Université Polytechnique Hauts-de-France
Institut National des Sciences Appliquées Hauts-de-France **2021-2022**

FISA Informatique 1 Graphes et Algorithmes

Fiche d'exercices pratiques 1 - 3h

Consignes

L'objectif de cette première fiche est d'implémenter les fonctionnalités de base pour manipuler un graphe orienté **ou** non orienté.
On considère ici uniquement des graphes simples, sans boucles.
Le langage de programmation est libre mais il **n'est pas autorisé** d'utiliser une bibliothèque de manipulation de graphes.
La mémoire doit être gérée au mieux (on n'utilisera par exemple pas une matrice de taille 500 × 500 pour un graphe contenant 10 sommets).
Attention : vous devez déposer votre travail à la fin de la séance 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.
Notez toutefois que les fonctionnalités de l'exercice 1 de ce TP seront nécessaires pour les TP suivants. Il est donc fondamental d'avoir une version fonctionnelle dès que possible.

Conseil : lisez l'intégralité de l'énoncé avant de foncer tête baissée.

Exercice 1 : représentation par listes d'adjacence

On associe à un graphe les éléments de base suivants : un type (orienté ou non orienté) ; le nombre `n` de sommets ; le nombre `m` de connexions (arcs ou arêtes) ; un tableau de listes permettant de stocker les sommets successeurs ou adjacents de chaque sommet.
Les sommets sont caractérisés par un identifiant unique géré, pour simplifier, par un entier. **Attention**, rien n'oblige à ce que les identifiants soient les entiers de 1 à n (ni de 0 à n-1).

Après avoir défini une structure de données appelée `Listes` (en ajoutant éventuellement d'autres données que vous jugez utiles), vous écrivez des fonctions pour :

- Créer un graphe pour un type donné et un nombre de sommets donné. La fonction initialisera le graphe **sans** connexion (pas d'arcs / arêtes) mais devra saisir (et stocker) les identifiants des sommets.
- Libérer l'espace mémoire associé à un graphe donné (selon le langage utilisé).
- Ajouter une connexion dans un graphe entre deux sommets connus par leurs identifiants.
- Supprimer une connexion dans un graphe entre deux sommets connus par leurs identifiants.
- Ajouter un sommet dont l'identifiant est passé en paramètre dans un graphe. Ce sommet est initialement sans connexion. La fonction n'ajoutera pas le sommet si l'identifiant existe déjà.
- Savoir si un sommet connu par son identifiant est un sommet adjacent **direct** d'un autre sommet donné.
- Afficher un graphe. On se contentera ici d'afficher le type du graphe, le nombre de sommets puis les listes d'adjacence de chaque sommet.
- Charger un graphe à partir d'un fichier **texte** dont le nom physique est passé en argument et respectant **scrupuleusement** le format suivant : la première ligne contient uniquement le type du graphe (0 pour

non orienté, 1 pour orienté), le nombre de sommets (`n`) et le nombre de connexions (`m`). Puis on a une ligne pour chaque connexion ($\rightarrow$ `m` lignes consécutives), contenant les deux identifiants des extrémités de l'arc ou de l'arête, séparés par un espace. Ainsi par exemple on pourrait avoir un fichier intitulé "mon_graphe.txt" et contenant :

```
1 7 6
1 2
1 3
3 4
5 1
5 6
4 7
```

— Sauvegarder un graphe dans un fichier texte dont le nom physique est passé en argument, et en respectant **scrupuleusement** le format de la question précédente.

Exercice 2 : génération de graphes

Dans cet exercice on cherche à implémenter un algorithme de génération aléatoire de graphes. Différentes techniques existent dans la littérature. Parmi celles-ci nous utiliserons une des deux méthodes proposées par Erdös-Rényi, celle notée $G(n, p)$. Le principe peut être résumé comme suit :

— Choisir une probabilité p ;

— Créer n sommets ;

— Chaque paire (x, y) de sommets est connectée de façon indépendante selon la probabilité p.

Cette méthode peut être implémentée de façon assez simple. Le paramètre p permet d'impacter sur la densité du graphe : plus p est proche de 1 et plus le graphe est dense (avec les cas extrêmes $p = 0$ et $p = 1$ correspondant à un graphe sans connexion et un graphe complet, respectivement).

1. Implémenter une version de cette approche pour n et p donnés. Le graphe généré est sauvegardé dans un fichier texte respectant le format utilisé dans la l'exercice 1.
2. Vérifier son bon fonctionnement sur des graphes de petite taille, orientés et non orientés, et en jouant avec le paramètre p.