Si en ayant parcouru tous les noeuds restants, aucun n'est de degré correspondant aux attentes, l'algorithme s'arrête et informe l'utilisateur que le graphe n'est pas k -dégénéré. Si au contraire, il a été possible de supprimer tous les noeuds, le graphe est k -dégénéré et la liste de suite de noeud est écrite dans un fichier.

2.2 Génération de graphes k -dégénérés maximaux

Après réflexion sur les graphe k -dégénérés, nous en avons conclut que nous pouvions déjà répertorié 2 cas différents. Il se différencie en fonction de sa valeur de dégénérescence k et son nombre de noeud n . En effet, si le nombre de noeud est inférieur ou égal à $k+1$, alors tous les noeuds doivent être relié entre eux. Dès que l'on passe se cap, après analyse à la manière d'une démonstration par récurrence, nous avons remarqué que les graphes maximaux respecte un paterne tel que si l'on rajoute un noeud, il suffit de le lier au k éléments précédents.

L'algorithme crée donc les noeuds 1 par 1, et pour chacun d'eux, lie directement celui-ci respectant les deux cas énoncés ci-dessus. Il sera finalement enregistré dans un fichier.

3 Difficultés

Nous allons parcourir dans cette section les points critiques des implémentations de ce projet et de la recherche pour celles-ci.

3.1 Performance algorithme k -dégénéré

De grands dilemmes nous ont été posés pour cet algorithme. Premièrement, nous n'avons pas eu de meilleurs idées afin d'obtenir une meilleur complexité que n^2 en pire cas. La grande question par après a été de nous demander quels sont les cas les plus probables afin d'au moins obtenir un meilleur cas moyen. Le fait est que les différents cas que nous avons répertorié ne sont pas improbables. En effet, par exemple, avec notre implémentation actuelle un graphe ayant beaucoup de noeuds de degré en même temps déjà inférieur se serait vu plus efficace avec un implémentation parcourant pratiquement toute la liste à chaque boucle afin de retirer plusieurs noeuds directement. D'autre part, cette manière aurait en revanche rendu les cas de graphes fort dense et se rapprochant de graphe k -dégénéré maximaux, nécessitant quant à eux de supprimer plus régulièrement certains noeuds avant de pouvoir en supprimer d'autres

juste après, extrêmement moins efficace. Il était donc difficile de trouver une technique étant convenable quelque soit le cas. Celle que nous avons adoptée ne l'est en effet pas pour tous les cas.

3.2 Cas générique pour graphes k -dégénérés maximaux

Comme nous l'avons énoncé dans la section 2.2, afin de trouver une solution algorithmique à ce problème, il nous a été nécessaire de définir quels étaient les différents cas que nous pouvions rencontrer. Nous avons donc suivi une logique de démonstrations par récurrence et nous avons réalisé que pour quelque soit la valeur de dégénérescence, les graphes ou sous-graphes des $k+1$ premiers éléments devait être tous reliés entre eux. Ensuite, pour les noeuds suivants, chaque nouveau devra de leur côté être reliés aux k éléments précédents. A partir du moment, où nous avons bel et bien vérifié que cette logique permet de couvrir tous les cas de k et de n , l'implémentation n'a pas été la partie la plus complexe. Et la complexité de l'algorithme est linéaire suivant le nombre de noeud et des arcs nécessaires.

4 Utilisation des programmes

Nous sommes partis sur une implémentation en python en utilisant la bibliothèque "Networkx" conseillé dans l'énoncé. Nous avons choisi que les graphes qui seront donnés et que renverront les programmes suivrait un modèle de fichier "adjlist". Ce système de fichier répertorie tous les noeuds et pour chaque noeud les autres noeuds auxquels il est lié. Comme son nom l'indique, telles les listes d'adjacences. Ceux-ci sont écrit dans des fichiers au format "txt". Nous avons préféré séparé les 2 grands problèmes en 2 programmes distincts "k-degenere.py" et "k-degenere_max.py". Le premier permettant de définir si un graphe donné en argument à l'exécution est k -dégénéré ou non et le second pour la création de graphe k -dégénéré maximaux. Nous avons également fourni un troisième programme que nous aurions pu intégré dans les 2 autres car il consiste juste à afficher graphiquement un graphe dans un fichier txt à la structure adjlist. Nous avons un léger quand à l'exécution de ceux-ci sur un autre ordinateur que l'autre vu qu'elle nécessite l'installation de la bibliothèque "matplotlib".

4.1 k-degenere.py

Le programme s'exécute avec python3 et nécessite l'ajout d'argument à l'exécution. Nous allons voir en revue les options de ce programme :

- -i Nom du fichier contenant le graphe
 - -k Une valeur entière correspondant à la valeur de dégénérescence voulant être vérifiée
 - -o (optionnel) Le nom du fichier dans lequel noté la suite de noeud dans le cas où le graphe est k -dégénéré. Cette option est optionnel. Si elle n'est pas donnée le programme écrira par défaut dans un fichier "default.txt".
 - -h (optionnel) Cette option ne nécessite pas d'argument et permet juste d'afficher l'aide.
- Voici quelques exemples d'exécutions :



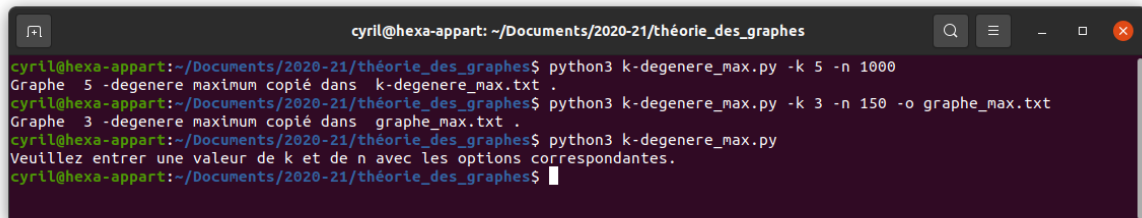
```
cyril@hexa-appart: ~/Documents/2020-21/théorie_des_graphes
cyril@hexa-appart:~/Documents/2020-21/théorie_des_graphes$ python3 k-degenere.py -i graphe_test.txt -k 3
Le graphe donné est 3 dégénéré.
La suite de noeud à supprimer a été copiée dans default.txt .
cyril@hexa-appart:~/Documents/2020-21/théorie_des_graphes$ python3 k-degenere.py -i graphe_test.txt -k 3 -o liste_noeud.txt
Le graphe donné est 3 dégénéré.
La suite de noeud à supprimer a été copiée dans liste_noeud.txt .
cyril@hexa-appart:~/Documents/2020-21/théorie_des_graphes$ python3 k-degenere.py -i graphe_test.txt -k -15
Veuillez donner une valeur de k valide.
Utilisez l'option -h pour afficher l'aide.
cyril@hexa-appart:~/Documents/2020-21/théorie_des_graphes$ python3 k-degenere.py
Veuillez donner la valeur de k-dégénérescence et le nom du fichier contenant le graphe.
Utilisez l'option -h pour afficher l'aide.
cyril@hexa-appart:~/Documents/2020-21/théorie_des_graphes$
```

FIGURE 1 – Exemples d'exécutions de k-degenere.py

4.2 k-degenere_max.py

Ce programme s'exécute de la même manière que pour k-degenere.py, mais leurs options diffèrent. Ce programme ci va produire un graphe k-degenere maximal avec n noeud. K et n sont des paramètres à transmettre à l'exécution et un nom de fichier dans lequel enregistrer le graphe au format txt suivant la structure "adjlist" peut être donné.

- -k Une valeur entière correspondante à la valeur de dégénérescence voulue.
 - -n Une valeur entière correspondante aux nombres de noeuds désiré.
 - -o (optionnel) Le nom du fichier dans lequel enregistrer le graphe nouvellement créé. Par défaut, le nom de fichier sera "k-degenere_max.txt".
 - -h (optionnel) Cette option ne nécessite pas d'argument et permet juste d'afficher l'aide.
- Voici quelques exemples d'exécutions :



```
cyril@hexa-appart: ~/Documents/2020-21/théorie_des_graphes
cyril@hexa-appart:~/Documents/2020-21/théorie_des_graphes$ python3 k-degenere_max.py -k 5 -n 1000
Graphe 5 -degenere maximum copié dans k-degenere_max.txt .
cyril@hexa-appart:~/Documents/2020-21/théorie_des_graphes$ python3 k-degenere_max.py -k 3 -n 150 -o graphe_max.txt
Graphe 3 -degenere maximum copié dans graphe_max.txt .
cyril@hexa-appart:~/Documents/2020-21/théorie_des_graphes$ python3 k-degenere_max.py
Veuillez entrer une valeur de k et de n avec les options correspondantes.
cyril@hexa-appart:~/Documents/2020-21/théorie_des_graphes$
```

FIGURE 2 – Exemples d'exécutions de k-degenere_max.py

4.3 affichage_graphe.py

Ce programme nécessite l'installation de la bibliothèque matplotlib. Ce programme permet d'afficher les graphes écrit sous forme d'"adjlist" dans des fichiers txt.

Il nécessite uniquement l'option -i suivi du nom du fichier contenant le graphe à afficher.

Voici un exemple d'affichage du programme avec l'utilisation du graphe contenu dans le fichier d'exemple "graphe_test.txt".

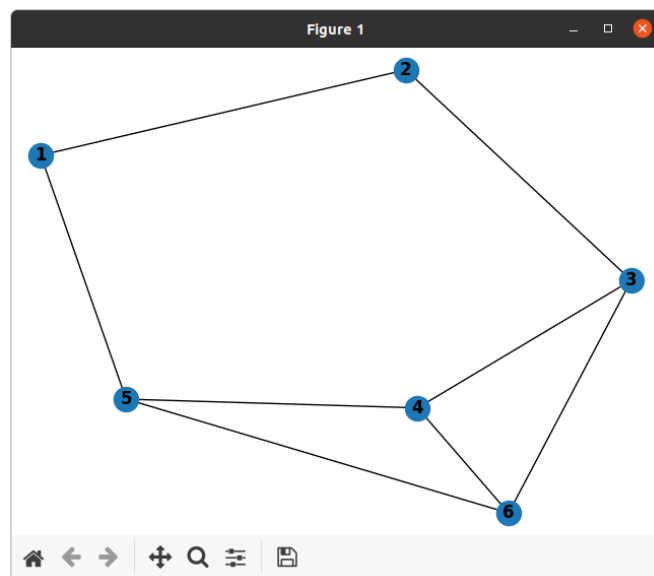


FIGURE 3 – Exemples d’affichage graphique d’un graphe