Fiche d'exercices pratiques 3 - 6h

Consignes

Les consignes de la première fiche de TP restent valides.
Attention : vous devez déposer votre travail à la fin des deux séances consacrées à cette fiche sur `Moodle` https://moodle.uphf.fr/course/view.php?id=1681

1. Le travail en binôme est autorisé.
2. N'oubliez pas d'indiquer vos nom(s) et prénom(s) en commentaire en haut de votre (vos) fichier(s).
3. Ne déposez que les fichiers sources (code) de votre travail.
4. Ne négligez pas les commentaires pour expliquer vos choix et les fonctions mises en œuvre.

Le respect des consignes sera pris en compte dans la notation.
La version déposée à la fin de la séance sera celle évaluée. Néanmoins une version améliorée / complétée pourra être déposée au plus tard une semaine après la séance, et pourra être prise en compte sous la forme d'un *bonus* dans la notation.

Exercice 1 : inversion d'un graphe

Dans cet exercice vous devez considérer la représentation par listes d'adjacence d'un graphe (cf. fiche 1).

Écrire un algorithme permettant de construire le graphe *inverse* d'un graphe orienté donné. On entend par graphe inverse un graphe dans lequel tous les arcs sont inversés.

Ainsi par exemple le graphe `1` $\rightarrow$ `2` $\rightarrow$ `3` doit être converti en `1` $\leftarrow$ `2` $\leftarrow$ `3`.
Vérifier le bon fonctionnement de l'algorithme sur des graphes générés aléatoirement avec la fonction de l'exercice 2 de la fiche 1.

Exercice 2 : parcours

Dans cet exercice on suppose qu'un nouveau langage vient d'être découvert via un ensemble de mots triés. On cherche à savoir quel est le bon ordre entre les lettres dans cette langue inconnue.

Par exemple, à partir de l'ensemble de mots `XWW - WXYZ - WXYW - YWX - YWZ` la méthode doit retourner `X - Z - W - Y`.

Nous allons utiliser un graphe pour répondre à ce problème. Sur l'exemple on peut observer qu'à partir des deux derniers mots, `YWX` et `YWZ`, la lettre `X` doit venir avant la lettre `Z`, puisque les deux premières lettres des deux mots sont identiques. On peut en fait procéder de façon similaire en comparant toutes les paires de mots qui se suivent dans le tableau afin de découvrir toutes les règles qui organisent le langage. Le principe est alors de stocker ces règles dans un graphe dans lequel un sommet est associé à chaque lettre, et les arcs représentent le fait qu'une lettre doit être suivie par les autres.

Dans notre exemple cela donne `W` $\rightarrow$ `Y`, `X` $\rightarrow$ `W, Z`, `Z` $\rightarrow$ `W`, alors que `Y` n'a pas de successeurs.
Pour pouvoir combiner l'ensemble des règles entre elles et ainsi déduire l'ordre des lettres il reste à faire un tri topologique du graphe. Celui-ci peut être réalisé via un parcours en profondeur.

Implémenter les algorithmes nécessaires pour résoudre ce problème, en représentant le graphe via sa matrice d'adjacence ou ses listes d'adjacence, au choix. Le tableau de mots sera fourni via un fichier texte contenant sur la première ligne le nombre de mots, suivi de chaque mot (1 par ligne).
Votre algorithme doit retourner l'ordre des lettres obtenu.