- TD 3. Parcours -

- Exercice 1 - Promenade dans le Petersen.

On note G le graphe obtenu en supprimant une arête ab du graphe de Petersen. Effectuer un parcours en largeur et un parcours en profondeur du graphe G en partant du sommet a (pour le parcours en profondeur, en proposer un dont l'arbre associé n'est pas un chemin). Pour chaque parcours, on précisera l'arbre associé et pour chaque sommet v, on donnera :

a. L'ordre et le niveau de v pour le parcours en largeur.

b. Les dates de début et de fin, $d[v]$ et $f[v]$, pour le parcours en profondeur, ainsi que la liste des intervalles de prsence dans la pile.

- Exercice 2 - Pile.

On a effectué un parcours en profondeur dans un graphe et la suite des opérations empiler (e) et dépiler (d) sur la pile AT a été : eeedeededddeeededdddd. Quel est l'arbre associé ?

- Exercice 3 - Labyrinthe.

Un labyrinthe est constitué d'un ensemble de salles reliées par des couloirs. Il possède deux salles particulières : la salle de départ et la salle contenant la porte de sortie.

a. *Question préliminaire :* écrire un algorithme MARCHE-EXHAUSTIVE(G, r) qui, prenant en entrée un graphe connexe G et un sommet r, retourne une marche pour laquelle chaque arête du graphe est parcourue dans les deux sens une seule fois. Pour cela, on pourra utiliser et compléter un des pseudo-codes de la fiche d'algo.

b. Pour sortir d'un labyrinthe, on adopte la stratégie (c-à-d l'ordre de découverte) donnée par un parcours en largeur. Pour k quelconque, proposer un labyrinthe avec k couloirs pour lequel la stratégie adoptée oblige à traverser $\Omega(k^2)$ couloirs.

c. Cela peut-il arriver avec un parcours en profondeur ? Justifier.

- Exercice 4 - Orientation fortement connexe.

Un graphe orienté $D = (V, A)$ est *fortement connexe* si pour tout $x, y \in V$, il existe un chemin orienté de x à y. Dans un graphe connexe non orienté G, un *pont* est une arête e de G telle que $G - e$ n'est pas connexe.

a. Proposer une orientation fortement connexe du cube.

b. Montrer que si un graphe non orienté G admet une orientation fortement connexe, alors G est connexe et sans pont.

c. Inversement, montrer que si G est connexe et sans pont, il admet une orientation fortement connexe. On pourra utiliser un arbre en profondeur de G.

- Exercice 5 - Vrai/Faux.

Confirmer (et prouver) ou infirmer (et donner un contre-exemple) les propriétés suivantes :

a. Tous les arbres sont bipartis.

b. Si un arbre T est différent d'un chemin alors tout ordre de parcours en largeur de T depuis une racine r est différent de l'ordre de début d'un parcours en profondeur de T depuis r.

c. Il est possible d'écrire une version du parcours en profondeur en une fonction de 5 lignes et un appel principal de 2 lignes. On ne fournira que la fonction *père* et on n'écrira qu'une instruction par ligne (!).

- Exercice 6 - Arêtes séparatrices.

Soit $G = (V, E)$ un graphe connexe, une arête e de G telle que $G - e$ n'est pas connexe est dite *arête séparatrice* (ou *pont*) de G.

a. Montrer qu'un arbre couvrant de G contient toutes les arêtes séparatrices de G.

b. Proposer un algorithme qui calcule toutes les arêtes séparatrices de G.

c. Essayer de trouver un tel algorithme avec un temps d'exécution en $O(m)$, où m est le nombre d'arêtes de G. Pour cela, on effectuera un parcours en profondeur de G dans lequel on calculera la fonction $bas[v]$, égale au minimum de $début[v]$ et de $début[w]$ pour toutes les arêtes uw avec u descendant de v et w ancêtre de v.

- Exercice 7 - Sommets séparateurs.

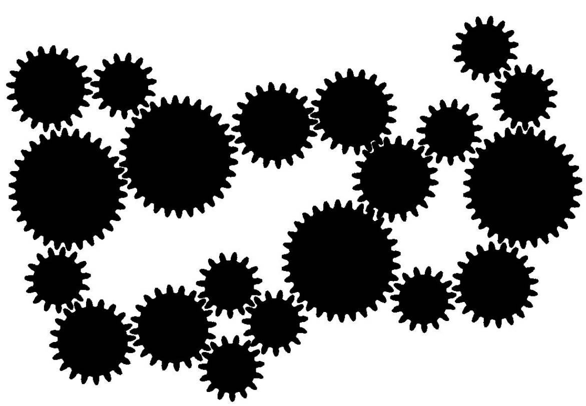
Soit G un graphe connexe. Un sommet v est *séparateur* si $G \setminus v$ n'est pas connexe. Soit r un sommet de G et T un arbre en profondeur de G de racine r.

a. Montrer que r est un sommet séparateur si et seulement si il possède au moins deux fils dans T.

b. Montrer qu'un sommet $x \neq r$ est séparateur si et seulement si x possède un fils dont aucun descendant n'a pour voisin dans G un ancêtre strict de x.

c. Ecrire un algorithme $SEPARATEURS(G)$ qui retourne l'ensemble des sommets séparateurs de G et prend un temps $O(m)$.

- Exercice 8 - Engrenages.



Ça tourne ?
Proposer un modèle et un algorithme pour résoudre le cas général.