DS d'informatique N°1 – Réseaux sociaux

Certaines questions demandent d'écrire du code python, vous veillerez autant que possible à utiliser une syntaxe valide et à indenter correctement le code (utilisez des lignes verticales pour marquer les différents niveaux d'indentation).

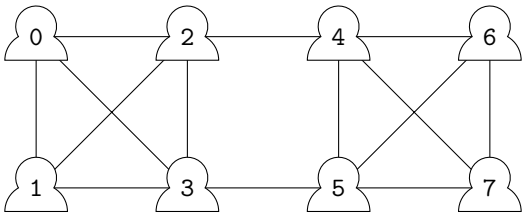
Structure de données. Nous supposerons que les individus sont numérotés de 0 à $n-1$ où n est le nombre total d'individus. Nous représenterons chaque lien d'amitié entre deux individus `i` et `j` par une liste contenant leurs deux numéros dans un ordre quelconque, c.-à-d. par la liste `[i,j]` ou par la liste `[j,i]` indifféremment.

Un réseau social R entre n individus sera représenté par une liste `reseau` à deux éléments où :

— `reseau[0] =` n contient le nombre d'individus appartenant au réseau ;

— `reseau[1] =` la liste non-ordonnée (et potentiellement vide) des liens d'amitié déclarés entre les individus.

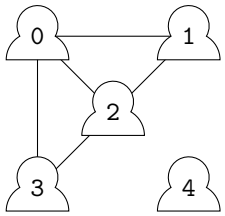
La figure 1 donne l'exemple d'un réseau social et d'une repréentation possible sous la forme de liste. Chaque lien d'amitié entre deux personnes est représenté par un trait entre elles.



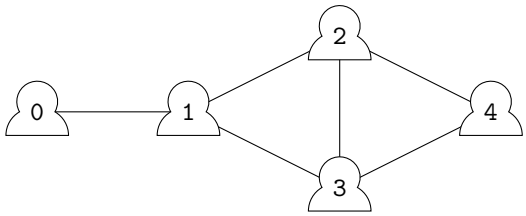

```
reseau = [8,
          [ [0,1], [1,3], [3,2], [2,0], [0,3], [2,1], [4,5],
            [5,7], [7,6], [6,4], [7,4], [6,5], [3,4], [5,3] ]
          ]
```

FIGURE 1 – Un réseau à 8 individus ayant 14 liens d'amitié déclarés et une de ses représentations possibles en mémoire sous forme d'une liste

1. Donner une représentation sous forme de liste pour chacun des deux réseaux sociaux ci-dessous :



Réseau A



Réseau B

2. Écrire une fonction `creerReseauVide(n)` qui renvoie la représentation sous forme de liste du réseau à n individus n'ayant aucun lien d'amitié déclaré entre eux.
3. Écrire une fonction `estUnLienEntre(paire, i, j)` où `paire` est une liste à deux éléments et `i` et `j` sont deux entiers, et qui renvoie `True` si les deux éléments contenus dans `paire` sont `i` et `j` dans un ordre quelconque ; et renvoie `False` sinon.
4. Écrire une fonction `sontAmis(reseau, i, j)` qui renvoie `True` s'il existe un lien d'amitié entre les individus `i` et `j` dans le réseau `réseau`, et `False` sinon.

   Quelle est la complexité en temps de votre fonction dans le pire des cas en fonction du nombre m de liens d'amitié déclarés dans le réseau ?
5. Écrire une fonction `declareAmis(reseau, i, j)` qui modifie le réseau `reseau` pour y ajouter un lien d'amitié entre les individus i et j *si ce lien n'y figure pas déjà.*

   Quelle est la complexité de votre fonction dans le pire des cas en fonction du nombre m de liens d'amitié déclarés dans le réseau ?
6. Écrire une fonction `listeDesAmisDe(reseau, i)` qui renvoie la liste des amis de `i` dans le réseau `reseau`.

   Quelle est la complexité de votre fonction dans le pire des cas en fonction du nombre m de liens d'amitié déclarés dans le réseau ?

On dit qu'un individu i est un ami d'ordre n d'un individu j si le plus court chemin reliant i et j est composé de n liens d'amitié. Par exemple, dans le réseau présenté sur le figure 1, l'individu `3` est un ami d'ordre `1` de l'individu `0` et l'individu `5` est un ami d'ordre `2` de l'individu `0`.

7. Écrire une fonction `listeDesAmisDeOrdre(reseau, i, n)` qui renvoie la liste des amis d'ordre `n` de `i` dans le réseau `reseau`.
   Quelle est la complexité de votre fonction dans le pire des cas en fonction du nombre m de liens d'amitié déclarés dans le réseau ?

Une représentation simplifiée, réduite à deux tables, de la base de données d'un réseau social est donnée dans la figure 2.

INDIVIDUS

id	nom	prenom
1	Potter	Harry
2	Granger	Hermione
⋮	⋮	⋮

LIENS

id1	id2
1	2
2	1
⋮	⋮

FIGURE 2 – Schéma de la base de données du réseau social

La table `INDIVIDUS` répertorie les individus et contient les colonnes :

— `id` (clé primaire), un entier identifiant chaque individu ;

— `nom`, une chaîne de caractères donnant le nom de famille de l'individu ;

— `prenom`, une chaîne de caractères donnant le prénom de l'individu.

La table `LIENS` répertorie les liens d'amitié entre individus et contient les colonnes :

— `id1`, entier identifiant le premier individu du lien d'amitié ;

— `id2`, entier identifiant le second individu du lien d'amitié.

On supposera par ailleurs que pour tout couple `(x,y)` dans la table `LIENS`, le couple `(y,x)` est également présent dans la table (contrairement à la partie précédente de l'énoncé).

8. Écrire une requête SQL qui renvoie les identifiants des amis de l'individu d'identifiant `x`.
9. Écrire une requête SQL qui renvoie les (noms,prénoms) des amis de l'individu d'identifiant `x`.
10. Écrire une requête SQL qui renvoie les identifiants des individus qui sont amis avec au moins un ami de l'individu d'identifiant `x`. (On pourra éventuellement faire une jointure de la table `LIENS` avec elle-même.)