## Université d'Aix-Marseille - Master Informatique $1^{ere}$ année
## UE Complexité - TD 2 - Rappels d'algorithmique de base sur les graphes

**Préambule :** Dans cette planche, nous abordons la manipulation de graphes en utilisant les deux représentations classiques, i.e. par matrices et par listes d'adjacence. Certaines questions seront à traiter avec les deux représentations (l'enseignant de TD vous le précisera). Dans tous les cas, il faudra fournir une évaluation de la complexité. Voici un rappel des types C utilisés (matrices et listes) pour le cas de graphes simples (i.e. pas des multigraphes) sans valuation :

```
#define NMAX .../* nombre maximum de sommets */

typedef int SOMMET; /* indice des sommets */

/* representation par matrices d'adjacence */
typedef struct {    int A[NMAX][NMAX]; /* matrice carree 0/1 */;
                    int n; /* valeur comprise entre 0 et NMAX */
} GRAPHEMAT;

/* representation par listes d'adjacence avec en premier definition des listes */
typedef struct maillon {  SOMMET st;
                          struct maillon *suivant;
} CHAINON;
typedef CHAINON *PTR_CHAINON;

typedef struct {    PTR_CHAINON A[NMAX]; /* tableau de listes */;
                    int n; /* valeur comprise entre 0 et NMAX */
} GRAPHELIST;
```

Cette "implémentation" en C des structures de données est proposée à titre indicatif et pour rappeler précisément le cadre. Mais comme il s'agit ensuite d'écrire des algorithmes, il va de soi qu'il n'est pas obligatoire de les reprendre dans les exercices.

### Exercice 1. Algorithmes de base sur les graphes

Il est proposé ici d'écrire plusieurs algorithmes de graphes. Il est suggéré de les écrire pour chacune des représentations (matrice et listes).

**Question 1.** Test d'existence d'un arc (graphes orientés). Donnez un algorithme qui prend en entrées un graphe orienté $G$ et deux sommets $x$ et $y$ et qui vérifie si l'arc $(x,y)$ est présent dans le graphe $G$. Donnez une évaluation de sa complexité.

**Question 2.** Calcul des successeurs d'un sommet (graphes orientés). Donnez un algorithme qui prend en entrées un graphe orienté $G$ et un sommets $x$ et qui calcule l'ensemble $E$ des sommets successeurs de $x$ dans le graphe $G$. Dans une première version de l'algorithme, vous utiliserez une représentation d'ensembles de sommets par tableau de booléns, et dans une seconde, vous utiliserez une représentation d'ensembles de sommets par liste simplement chaînée. Donnez une évaluation de la complexité pour chaque version de l'algorithme.

**Question 3.** Calcul des prédécesseurs d'un sommet (graphes orientés). Donnez un algorithme qui prend en entrées un graphe orienté $G$ et un sommets $x$ et qui calcule l'ensemble $E$ des sommets prédécesseurs de $x$ dans le graphe $G$. Dans une première version de l'algorithme, vous utiliserez une représentation d'ensembles de sommets par tableau de booléns, et dans une seconde, vous utiliserez une représentation d'ensembles de sommets par liste simplement chaînée. Donnez une évaluation de la complexité pour chaque version de l'algorithme.

**Question 4.** Graphe réciproque (graphes orientés). Donnez un algorithme qui prend en entrée un graphe orienté $G = (S,A)$ et calcule son graphe réciproque $G^{-1} = (S,A^{-1})$. Donnez sa complexité.

**Question 5.** Symétrisation (graphes orientés). Donnez un algorithme qui prend en entrée un graphe orienté $G = (S,A)$ et calcule son graphe symétrisé $G_{Sym} = (S,A \cup A^{-1})$. Donnez une évaluation de sa complexité. Pour le cas des listes d'adjacence, il existe un algorithme linéaire... Il faudrait le trouver, sachant qu'il est interdit de construire un multigraphe, c'est-à-dire, un graphe avec la duplication d'arcs.

**Question 6.** Complémentaire (graphes non-orientés). Donnez un algorithme qui prend en entrée un graphe non-orienté $G = (S,A)$ et calcule son graphe complémentaire $\overline{G}$ (une arête est présente dans $\overline{G}$ si elle ne figure pas dans $G$ et elle est absente dans $\overline{G}$ si elle se trouve dans $G$). On ne considère pas les arêtes réflexives, i.e. reliant un sommet à lui-même. Donnez sa complexité.

**Question 7.** Test de clique (graphes non-orientés). Donnez un algorithme qui prend en entrées un graphe non-orienté $G = (S,A)$ et un sous-ensemble $K$ de ses sommets, et vérifie si cet ensemble $K$ est une clique du graphe $G$ (une *clique* est un ensemble de sommets mutuellement voisins). Donnez une évaluation de sa complexité.

**Question 8.** Test de clique maximale (graphes non-orientés). Donnez un algorithme qui prend en entrées un graphe non-orienté $G = (S,A)$ et un sous-ensemble $K$ de ses sommets, et vérifie si cet ensemble $K$ est une clique maximale du graphe $G$. On dit qu'une clique de $G$ est *maximale* si elle n'est incluse strictement dans aucune autre clique de $G$. Donnez une évaluation de sa complexité.

**Question 9.** Clique maximum (graphes non-orientés). Que faudrait-il faire pour pour s'assurer qu'étant donné un graphe non-orienté $G = (S,A)$ et un sous-ensemble $K$ de ses sommets, $K$ est une clique maximum du graphe $G$ (i.e. il n'existe aucune autre clique de $G$ qui contienne strictement plus de sommets que $K$)? Attention, il n'est pas demandé ici de donner un algorithme.

**Question 10.** Existence d'un circuit (graphes orientés). Donnez un algorithme qui prend en entrée un graphe orienté $G = (S,A)$ et calcule un circuit de $G$ s'il en existe un.

## Exercice 2. Problème du transversal

Un *transversal* dans un graphe non-orienté $G = (S,A)$, est un sous-ensemble $T$ de $S$ tel que pour toute arête $\{x,y\} \in A$, alors $x \in T$ ou $y \in T$ (*ou* non exclusif). En d'autres termes, un transversal est un sous-ensemble de sommets partageant au moins un sommet avec chaque arête du graphe. Seule la représentation de graphes par matrices d'adjaence sera considérée ici.

**Question 1.** Donnez un algorithme qui prend en entrées un graphe non-orienté $G$ et un ensemble de sommets $T$, et qui vérifie si $T$ est un transversal de $G$. Donnez une évaluation de sa complexité.

**Question 2.** Donnez un algorithme qui prend en entrées un graphe non-orienté $G$ et un ensemble de sommets $T$, et qui vérifie si $T$ est un transversal minimal de $G$, c'est-à-dire que $T$ ne possède aucun sous-ensemble strict qui soit aussi un transversal. Donnez une évaluation de sa complexité.

**Question 3.** Donnez un algorithme qui prend en entrée un graphe non-orienté $G$ et qui fournit en résultat un transversal minimal de $G$. Donnez une évaluation de sa complexité.

**Question 4.** Donnez un algorithme qui prend en entrées un graphe non-orienté $G$ et un entier $k$, et teste si le graphe $G$ possède un transversal de taille $k$ ou moins. Donnez sa complexité.