TP N°05 : COLORATION DE GRAPHES

OBJECTIFS DU TP

- Modéliser un problème à l'aide d'un graphe ;
- Utiliser la coloration des graphes pour résoudre un problème.

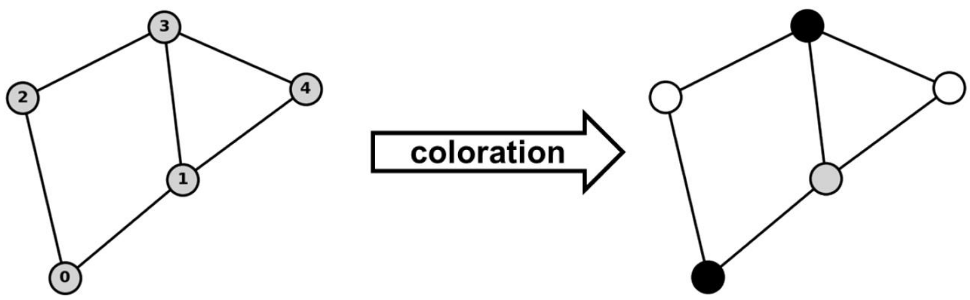
Vous pouvez utiliser le script d'affichage de graphe disponible sur cahier de prépa.

Considérons un lycée où les étudiants peuvent suivre différents enseignements disciplinaires, indépendamment de la classe à laquelle ils appartiennent. L'établissement souhaite organiser une semaine d'examens blancs. Cependant, il faut veiller à ce qu'aucun étudiant n'ait deux examens sur un même créneau horaire. Nous allons modéliser ce problème par une coloration de graphe.

Construisons un graphe dont les sommets sont les différents enseignements. Une arête relie deux sommets si les enseignements correspondants possèdent des étudiants en commun.

On va alors chercher à colorier les sommets de telle sorte que deux sommets adjacents soient de couleurs différentes. C'est ce qu'on appelle une **coloration**. On pourra alors organiser les examens correspondants aux enseignements de sommets de même couleur sur un même créneau horaire car ils n'ont pas d'étudiants en commun.

Voici un **exemple** de 3-coloration du graphe suivant :



Considérons le tableau de répartition ci-dessous. Lorsque deux enseignements possèdent des élèves en communs, la cellule correspondante contient un 1, sinon un 0. On note G(S,A) le graphe associé à ce problème.

Spécialités		LLCE	PC	SVT	SI	NSI	Maths
	Sommet associé	0	1	2	3	4	5
LLCE	0	0	0	0	0	1	1
PC	1	0	0	1	1	1	0
SVT	2	0	1	0	1	0	1
SI	3	0	1	1	0	1	1
NSI	4	1	1	0	1	0	0
Maths	5	1	0	1	1	0	0

Q1. Représenter le graphe G(S,A). Proposer une 5-coloration et une 4-coloration de ce graphe.

En programmation, réaliser une coloration consiste donc à construire une liste C où chaque élément C[i] correspond à la couleur du sommet s_i. En associant une leur à chaque couleur (par exemple : 0 pour blanc, 1 pour gris, 2 pour noir), la 3-coloration du graphe donné en exemple peut donc s'écrire :

C = [2,1,0,2,0]

Q2. Proposer deux listes C_5 et C_4 correspondant respectivement à une 5-coloration et une 4-coloration du graphe G(S,A). Est-il possible de proposer une 2-coloration ?

Le **nombre chromatique χ** associé au graphe G est le **nombre minimal de couleurs** nécessaires pour réaliser une coloration du graphe G.

On souhaite souvent utiliser un minimum de couleur pour réaliser une coloration. Dans ce problème, minimiser le nombre de couleurs revient à minimiser le nombre créneaux d'examens et ainsi minimiser le temps consacré aux examens blancs.

Q3. Justifier que quel que soit le graphe G, $1 \leq \chi(G) \leq n$, avec n l'ordre du graphe.

Document 1 : Théorème de Brooks

Il est possible d'affiner la borne supérieure de cet encadrement au moyen du théorème de Brooks :

$$\chi(G) \leq \Delta(G) + 1.$$

Avec $\Delta(G)$ le degré maximal du graphe G(S,A) :

$$\Delta(G) = \max\{\, d(s)\}\, avec\ s \in S$$

Q4. Écrire une fonction `delta_max` prenant comme argument la matrice d'adjacence G associée au graphe G, et retournant le degré maximal Δ(G) de ce graphe.

Il est possible de réaliser une coloration d'un graphe à l'aide de l'algorithme de coloration gloutonne :

- On choisit un sommet s non-colorié de G ;
- On détermine l'ensemble des voisins de ce sommet s_i ;
- En considérant les voisins déjà coloriés, on établit une liste de couleurs déjà utilisées par ce sommet ;
- On colorie ce sommet avec la première couleur (par rang croissant) dans la liste de couleurs disponibles ;
- On passe au sommet suivant, puis on répète l'opération jusqu'à obtenir un graphe totalement colorié.

Q5. Écrire une fonction `voisins` prenant comme arguments la matrice d'adjacence G associée au graphe G et le rang i du sommet s_i considéré, et retournant la liste des indices j des voisins s_j du sommet s_i.

Q6. Écrire une fonction `glouton` prenant comme argument la matrice d'adjacence G associée au graphe G, et retournant une (Δ(G)+1)-coloration du graphe G sous la forme d'une liste de couleurs C (chaque couleur étant repérée par un indice allant de 0 à Δ(G)). On pourra utiliser la fonction `voisins`.

Si besoin, des aides pour cette question sont disponibles ci-dessous :

Aide Q6 : Vous aurez au minimum besoin de créer 4 listes différentes, bien identifiées :

- *une liste contenant la liste de toutes les couleurs, (couleur)*
- *une liste contenant des booléens pour savoir si le sommet i est coloré ou non, (colorie)*
- *une liste contenant les couleurs disponibles pour coloré un sommet i (couleur_dispo)*
- *une liste correspond à la coloration qu'il faut renvoyer. (coloration)*

Q7. Vérifier le bon fonctionnement de la fonction `glouton` en utilisant l'argument optionnel `node_color` de la méthode `draw` de la bibliothèque `networkx`. Par exemple, en supposant le graphe G d'ordre 4 déjà déclaré, la suite d'instructions :

```
>>> import networkx as nx
>>> import matplotlib.pyplot as plt
>>> nx.draw(G, node_color = ["r", "g", "r", "g"])
>>> plt.show()
```

conduit à l'affichage d'une coloration du graphe G où les sommets d'indice pair sont affichés en rouge et les sommets d'indice impair sont affichés en vert.

Sous-questions (optionnelles) pour vous aider :

a. *Ecrire une liste d'instruction permettant de générer aléatoirement la matrice d'adjacence d'un graphe non-orienté G de d'ordre n. On pourra utiliser la fonction* `randint` *de la bibliothèque* `random`.

b. *Effectuer la coloration du graphe.*

c. *Ecrire une liste d'instruction permettant de générer une liste de couleurs utilisable en argument* `node_color` *de la méthode* `draw` *quel que soit l'ordre du graphe G.*

d. *Ecrire une liste d'instruction permettant d'afficher la coloration du graphe en utilisant l'argument optionnel* `node_color` *de la méthode* `draw` *de la bibliothèque* `networkx`.

Comme indiqué par le théorème de Brooks, la (Δ(G)+1)-coloration du graphe G obtenue par la mise en œuvre d'un algorithme glouton n'est pas toujours optimale.

Q8. Proposer :

a. Un exemple de graphe G d'ordre 5 de nombre chromatique $\chi(G) = \Delta(G) + 1$;

b. Un exemple de graphe G d'ordre 5 de nombre chromatique $\chi(G) < \Delta(G) + 1$.

Q9. Proposer un algorithme permettant de vérifier, par force brute, si un graphe G admet une k-coloration, pour une valeur de k donnée ($1 \leq k \leq \Delta(G) + 1$). Quelle est la complexité de cet algorithme (exprimée en notation de Landau) dans le pire des cas ?

Remarque : L'attaque par force brute est une méthode utilisée en cryptanalyse pour trouver un mot de passe ou une clé. Il s'agit de tester, une à une, toutes les combinaisons possibles jusqu'à en trouver une qui fonctionne.

Q10. Écrire une fonction `force_brute` prenant comme argument la matrice d'adjacence G associée au graphe G et un entier k, et retournant une k-coloration du graphe G sous la forme d'une liste de couleurs C (chaque couleur étant repérée par un indice allant de 0 à k -1). Si le graphe G n'admet pas de k-coloration, la fonction coloration retourne une liste vide. On pourra utiliser la fonction `voisins`.

Q11. Ecrire une fonction `coloration_optimale` prenant comme argument la matrice d'adjacence G associée au graphe G, et retournant une Δ(G)-coloration du graphe G sous la forme d'une liste de couleurs C (chaque couleur étant repérée par un indice allant de 0 à Δ(G) - 1). On utilisera la fonction `force_brute`.

Q12. Comparer les résultats obtenus pour les fonctions `glouton` et `coloration_optimale` sur le problème posé. Conclure.