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

UE Complexité - Examen partiel - 2023-2024

Durée : 1h30 - Tous documents interdits

Préambule : Pour les questions où vous devez fournir des algorithmes, ceux-ci devront être écrits dans un langage algorithmique avec des instructions de genre "`si alors sinon`", "`tantque`", "`pour`" etc. mais il faudra les définir sans d'ambiguïté de sorte à ce que le correcteur puisse s'assurer de leur validité. Ainsi, n'utilisez pas les particularités et les spécificités que peuvent avoir certains langages de programmation (par exemple, les boucles "`pour`" sont exprimées de façons différentes en C et en Python ; pour ce type de boucle, il faut préciser exactement les conditions initiales et celles d'arrêt, car sans cela, il est parfois impossible de déterminer la validité d'un algorithme). Quand vous utilisez des boucles, celles-ci doivent impérativement être écrites sur une seule page. Toute structure de données introduite dans vos algorithmes doit auparavant être explicitée. Il faudra aussi définir avec un maximum de précision vos algorithmes de sorte à ce que l'évaluation de leur complexité soit la plus précise possible (ces évaluations devront toujours être justifiées). Concernant celles-ci, sauf indication contraire, nous utiliserons le modèle de base pour lequel toutes les actions élémentaires, commme les opérations du type tests, affectations ou opérations arithmétiques, sont exécutables en temps constant. Et quand la notation Θ est possible, il faut la privilégier par rapport à la notation O.

Exercice 1 (2 points). On considère ici des tableaux de n caractères (avec $n \geq 0$), qui ne peuvent contenir que les caractères "X", "Y" ou "B" (blanc), sachant que les caractères "X" ou "Y" ne peuvent apparaître que dans le début du tableau, et dès que le caractère "B" apparaît, toutes les cases suivantes contiennent le caractère "B" jusqu'à la fin du tableau. Il vous faut donner ici un algorithme qui prend en entrée un tel tableau et qui vérifie si ce tableau contient une séquence de caractères qui débute par "XY" et termine par "YX" (il peut bien sûr y avoir des "B" après). Donnez une évaluation de la complexité de l'algorithme.

Exercice 2 (4 points). Définissez un programme pour Machine de Turing Déterministe qui reconnaît le langage L défini sur l'alphabet {0,1} et constitué des mots qui contiennent comme préfixe 01 et comme suffixe 10, c'est-à-dire les mots qui commencent par 01 et terminent par 10. On rappelle que pour définir un programme pour Machine de Turing Déterministe, il faut notamment préciser ses états, son alphabet d'entrée ainsi que son alphabet de ruban, et la fonction de transition (vous pouvez fournir celle-ci par un dessin). Donnez une évaluation de la complexité de votre programme. À quelle classe de complexité appartient ce langage L ? Justifiez votre réponse.

Exercice 3 (14 points + 2 "points bonus"). On considère ici seulement des graphes non-orientés et sans boucle (donc sans arête reliant un sommet à lui-même). Un *ensemble forêt* dans un graphe non-orienté $G = (S,A)$ est un sous-ensemble F de S, i.e. $F \subseteq S$ tel que le sous-graphe de G induit par les sommets appartenant à F, i.e. le graphe $G[F]$ est une forêt. On rappelle qu'une forêt est un graphe non-orienté ne contenant aucun cycle (oui... une forêt est un ensemble d'arbres !). À partir de cette notion d'ensemble forêt, il est possible de définir le problème de décision suivant :

> **ENSEMBLE FORÊT**
> **Instance** : Un graphe non-orienté $G = (S,A)$ et un entier k.
> **Question** : G possède-t-il un ensemble forêt de taille k ou plus ?

Question 1 (1 point). Dessinez un graphe non-orienté G de 9 sommets et 14 arêtes tel que le couple $(G,6)$ (i.e. $k = 6$) est une instance positive de *ENSEMBLE FORÊT* et le couple $(G,7)$ (i.e. $k = 7$) est une instance négative de *ENSEMBLE FORÊT*. Dans chaque cas, il vous faut justifier votre réponse.

Question 2 (2 points). À partir du problème de décision *ENSEMBLE FORÊT*, définissez le problème de **recherche** associé, puis celui d'**optimisation** (ici on considère bien sûr comme critère d'optimisation la taille de l'ensemble forêt), celui de **comptage** et enfin, celui d'**énumération**.

Dans la suite de cet exercice, on supposera que vous disposez d'une fonction appelée `FORET` qui prend en entrée un graphe non-orienté G et a pour résultat `Vrai` si ce graphe est sans cycle (on dit qu'il est *acyclique*) et `Faux` sinon. Vous ne devez pas écrire l'algorithme associé à cette fonction (sauf si vous traitez la question "bonus" en fin d'exercice). Vous supposerez de plus que sa complexité est $T_{FORET}(n+m)$ pour le cas d'un graphe de n sommets et m arêtes en entrée (on supposera que cette complexité est polynomiale).

Question 3 (2 points). Donnez un algorithme qui prend en entrées un graphe non-orienté G et un ensemble de sommets F représenté par un tableau de booléens, et qui vérifie si F est un ensemble forêt de G. Donnez une évaluation de la complexité de votre algorithme. Nous vous imposons d'**utiliser à partir de cette question, une représentation des graphes par matrices d'adjacence** même si cela peut engendrer une (petite) dégradation de l'efficacité de vos algorithmes.

Question 4 (3 points). Donnez un algorithme qui prend en entrées un graphe non-orienté G et un ensemble de sommets F, et qui vérifie si F est un ensemble forêt maximal de G, c'est-à-dire qu'il n'existe aucun ensemble de sommets F' incluant strictement F (i.e. $F \subsetneq F'$) et qui soit aussi un ensemble forêt de G. Donnez une évaluation de la complexité de votre algorithme.

Question 5 (2 points). Donnez un algorithme qui prend en entrée un graphe non-orienté G et qui fournit en résultat un ensemble forêt de G. Donnez une évaluation de la complexité de votre algorithme.

Question 6 (3 points). Donnez un schéma d'algorithme (il n'est nécessaire de donner votre algorithme dans le détail) qui prend en entrées un graphe non-orienté G et un entier k, et teste si le graphe G possède un ensemble forêt de taille k ou plus. Donnez une évaluation de la complexité de votre algorithme.

Question 7 (1 point). Pouvez-vous affirmer que le problème de décision *ENSEMBLE FORÊT* appartient à la classe de complexité **P** ? Justifiez votre réponse.

Question "Bonus" (2 points). Donnez un algorithme appelé `FORET` qui prend en entrée un graphe non-orienté G et vérifie si ce graphe est sans cycle. Une méthode simple pour cela est basée sur la notion d'*effeuillage d'arbre*. Au sens de la Théorie des Graphes, un *arbre* est un graphe non orienté connexe et sans cycle, et un graphe sans cycle est une *forêt*, c'est-à-dire, une collection d'arbres (chacune de ses composantes connexes est un arbre). Aussi, pour savoir si un graphe est sans cycle, il suffit de supprimer les sommets de degré 1 ou 0, et de poursuivre ce traitement tant que c'est possible. S'il reste des sommets, cela signifie que leur degré est au moins égal a 2 et qu'en conséquence, ils figurent tous dans un cycle au moins. Par contre, si tous les sommets ont été supprimés, c'est qu'il n'existait pas de cycle initialement. Donnez une évaluation de la complexité de votre algorithme.