Algorithmie Avancée

TP/TD Représentation par matrice d'adjacences

A. Document Algo_TDParE_Birmele.pdf

Vous traiterez les exercices **1.1** et **1.2 en reprenant des exercices si besoin venant des activités supplémentaires du TP précédent (réflexion sur les arbres, connexité, nombre d'arêtes, Thorème de Cayley…)**

B. Structure de données : la matrice d'adjacence

Vous implémenterez une procédure permettant de visiter les nœuds d'un graphe en C à partir d'un sommet de référence en utilisant la structure de donnée dite de matrice d'adjacence (***int **adjacence***) pour représenter des graphes. Vous créerez une fonction main de test qui permette ce charger un graphe représenté par une telle matrice (une fonction ***chargeGraphe()*** est à définir dans votre fichier ***graphe.c*** à partir de *scanf* à partir du *stdin* ou d'un fichier texte de poids 0 ou 1 dont la première ligne indiquera l'ordre du graphe par exemple). Vous afficherez également l'ordre de marquage ou chemin de visite.

```
// Procédure qui marque tous les sommets par ordre de voisinage depuis un sommet de
// référence
//        Paramètres :
//        adjacence : matrice d'adjacence du graphe
//        ordre : nombre de sommets
//        s : numéro du sommet de référence

void  marquerVoisins (int** adjacence, int ordre, int s) {
        // Variables locales
        int *marques ; // tableau dynamique indiquant si les sommets sont marqués ou non
        int x, y ; // numéros de sommets intermédiaires

        // Allouer le tableau marques de taille « ordre »
        …

        // Initialiser les marquages à 0
        for (x=0 ; x<ordre ; x++)
                marques[x] = 0 ;

        // Marquer le sommet s à 1
        marques[s] = 1 ;

        // Pour tous les sommets x marqués
        // Marquer les sommets non marqués y adjacents à x
        for (x=0 ; x<ordre ; x++)
                if (marques[x])
                        for (y=0 ; y<ordre ; y++)
                                if (adjacence[x][y] && !marques[y])
                                        marques[y] = 1;
}
```

Est-ce que cet algorithme marque tous les sommets ?
Prévoir une fonction qui affiche tous les sommets marqués.
Si non, modifier le programme pour le cas où après une passe de la fonction marquerVoisins() tous les sommets ne sont pas marqués. Quelle est la complexité calcul de cet algorithme ? Peut-on faire mieux ? Quelle est la complexité mémoire ?
S'il vous reste du temps, vous étendrez cette fonction à une version linéarisée de la matrice d'adjacence (int * avec ordrexordre indices comme vu en cours).