Université d'Aix-Marseille - Master Informatique 1^{ere} année

UE Complexité - TP1 - 3 Mini-projets

Préambule : Dans ce premier travail à réaliser, 3 exercices (des "mini-projets") assez simples sont à traiter, de préférence dans l'ordre (sauf indication contraire de vos enseignants). Le langage de programmation recommandé est le langage C mais vous n'y êtes pas astreints. Pour chaque mini-projet, vous devrez bien sûr rédiger un petit compte rendu qui décrira notamment les algorithmes implémentés et leur complexité et fournira tout élément nécessaire à la prise en main des programmes correspondants.

Mini-projet 1. Calcul des nombres de Fibonacci : le point de vue expérimental

Il est proposé ici d'implémenter les 3 versions du calcul des nombres de Fibonacci :

- Version de base récursive ;
- Version de base itérative ;
- Version basée sur l'exponentiation de matrice.

Pour la troisième version, contrairement aux exercices 2 et 3 de la planche de TD 1, ici, lorsqu'il sera question d'exponentiation, il ne pourra être fait d'hypothèse sur la valeur de l'exposant, à savoir que nous ne supposerons pas que l'exposant est une puissance de 2 (cf. quand on supposait que pour un exposant p, il existe un entier k tel que $p = 2^k$). Et pour le cas où p n'est pas une puissance de 2, pour un calcul efficace de x^p, il faut utiliser l'approche suivante :

- si p est pair, alors $x^p = (x^2)^{(p/2)}$
- si p est impair, alors $x^p = x \times x^{(p-1)}$

Il est demandé d'implémenter les 3 versions et de les expérimenter de sorte à comparer l'efficacité relative de ces 3 approches, notamment pour savoir jusqu'à quelle valeur de n il est possible d'aller pour calculer f_n en un temps raisonnable. Pour le cas où vous auriez besoin de manipuler de très grands nombres, il est suggéré d'utiliser des flottants de grande taille.

Mini-projet 2. Manipulation de graphes non-orientés

Il est demandé ici d'implémenter quelques fonctionnalités sur des graphes non-orientés. Nous allons nous intéresser à la notion de "*zone dense*" d'un graphe. On dira, pour un graphe non-orienté $G = (S,A)$, qu'un sous-ensemble X de S est une *zone dense* de G si dans le sous-graphe de G induit par X, i.e. dans $G(X)$, tous les sommets sont relíes deux à deux par des arêtes. Pour la représentation des graphes, vous pourrez utiliser au choix une représentation matricielle ou par listes d'adjacence. Par ailleurs, bien que s'agissant d'un TP, il vous faudra préciser la complexité des différents algorithmes que vous aurez implémenté. Par ailleurs, vous devrez fournir également des résultats expérimentaux pour tester l'efficacité des solutions conçues, en termes de qualité des solutions, mais aussi, pour évaluer l'efficacité pratique en termes de temps de calcul.

Question 1. Test de vérification. Vous devez concevoir un traitement qui prend en entrées un graphe non-orienté $G = (S,A)$ et un sous-ensemble X de S et vérifie si X est une zone dense de G.

Question 2. Calcul de zone dense maximale. Vous devez concevoir un traitement qui prend en entrée un graphe non-orienté $G = (S,A)$ et calcule un sous-ensemble X de S qui est une zone dense maximale de G, c'est-à-dire que X n'est pas un sous-ensemble strict d'un sous-ensemble Y de S qui serait aussi une zone dense de G.

Question 3. Calcul de zone dense maximum (méthode "complète"). Vous devez concevoir un traitement qui prend en entrée un graphe non-orienté $G = (S,A)$ et calcule un sous-ensemble X de S qui est une zone dense maximum de G, c'est-à-dire que dans G il n'existe pas de zone dense Y de G qui soit de plus grande taille que X (i.e. telle que l'on aurait $|Y| > |X|$).

Question 4. Calcul de zone dense maximum (méthode "incomplète"). Pour le cas où le traitement que vous avez conçu pour la question 3 se révèlerait assez inefficace en pratique (ce devrait être le cas), concevez une traitement qui ne garantit pas l'optimalité du résultat calculé (aucune garantie d'avoir la plus grande zone dense en résultat) mais qui d'une part est plus efficace en pratique (en termes de temps d'exécution) que la solution proposée dans la question 3, et propose des solutions en général de plus grande taille que celles calculées avec le traitement implémenté pour la question 2.

Mini-projet 3. Simulation d'une Machine de Turing Déterministe.

Il est demandé ici d'implémenter un programme simulant le calcul réalisé par un programme pour Machine de Turing Déterministe. Votre programme devra donc considérer deux entrées possibles :

- d'une part, le programme M devant être simulé ;
- d'autre part, la donnée en entrée de ce programme M, soit un mot devant être traité par M.

Vous pourriez, pour vous amuser, concevoir d'une sortie graphique du fonctionnement de la Machine de Turing (visualisation du ruban, de la tête de lecture écriture, etc.), mais ce n'est bien sûr pas demandé dans le cadre d'un mini-projet.