DM3

I. Exercice 100 p.204 (*Manuel Mathématiques Expertes - Le Livre Scolaire*)

Partie A : Étude du problème

- **Colorier un graphe**, c'est affecter une couleur à chaque sommet, de sorte que deux sommets adjacents ne portent pas la même couleur.
- Le **nombre chromatique**, noté γ, est le plus petit nombre de couleurs nécessaires pour colorier un graphe.
- Considérons un graphe $\mathcal{G}$.
  Un **sous-graphe** de $\mathcal{G}$ est un graphe $\mathcal{G}'$ composé de certains sommets de $\mathcal{G}$ et de toutes les arêtes reliant ces sommets.

Notons m l'ordre du plus grand des sous-graphes complets de $\mathcal{G}$ et Δ le plus grand degré des sommets de $\mathcal{G}$.

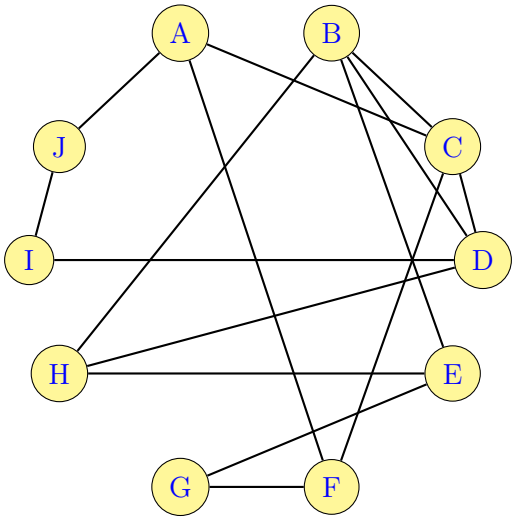
1. Démontrer : $m \leqslant \gamma \leqslant \Delta + 1$.

Partie B : Algorithme de Welsh-Powell

Pour colorier un graphe, on utilise l'algorithme suivant.

- **Étape 1** : Lister les sommets par ordre de degré décroissant/
- **Étape 2** : Attribuer une couleur C_1 au premier sommet de la liste.
- **Étape 3** : Attribuer cette même couleur à tous les sommets qui ne sont pas adjacents avec le premier sommet de la liste et qui ne sont pas adjacents entre eux.
- **Étape 4** : Répéter les étapes 2 et 3 tant que tous les sommets ne sont pas coloriés.

2. Colorier le graphe ci-dessous à l'aide de cet algorithme.



II. Parcours en profondeur

On travaillera sur le fichier `DM3.py`. Pour cela, on l'ouvrira dans une interface **Python** de son ordinateur (ceux d'entre vous qui ne possèdent pas encore **Python** peuvent télécharger Anaconda au lien suivant : https://www.anaconda.com/products/individual). Toute interface graphique convient (Anaconda fournit l'interface Spyder).
Il faut compléter et commenter ce fichier lorsque demandé et me le rendre par mail, avec votre DM scanné à la date de rendu fixée au 27 avril 2021.

1. En ligne 9, on définit la matrice d'adjacence M d'un graphe $\mathcal{G}$.
Écrire cette matrice sur votre copie et tracer le graphe $\mathcal{G}$ correspondant.

2. On définit ensuite la fonction `ordre`. Que renvoie l'appel de la ligne 19 ? Est-ce cohérent avec votre réponse en question précédente ?

3. On souhaite ensuite créer la fonction `matAdj_to_listesAdj` qui prend en paramètre la matrice d'adjacence M d'un graphe $\mathcal{G}$ et renvoie une liste `listesAdj` dont le $i^{\text{ème}}$ élément est la liste des sommets adjacents au sommet i.

a) Compléter le script fourni dans `DM3.py` (entre les lignes 23 à 37).

b) On fournit en ligne 43 une manière d'obtenir le même résultat que la fonction `matAdj_to_listesAdj` en utilisant des compréhensions de listes.
Que fait l'appel de la ligne 44 : `print(L1 == listesAdj)` ?

4. Les lignes 48 à 57 permettent de définir une fonction `liste_degres` qui prend en paramètre la matrice d'adjacence M d'un graphe $\mathcal{G}$ et renvoie la liste `listeDegres` dont le $i^{\text{ème}}$ élément est le degré du sommet i.

a) Que permet de faire l'opérateur + en ligne 56 : `listeDegres = listeDegres + [len(l)]` ?

b) En ligne 63, **en utilisant une compréhension de liste**, stocker dans la variable `L2` la liste dont le $i^{\text{ème}}$ élément est le degré du sommet i du graphe $\mathcal{G}$ de matrice d'adjacence M. (Décommenter cette ligne une fois complétée)
(on pourra utiliser la liste `listesAdj` *ou la liste* `L1`*)*

c) Comment vérifier que votre commande (à l'aide de la compréhension de liste) fournit bien le même résultat que la fonction `liste_degres` ?

5. On définit une fonction `mystere1` (de la ligne 68 à la ligne 85).

a) Que renvoie la fonction `mystere1` ?

b) Que renvoie l'appel en ligne 88 : `M3 = mystere1(mystere1(M, M), M)` ?

c) On rappelle que M est la matrice d'adjacence d'un graphe $\mathcal{G}$. D'après le cours, à quoi correspond la variable `M3` pour le graphe $\mathcal{G}$?

6. On définit une fonction `mystere2` (de la ligne 92 à la ligne 102).

a) Que renvoie la fonction `mystere2` ?

b) Que renvoie l'appel en ligne 105 : `N = mystere2(M, 3)` ?

c) On compare en ligne 108 (`print(M3 == N)`) le contenu des variables `M3` et `N`. Qu'obtenez-vous ?

7. La fonction `admet_cycle` prend en paramètre la matrice d'adjacence M d'un graphe $\mathcal{G}$ et renvoie un booléen.

a) Si la fonction `admet_cycle` renvoie `True`, que cela signifie-t-il pour le graphe $\mathcal{G}$?

b) Si la fonction `admet_cycle` renvoie `False`, que cela signifie-t-il pour le graphe $\mathcal{G}$?

c) Cela est-il cohérent avec les résultats énoncés dans le cours ?

8. La fonction `est_eulerien` prend en paramètre la matrice d'adjacence M d'un graphe $\mathcal{G}$ et renvoie un booléen.

a) Si la fonction `est_eulerien` renvoie `True`, que cela signifie-t-il pour le graphe $\mathcal{G}$?

b) Si la fonction `est_eulerien` renvoie `False`, que cela signifie-t-il pour le graphe $\mathcal{G}$?

c) Cela est-il cohérent avec les résultats énoncés dans le cours ?

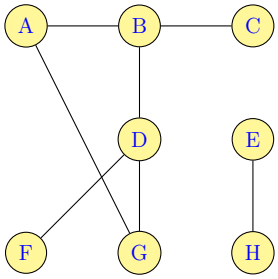
9. On donne les informations suivantes :

- la fonction `explorer` est la fonction d'exploration à partir d'un sommet définie dans le cours.
- la fonction `arbre_parcours_profondeur` est l'algorithme de parcours en profondeur défini dans le cours.
- la variable `marque` est la liste dont le $i^{\text{ème}}$ élément est un booléen correspondant au statut du sommet i : le $i^{\text{ème}}$ élément de cette liste est donc `False` si le sommet i n'est pas marqué, et `True` si le sommet i est marqué.
- la variable `predecesseur` est la liste dont le $i^{\text{ème}}$ élément est le sommet qui a permis de marquer le sommet i. Le $i^{\text{ème}}$ élément de cette liste est donc le prédécesseur du sommet i dans l'algorithme de parcours en profondeur.

a) À quoi servent les lignes 164 (`marque = [False] * n`) et 165 (`predecesseur = [-1] * n`) ?
(on pourra tester l'appel `[0] * 3` *dans la console)*

b) Compléter la fonction `arbre_parcours_profondeur` pour qu'elle réalise l'algorithme de parcours en profondeur.

c) La variable `P` contient la matrice d'adjacence définie par le graphe étudié dans le cours au moment du parcours en profondeur.



Le dernier appel `pred = arbre_parcours_profondeur(P)` renvoie la liste suivante :

$$[-1,\ 0,\ 1,\ 1,\ -1,\ 2,\ 2,\ 4]$$

(i) Comment repère-t-on les sommets racines des arbres obtenus par l'algorithme de parcours en profondeur ?
(on rappelle que chaque arbre fourni par l'algorithme de parcours en profondeur est une composante connexe du graphe $\mathcal{G}$)

(ii) Comment repère-t-on les fils de ces sommets racines ?

(iii) Dessiner alors les arbres fournis par le parcours en profondeur obtenu ici.

10. `BONUS` : Créer une fonction `comp_connexes` qui prend en paramètre la liste des prédecesseurs de chaque sommet d'un graphe $\mathcal{G}$ (obtenue à l'aide de la fonction `arbre_parcours_profondeur`) et renvoie la liste des composantes connexes de ce graphe.