MATH0499: Projet 5 : coloriage

Maxime Deravet, Abdelilah khaliphi

Table des matières

1 Les deux parties du développement

Pour des raisons de simplicité, nous avons choisi de réaliser ce projet en Python.

Nous avons donc dû développer les fonctions qui permettaient la lecture d'un fichier, de le transformer en graphe, et de pouvoir afficher le résultat à l'écran. Tout ça représente la "première partie".

Ensuite, nous voilà au problème à proprement parler. Pour rappel, le projet 5 consiste à colorier, avec un minimum de couleur, les sommets d'un graphe simple non orienté de manière à ce que aucuns sommets adjacents ne reçoivent la même couleur. Le graphe doit être fourni dans un fichier.

Une première idée était de fonctionner de "proche en proche", et de se balader d'un sommet à un autre en fonction de leurs arêtes communes. Nous nous sommes vite rendu compte que cette méthode allait poser problème dans le cas d'un graphe non connexe.

Notre deuxième approche était d'énumérer chaque sommet dans l'ordre, et de simplement lui attribuer une couleur la plus petite possible, différente de ses voisins.

Cette méthode nous avait l'air un peu bancale au premier abord car elle ne résisterait peut-être pas à certains cas particuliers. Eh bien si ! A notre bonne surprise, elle fonctionnelle sur tous les graphes que nous avons testés. Plus de détails dans la partie "benchmarking".

2 Nos fonctions

```
#@Préconditions : matrice et ligne sont des listes vides déjà déclarées, un fichier "file", se trouvant dans le répertoire courant, correctement formaté comme montré à la page 152 du syllabus du cours de théorie des graphes édition 2009-2010
#@Postcondition: la liste matrice est remplie comme le fichier "file"
def lecture_fichier(matrice, ligne, file):
return matrice
```

Extrait de Code 1 – lecture du fichier

```
#@Préconditions: matrice est un liste contenant une matrice représentant un graphe
#@Postcondition: G est un graphe créé avec NetworkX
def convert_networkX(matrice):
return G
```

Extrait de Code 2 – Convertir la matrice en graphe

```
#@Précondition : G est un graphe correctement créé avec NetworkX
#@Postcondition: Affiche une représentation du graphe à l'écran
#Source : NetworkX documentation, "Drawing Graphs"
def draw_graph(G):
    nx.draw(G,with_labels=True, font_weight='bold')
    plt.show()
```

Extrait de Code 3 – Afficher le graphique à l'écran

```
#@Préconditions: - G est un graphe correctement créé avec NetworkX
# - couleur est une liste vide déjà initialisée
#
#@Postcondition: couleur contient une "couleur" (ici un nombre) à chaque index correspondant au sommet
def coloriage(G, couleur):
return couleur
```

Extrait de Code 4 – Coloriage des sommets

Cette dernière fonction est celle qui nous intéresse en particulier. Comme dit dans la postcondition, ce ne sont pas des couleurs à proprement parler qui seront associées aux sommets, mais bien des nombres, afin de pouvoir en créer une infinité.

Par exemple :

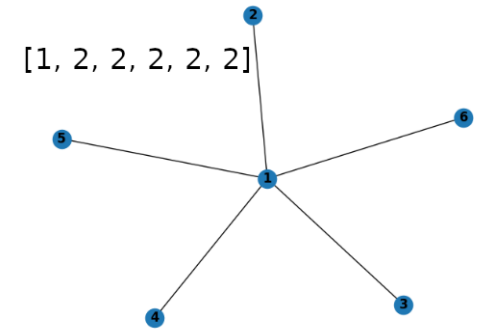
— le sommet 1 a la couleur 1
— le sommet 2, la couleur 2
— le sommet 3, la couleur 1

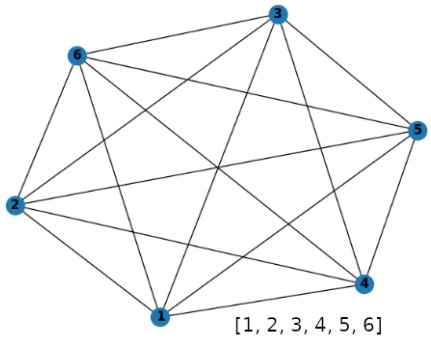
Alors la liste couleur se présentera comme ceci : [1,2,1]

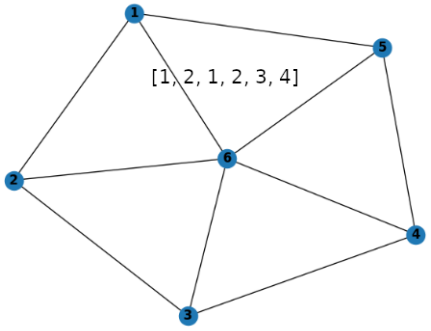
NB : le programme a l'air efficace, mais il ne l'est probablement pas à 100%. Nous n'avons cependant pas encore rencontré de graphe qui produisait un résultat erroné

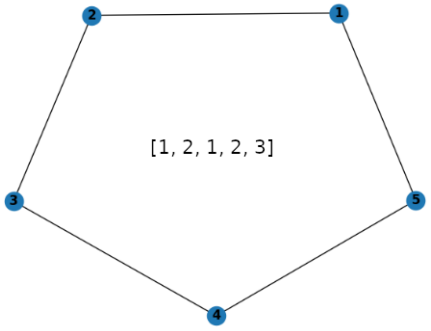
3 Benchmarking

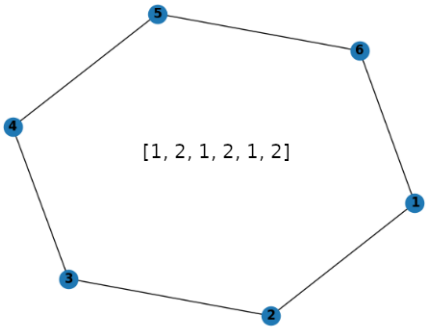
Nous avons effectué 7 tests représentatifs avec divers types de graphes, voici leur résultat graphique ainsi que leur liste couleur associées à chacun :

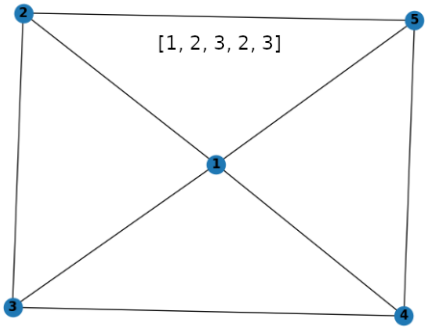


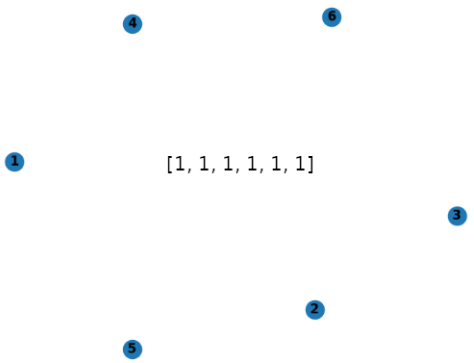












Tous ces résultats ont été obtenu en moyenne en 0.25 secondes, sur un ordinateur standard