L3 INF 14 décembre 2020

Théorie et Algorithmique des Graphes

Contrôle de TP n°2

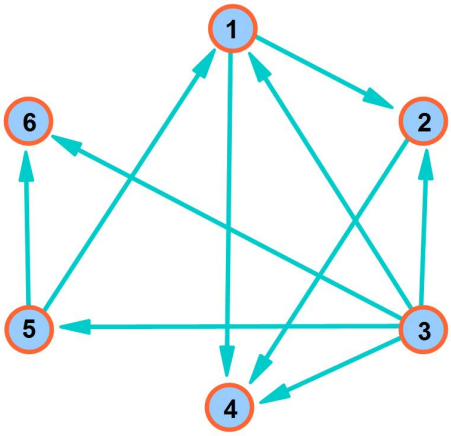
(2h – Aucun document autorisé)

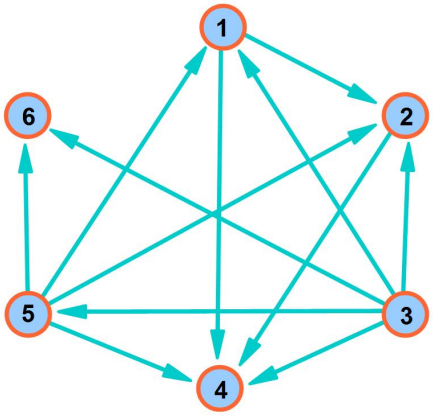
Commencez par effectuer le QCM disponible sur Moodle (20 mn), puis récupérez sur Moodle les trois fichiers examTP2exo_i.py. Copiez les sur votre machine et renommez les nom.prénom.exo_i.py

Toutes les fonctions seront testées sur les graphes fournis en exemple. Les trois exercices sont indépendants.

Exercice 1 : Graphes orientés transitifs

Dans cet exercice, G est un graphe orienté simple et connexe, représenté par listes d'adjacences. Les fonctions seront testées sur les deux graphes suivants :





Compléter les fonctions suivantes :

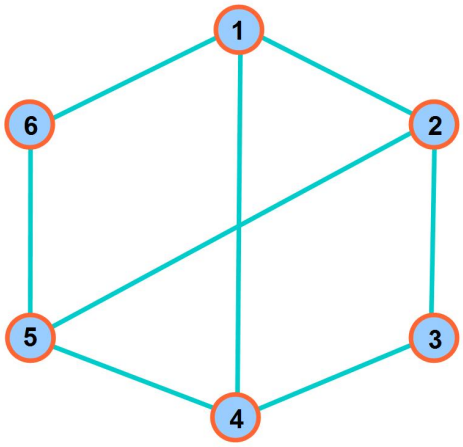
1) Écrire la fonction ***toMat(G) :*** qui calcule la matrice d'adjacence du graphe à partir des listes d'adjacence.

2) Un graphe est dit *transitif* ssi pour tous sommets i,j,k de G, si les arcs (i,j) et (j,k) sont dans G alors l'arc (i,k) est aussi dans G. Le premier graphe n'est pas transitif car il possède les arcs (5,1) et (1,4), mais pas (5,4). Le deuxième graphe est transitif.

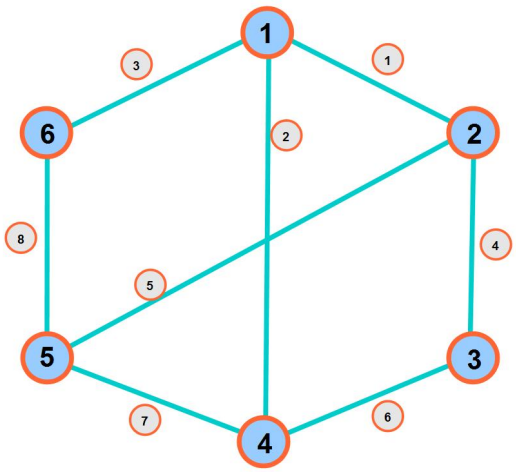
Écrire la fonction ***isTransitif(G)***, testant si le graphe est bien transitif. On pourra utiliser la matrice d'adjacence pour vérifier l'existence d'un arc (i,k).

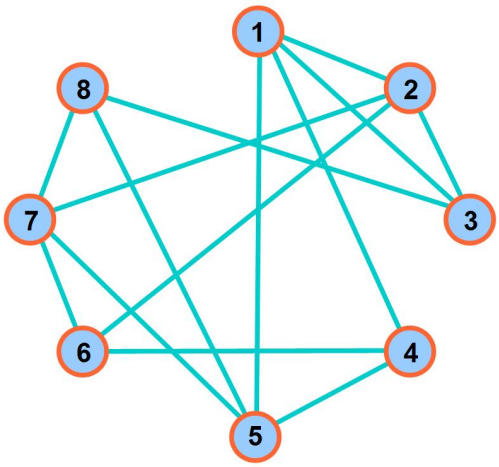
Exercice 2 : Graphe adjoint d'un graphe non orienté

Dans cet exercice, G est un graphe non orienté simple et connexe, représenté par listes d'adjacences. Les fonctions seront testées sur le graphe suivant :



Le graphe adjoint d'un graphe G est un graphe non orienté G_{adj} dont les sommets sont les arêtes de G et tel que si deux arêtes sont incidentes dans G (ont un sommet en commun), alors elles sont reliées par une arête dans G_{adj}. Partant du graphe G précédent, on peut commencer par numéroter ses arêtes, puis construire son graphe adjoint : Par exemple, dans le graphe adjoint, il y a une arête entre 7 et 8 car les arêtes de numéros 7 et 8 sont incidentes en 5 dans G.





1) Écrire la fonction ***nbAretes(G)*** donnant le nombre d'arêtes de G.

2) Écrire la fonction ***numAretes(G)*** ***(***inspirée de ***toMat(G)),*** qui construit une matrice M donnant le numéro des arêtes. Pour le graphe G précédent, on obtient la matrice de numérotation : M[(4,6)] vaut 0 car il n'y a pas d'arête entre 4 et 6 dans G.

0	1	0	2	0	3
1	0	4	0	5	0
0	4	0	6	0	0
2	0	6	0	7	0
0	5	0	7	0	8
3	0	0	0	8	0

M[(4,5)] vaut 7 car l'arête entre 4 et 5 est la 7ème arête de G.

3) Écrire la fonction ***adjoint(G) :*** permettant de construire le graphe adjoint du graphe G.

Exercice 3 : Coloration d'un graphe non orienté

Dans cet exercice, G est un graphe non orienté simple, représenté par listes d'adjacences. Les fonctions seront testées sur le graphe suivant :

Compléter les fonctions suivantes :

1) **plusPetitPasDans(L)** : qui détermine le plus petit entier ≥1 qui n'appartient pas à la liste L. On se servira de cette fonction pour déterminer la plus petite couleur n'appartenant pas à la liste des couleurs interdites.

2) **colorNaive(G)** : qui détermine une coloration du graphe G par l'algorithme naïf.

L'algorithme de coloration Dsatur, est un algorithme de type heuristique, mis au point en 1979 par la mathématicien et homme politique suisse Daniel Brélaz. Il permet de réduire sensiblement le nombre de couleurs utilisées par l'algorithme de coloration naïve.

On appelle indice de saturation d'un sommet non coloré, le nombre de couleurs déjà utilisées pour colorier ses voisins. Un sommet coloré est d'indice nul. L'idée de l'algorithme Dsatur est simplement d'effectuer une coloration naïve en traitant les sommets dans l'ordre décroissant de leur indice de saturation. En cas d'égalité on traite d'abord le sommet de degré maximal.

3) Compléter la fonction **Dsatur(G)** qui détermine une coloration du graphe G par l'algorithme Dsatur. On pourra utiliser un dictionnaire colorV qui à un sommet i associe l'ensemble des couleurs de ses voisins.