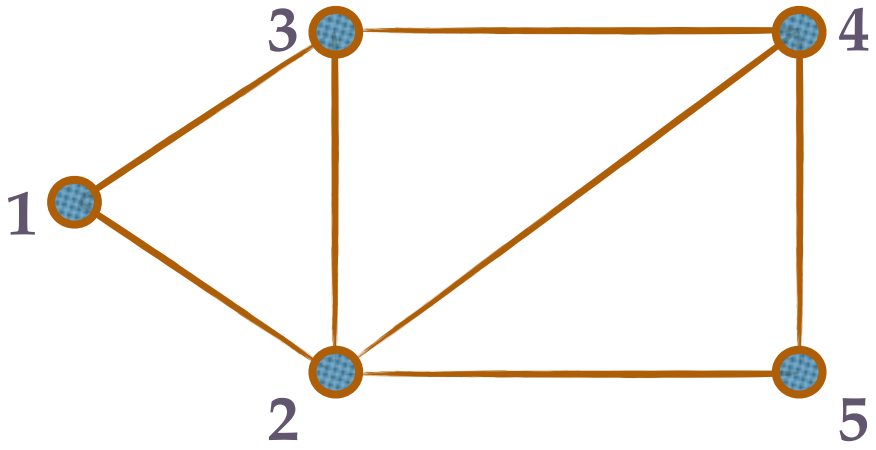
INTRODUCTION AUX GRAPHES

On se propose ici de se familiariser avec la notion de graphe en jouant avec différentes représentations possibles pour un même graphe simple sans distance ni directivité.
Le graphe étudié pour exemple est le suivant : soit un graphe de 5 noeuds et de 7 cotés !



I PRÉPARATION : LES DIFFÉRENTES REPRÉSENTATIONS

Lire la partie de cours consacrée aux graphes et en particulier aux différentes représentations possibles (à la fin), puis donner les représentations de ce graphe sous les 3 formes classiques :
- Liste des cotés
- Matrice d'adjacences
- Dictionnaire { noeud : (noeuds connectés) }

II VERSION LISTE DE COTÉS

On suppose dans cette partie que le graphe est donné par sa liste de cotés.
Attention : les noeuds sont numérotés mais le graphe étant dynamique on n'a aucune garantie sur l'indexation : une collection de noeuds [2, 5, 7, 19] est donc possible.

a - Ecrire un « bout de code » déterminant le nombre de noeuds du graphe. Quelle en est la complexité ? Rq : En toute rigueur, on s'interdit ici d'utiliser la commande IN qui suppose la notion d'ensemble et masque la complexité réelle du programme.

b - Ecrire un « bout de code » qui détermine le noeud de connectivité maximale.
Quelle en est la complexité ?

c - Ecrire une fonction **supNoeudCotes**(listeCotes, n) qui renvoie la liste des cotés, une fois le noeud n supprimé du graphe. Quelle en est la complexité ? Quelle conséquence sur la connexité du graphe ? Trouver un exemple de graphe à N noeuds pour lequel le résultat de la suppression d'un noeud est une liste vide.

d - Ecrire une fonction **addNoeudCotes**(graphCotes, n, connec) qui renvoie la liste de cotés une fois ajouté un noeud n et ses connections (connec : liste de noeuds à lier au noeud n). La fonction ne doit pas rajouter de connection déjà établie (attention les 2 sens (p,q) et (q,p) sont équivalents).

III VERSION MATRICE D'ADJACENCES

On suppose dans cette partie que le graphe est donné par sa matrice d'adjacences, soit une liste de listes ne contenant que des 0 ou des 1. Le numéro d'un noeud désignant à la fois une colonne ou une ligne.

a - Comment obtenir simplement le nombre N de noeuds du graphe. Quelle en est la complexité ? Pouvez-vous vraiment connaitre les indices des noeuds ? Comment faire ? Un graphe avec une collection de noeuds [2, 5, 7, 19] est il possible ?

b - Ecrire un « bout de code » qui détermine le noeud de connectivité maximale.
On donne pour cela la commande **sum** qui agit sur une liste : **sum**([1,2,3]) renvoie 6.
Quelle est la complexité réelle de ce bout de code (on déterminera celle de sum dans le code) ?

c - Ecrire une fonction **supNoeudMatrice**(matrice, p) qui renvoie la matrice une fois le noeud d'indice p et toutes ses connections supprimés du graphe. Notez bien que le noeud étant identifié par sa colonne, vous ne pouvez pas supprimer de colonne ou de ligne : quel serait le problème ?

d - Ecrire une fonction **addNoeudMatrice**(matrice, n, connec) qui renvoie la matrice avec le noeud n et ses connections en n-1 ème colonne. Noter que si $n \leqslant N$ la taille de la matrice est inchangée mais si $n > N$ il faut rajouter autant de ligne et colonne que nécessaire.
Rq : les connections passées dans **connec** doivent être des noeuds déjà existants.

IV VERSION LISTE D'ADJACENCES

On suppose dans cette partie que le graphe est donné par un dictionnaire { noeud : [connec] }. Relire si nécessaire le cours sur les dictionnaires.

a - Comment obtenir simplement le nombre N de noeuds du graphe. Complexité ?
b - Ecrire un « bout de code » qui détermine le noeud de connectivité maximale. Complexité ?

c - Ecrire une fonction **supNoeudDico**(dico, p) qui renvoie le dictionnaire une fois le noeud p et toutes ses connections supprimées. On donne pour cela la méthode **.pop**(k) qui fonctionne sur les listes et les dictionnaires en renvoyant l'élément d'indice k (ou de clef k pour un dictionnaire), l'élément est alors extrait de la liste ou du dictionnaire.

d - Ecrire une fonction **addNoeudDico**(dico, p, connec) qui renvoie le dictionnaire une fois le noeud et ses connections introduites. On utilisera pour cela la commande **update** qui agit sur les dictionnaires (cf cours.). Penser bien qu'il faut ajouter le noeud p, mais également modifier les connections des autres noeuds qui seraient reliés à p.

V °°° GRAPHE ÉTOILÉ °°°

Pour chacun des 3 types de structures de données précédentes proposer une suite d'instructions simples pour se ramener à un graphe étoilé autour du noeud 2 et avec un 6ème noeud.
RQ : noter qu'on ne sait pas enlever un coté seul à vous d'adapter votre stratégie.
Afficher la structure de données. Puis supprimer le noeud 2 et afficher à nouveau. Conclure.
Les stratégies sont-elles différentes d'un structure de données à l'autre ? Et les complexités ?

FIN