TP N°04 : REPRESENTATIONS DE GRAPHES

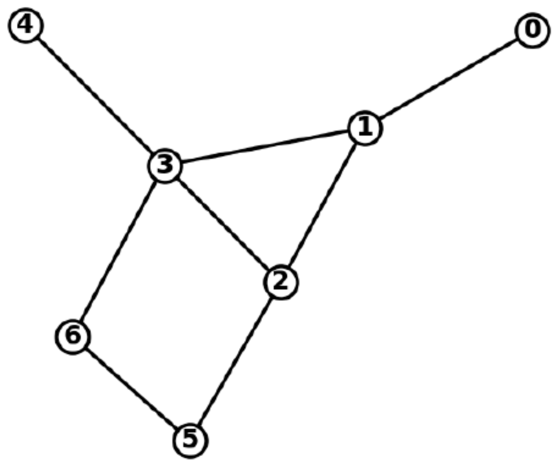
OBJECTIFS DU TP

- Représenter un graphe à l'aide d'une liste et d'une matrice d'adjacence.
- Déterminer de degré, la taille d'un graphe ainsi que l'ordre entrant et sortant des sommets.

I. REPRESENTATION DE GRAPHES EN UTILISANT LES LISTES D'ADJACENCE

I.1. Quelques manipulations classiques autour des graphes

On considère le graphe G(S,A) ci-dessous :



Q1. Ecrire la liste d'adjacence L du graphe G(S,A).

Q2. Ecrire un fonction `ajout_sommet` qui prend en argument une liste L correspondant à la liste d'adjacence d'un graphe G et ajoutant un sommet à ce graphe. La fonction retourne la nouvelle liste d'adjacence du graphe.

Q3. Ecrire une fonction `ajout_arete` qui prend en argument une liste L correspondant à la liste d'adjacence d'un graphe G et deux entiers i et j représentant deux sommets du graphe G, permettant de créer, si besoin, une arête entre deux sommets. La fonction renvoie la nouvelle liste d'adjacence.

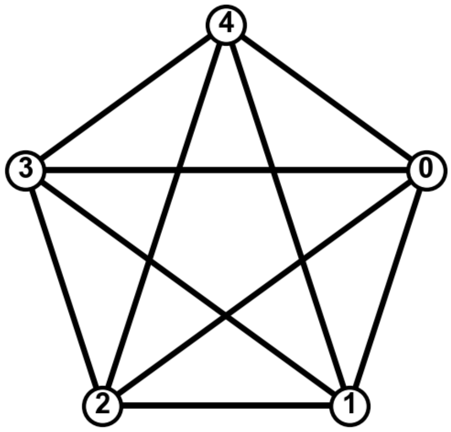
Q4. Ecrire une fonction `ordre` prenant en argument une liste L correspondant à la liste d'adjacence d'un graphe G et retournant un entier correspondant à l'ordre du graphe G.

Q5. Ecrire une fonction `degre` prenant en argument une liste L correspondant à la liste d'adjacence d'un graphe G et un entier i correspondant à un sommet du graphe et retournant un entier d correspondant au degré du sommet i dans le graphe G.

Q6. Ecrire une fonction `sont_voisins` prenant en argument une liste L correspondant à la liste d'adjacence d'un graphe G et deux entiers i et j représentant deux sommets du graphe non-orienté G et retournant le booléen `True` si les deux sommets sont adjacents, et `False` sinon.

Q7. Ecrire la matrice d'adjacence M du graphe G(S,A). Comparer la taille de l'espace mémoire pour stocker un graphe sous forme d'une liste d'adjacence ou sous forme d'une matrice d'adjacence.

Soit H(S,A), le graphe ci-dessous :



Q8. Représenter la matrice et la liste d'adjacence du graphe H(S',A') ci-dessus. Conclure sur l'avantage de la représentation à partir d'une matrice d'adjacence. Dans quels cas cette utilisation est-elle pertinente ?

En fonction du problème rencontré, il peut donc être plus pertinent de manipuler une liste d'adjacence plutôt qu'une matrice et inversement.

Q9. Ecrire une fonction `matrice_vers_liste` qui prend en argument une matrice M correspondant à la matrice d'adjacence d'un graphe G et renvoyant la liste d'adjacence du graphe G correspondant.

Q10. Ecrire une fonction `liste_vers_matrice` qui prend en argument une liste L correspondant à la liste d'adjacence d'un graphe G et renvoyant la matrice d'adjacence du graphe G correspondant.

II. Utilisation de graphes tournois

Dans cette partie, on utilisera la représentation des graphes sous forme de matrices d'adjacence.

Les tournois forment une famille de graphes utilisés notamment dans la modélisation des systèmes électoraux. Cette famille de graphes a été introduite par Landau dans les années 1950 afin de modéliser les relations de domination dans un groupe de... poulets.
Un tournoi est un graphe orienté construit à partir d'un graphe complet (c'est-à-dire d'un graphe non-orienté dont les sommets sont tous adjacents deux-à-deux) en orientant chaque arête (s_i, s_i) afin d'obtenir soit l'arc (si, sj), soit l'arc (sj, si). Nous utiliserons ce type de graphe dans le cadre du tournoi d'un jeu.

II.1. Jeu d'enfant

On s'intéresse à un jeu nommé J pour lequel les parties se jouent entre deux joueurs ; pour chaque partie du jeu J, il y a un gagnant et un perdant, il n'y a pas de match nul.

On considère une compétition du jeu J effectuée par n joueurs. Chaque joueur joue une seule fois au jeu J contre chaque autre joueur. Chaque joueur est représenté par un sommet d'un graphe non-orienté T, une arête de ce graphe entre les sommets s_i et s_j correspond à une partie du jeu J entre les joueurs i et j.
Par convention, avant le tournois, $T_{ij}=1\ \forall(i,j) \in [\![0, n-1]\!]^2$ *tel que* $i \neq j$. Un joueur ne pouvant pas jouer contre lui-même, T_{ii} vaut 0.

Q11. Représenter le graphe T d'un tournoi à 4 joueurs. Ecrire la matrice d'adjacence T d'un tournois à 4 joueurs.

Pour organiser le tournois, il faut s'assurer que tous les joueurs jouent entre eux. Vérifier cette condition revient à vérifier que le graphe T est complet.

Graphe complet

On rappelle la définition d'un **graphe complet** :
Un graphe (non-orienté) complet G(S,A) vérifie la propriété :

$$\forall(s_i, s_j) \in S^2 \text{ tel que } i \neq j, \qquad (s_i, s_j) \in A^2$$

Q12. Ecrire une fonction `est_complet` prenant comme arguments la matrice d'adjacence G associée au graphe (non-orienté) G(S,A), et retournant le booléen `True` si le graphe G est complet, `False` sinon.

Q13. Ecrire une fonction `construction` prenant en argument un entier n correspondant à un nombre de joueurs et renvoyant la matrice d'adjacence T du graphe T(S,A) décrivant le tournoi.

Q14. Proposer une suite d'instruction pour construire la matrice d'adjacence T du graphe d'un tournoi à 4 joueurs.

Lors de l'organisation du tournois, il peut être nécessaire d'ajouter des joueurs au jeu au dernier moment. Cela revient à ajouter un sommet au graphe T.

Q15. Ecrire une fonction `ajout_joueur` prenant en argument une matrice T représentant la matrice d'adjacence d'un graphe T(S,A) et ajoutant un sommet au graphe T et en ajoutant les parties entre les autres joueurs et celui-ci.

Q16. Proposer une suite d'instruction pour ajouter un 5ème joueur au tournoi précédent.

Q17. Ecrire une fonction `partie` qui prend en argument deux entiers i et j correspondant à deux joueurs du tournoi et renvoyant aléatoirement l'entier i ou j correspondant au vainqueur de la partie. On pourra utiliser la fonction `choice` de la bibliothèque `random`.

Q18. Ecrire une fonction `tournoi` qui prend comme argument la matrice d'adjacence T associées au graphe (non-orienté) T(S,A) et qui retourne la matrice d'adjacence R du graphe orienté R(S,A) représentant les résultats du tournois avec la convention suivante : l'arc est orienté du gagnant vers le perdant.

Q19. Proposer une suite d'instruction pour afficher la matrice d'adjacence R des résultats du tournoi de matrice d'adjacence initiale T.

Le vainqueur du tournoi est le joueur qui a remporté le plus de parties.

Q20. Ecrire une fonction `vainqueur` prenant en argument un graphe T représentant les parties d'un tournoi et renvoyant :

- un entier correspondant au numéro du gagnant ou
- une liste avec le numéro des gagnants en cas d'égalité

Q21. Déterminer le vainqueur du tournoi.

II.2. Marquer des points

La plupart du temps, lors d'un match ou d'un jeu c'est le nombre de points marqués qui permet de déterminer le gagnant, il est donc nécessaire de **pondérer** les graphes avec des scores, le cas de match nul ou d'égalité devient alors possible.

Q22. Ecrire une fonction `score_parties` prenant en argument une matrice T correspondant à la matrice d'adjacence d'un graphe tournoi et renvoyant une matrice V telle que le terme V[i][j] correspondent au nombre de points marqués par le joueur i lors du match contre j.

On utilisera `randint` de la bibliothèque `random`. Les scores pourront être fixés entre 0 et 50.

Afin de classer les différents joueurs (sommets), on attribue un score a(s_i) (selon la définition de Copeland) à chaque sommet $s_i \in S$, défini de la manière suivante :

$$a(s_i) = \sum_{s_j \in S} r(s_i, s_j)$$

Avec

$$r(s_i, s_j) = \begin{cases} +1 \; si \; le \; score \; du \; joueur \; i \; est \; supérieur \; à \; celui \; de \; j \\ -1 \; si \; le \; score \; du \; joueur \; i \; est \; inférieur \; à \; celui \; du \; joueur \; j \\ 0 \; sinon \end{cases}$$

Le vainqueur du tournoi (au sens de Copeland) est le sommet s_v tel que $a(s_v) = \max\{s_i\}_{s_i \in S}$.

Q23. Ecrire une fonction `vainqueur_Cop` prenant en argument une matrice V correspondant à la matrice d'adjacence du graphe orienté pondéré V(S,A) contenant les scores des parties et renvoyant :

- un entier correspondant au numéro du gagnant au sens de Copeland ou
- une liste avec le numéro des gagnants en cas d'égalité.