université de BORDEAUX	**Année Universitaire 2016/2017, Session 1 d'automne** Master CSI, UE 4TCY701U, Théorie de la complexité. Jeudi 8 décembre 2016. 9h–12h Durée : 3h. **Documents autorisés** : 3 feuilles A4 manuscrites recto-verso.	**Collège Sciences et Technologies**

La notation attachera une grande importance à la clarté et à la concision des justifications.

Le barème est indicatif.

Exercice 1 — Applications du cours (3 points). Répondez aux questions suivantes en **justifiant** vos réponses. Toute réponse non justifiée vaut 0 points.

1) Si ψ est une formule propositionnelle, x l'une de ses variables, et $\mathtt{b}$ un booléen ($\mathtt{true}$ ou $\mathtt{false}$), on note $\psi[x \mapsto \mathtt{b}]$ la formule obtenue à partir de ψ en remplaçant toutes les occurrences de x par $\mathtt{b}$. Par exemple, si $\psi = \neg\mathtt{false} \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge \neg(x_2 \vee x_1)$ alors $\psi[x_2 \mapsto \mathtt{true}] = \neg\mathtt{false} \wedge (x_1 \vee \mathtt{true} \vee \neg x_3) \wedge \neg(\mathtt{true} \vee x_1)$.
On considère les deux problèmes suivants :
   (a) Sat : satisfaisabilité des formules booléennes avec variables, par ex. $\varphi = (x_1 \vee \mathtt{true} \vee \neg x_3) \wedge \neg(\mathtt{true} \vee x_1)$.
   (b) Eval : évaluation des formules booléennes sans variables, par ex. $\varphi' = \mathtt{false} \vee \mathtt{true} \wedge \neg(\mathtt{false} \vee \mathtt{true})$.
   On réduit Sat à Eval en utilisant une fonction calculable f définie récursivement comme suit :

$$f(\varphi) = g_n(\varphi) \qquad \text{où les variables de } \varphi \text{ sont } \{x_1, \ldots, x_n\}, \text{ et}$$
$$g_i(\varphi) = \begin{cases} \varphi & \text{si } i = 0 \\ g_{i-1}(\varphi[x_i \mapsto \mathtt{true}]) \ \vee \ g_{i-1}(\varphi[x_i \mapsto \mathtt{false}]) & \text{si } i > 0. \end{cases}$$

   En particulier, g_i élimine la variable x_i. La réduction f prouve-t-elle que Eval est **NP**-difficile ?

2) L'existence d'un problème **NP**-complet dont le complémentaire est dans **NP** entraîne-t-elle **NP** = co-**NP** ?

3) Est-il vrai que **P** = **NP** si et seulement si tout problème non-trivial dans **NP** est **NP**-difficile ? ∎

Exercice 2 — Configurations répétées (3 points). Dans cet exercice, on utilise des machines de Turing déterministes à plusieurs bandes.

1) À partir d'une telle machine de Turing M, construire une autre machine de Turing déterministe M' avec les 2 propriétés suivantes :
   a) Pour tout mot d'entrée x, M accepte x si et seulement si M' accepte x.
   b) Pour tout mot d'entrée x, la suite de configurations de M' sur l'entrée x ne contient pas 2 fois la même configuration.

   Rappel. Une configuration contient l'état courant, le contenu de chacune des bandes et la position de chaque tête de la machine.

2) En utilisant la question 1), déterminer si le problème suivant est décidable ou non.
   ENTRÉE : Le code d'une machine de Turing déterministe M.
   QUESTION : À partir de la configuration initiale sur le mot vide, le calcul de M contient-il au moins 2 fois la même configuration ? ∎

Exercice 3 — NP-Complétude (5 points). Montrez que chacun des problèmes suivants est **NP**-complet. N'oubliez pas que vous devez donc prouver pour chaque problème (1) qu'il est dans **NP** *et* (2) qu'il est **NP**-difficile.

Remarque : toutes les réductions sont faciles, la principale difficulté est de trouver le bon problème à réduire. Pour cela, vous pouvez utiliser tout problème vu en cours ou en TD (mais seulement ceux-là).

1) Sac à dos

ENTRÉE : Deux suites d'entiers $v_1, \dots, v_n \in \mathbb{N}$ et $p_1, \dots, p_n \in \mathbb{N}$ de la même longueur n, et deux entiers $V, P \in \mathbb{N}$.
QUESTION : Existe-t-il un ensemble $I \subseteq \{1, \dots, n\}$ tel que $\sum_{i \in I} p_i \leqslant P$ et $\sum_{i \in I} v_i \geqslant V$.

2) Arbre de poids fixé

ENTRÉE : Un entier $P \in \mathbb{N}$, un graphe $G = (V, E)$ non orienté, et une fonction $p : E \to \mathbb{N}$.
QUESTION : Existe-t-il un sous-ensemble $F \subseteq E$ des arêtes de G tel que (V, F) est un arbre (c'est-à-dire un graphe connexe sans cycle) et $\sum_{e \in F} p(e) = P$?

3) Dix pour cent

ENTRÉE : Des entiers $n_1, n_2, \dots, n_k$.
QUESTION : Existe-t-il un ensemble d'indices $I \subseteq \{1, \dots, n\}$ tel que $\sum_{i \in I} n_i = \frac{1}{10} \sum_{i \notin I} n_i$?

4) Recouvrement par sous-ensembles

ENTRÉE : Un ensemble fini A, des sous-ensembles $B_1, \dots, B_n$ de A, et un entier $k \in \mathbb{N}$.
QUESTION : Existe-t-il un ensemble $I \subseteq \{1, \dots, n\}$ tel que I a k éléments, et $A = \bigcup_{i \in I} B_i$? ∎

Exercice 4 — Machines reset (6 points). Dans cet exercice, toutes les machines de Turing considérées fonctionnent sur une seule bande, infinie à droite seulement (les cases sont numérotées par les entiers naturels, la case la plus à gauche par 0). L'entrée des machines est écrite à partir de la case 0, et initialement la tête est sur la case 0.

Une *machine reset* est une machine de Turing à une bande lecture-écriture, infinie à droite, potentiellement non déterministe, et disposant de 2 types d'instructions :

- Type 1 : la machine remplace le symbole sous la tête, change d'état, et déplace la tête d'une case vers la droite.
- Type 2 : la machine remplace le symbole sous la tête, change d'état, et repositionne la tête sur la case 0 (c'est-à-dire, sur la case la plus à gauche de la bande).

Par contre, la machine n'a pas d'instruction pour déplacer la tête d'une case vers la gauche. Une telle machine peut être *non déterministe*. Avec les notations vues en cours, on a donc une fonction de transition $\delta : Q \times \Gamma \to \mathscr{P}(Q \times \Gamma \times \{\triangleright, \lhd\lhd\})$, où $\triangleright$ désigne le déplacement de la tête d'une case à droite et $\lhd\lhd$ le repositionnement de la tête en début de bande (et où, si E est un ensemble, $\mathscr{P}(E)$ désigne l'ensemble des sous-ensembles de E).

1) Écrire une machine reset effectuant la multiplication par 2 du nombre binaire donné en entrée.
2) Écrire une machine reset effectuant la division par 2 du nombre binaire donné en entrée.
3) On se donne maintenant une machine de Turing habituelle M, à une bande infinie à droite : M a des instructions déplaçant la tête de lecture d'une case vers la gauche, mais pas d'instruction permettant de ramener directement la tête de lecture en position 0. Construire une machine *reset* M_{reset} qui simule M, c'est-à-dire telle que pour tout mot d'entrée x, M accepte x si et seulement si M_{reset} accepte x.

Indication. Il s'agit d'expliquer comment simuler une instruction de M qui déplace la tête d'une case à gauche. Vous pouvez ici vous aider du non-déterminisme. Vous pouvez aussi marquer des cases, par exemple en écrivant un couple $(b, \#)$ là où la machine d'origine écrit b. Vous pouvez enfin utiliser plusieurs types de marques.

4) Le problème suivant est-il décidable ? Justifier précisément la réponse.

ENTRÉE : Le code d'une machine reset.
QUESTION : Cette machine reset s'arrête-t-elle sur le mot vide ?

5) Peut-on simuler une machine reset par une machine de Turing habituelle ? Justifier la réponse.

On considère maintenant les machines reset <u>déterministes</u>. Une telle machine est donnée par une fonction de transition $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\triangleright, \lhd\lhd\}$.

6) Écrire une machine reset <u>déterministe</u> effectuant la division par 2 du nombre binaire donné en entrée.

7) Expliquer comment simuler une machine de Turing à une bande habituelle par une machine reset déterministe.

8) Montrer que si un problème est dans la classe **P**, il peut être décidé par une machine reset déterministe effectuant un nombre polynomial de pas de calcul par rapport à la taille de l'entrée. ∎

Exercice 5 — Sous-mots communs (4 points). On dit qu'un mot u est un *sous-mot* d'un mot w si u est obtenu à partir de w en effaçant des lettres. Par exemple, *aba* est un sous mot de *bccadabaac*, mais *aba* n'est pas un sous-mot de *bacdddabbbbcbddcb*.
Formellement, $u = a_1 \cdots a_n$ est un *sous-mot* de $w = b_1 \cdots b_m$ s'il existe n positions dans le mot w, disons $1 \leqslant i_1 < i_2 < \cdots < i_n \leqslant m$, telles que pour tout j, on a $a_j = b_{i_j}$.

On définit le problème Sous-mot commun suivant :

ENTRÉE : Un alphabet fini A, une liste de mots $u_1, \ldots, u_n$ sur l'alphabet A, et un entier k.
QUESTION : Existe-t-il un mot de longueur k qui est sous-mot de chacun des mots $u_1, \ldots, u_n$?

On veut montrer que ce problème est **NP**-complet.

1) Montrer que ce problème est dans la classe **NP**.

Pour montrer que le problème est **NP**-complet, on propose la réduction suivante, à partir du problème Couverture par sommets dont on rappelle la définition.

ENTRÉE : Un graphe non-orienté $G = (V, E)$ et un entier p.
QUESTION : Existe-t-il un sous-ensemble $C \subseteq V$ de sommets tel que $|C| = p$ et tel que toute arête a au moins une de ses extrémités dans C ?

À partir d'une entrée (G, p) du problème de Couverture par sommets, on construit l'instance suivante du problème Sous-mot commun.

— On choisit comme alphabet l'ensemble des sommets de G : $A = V = \{s_1, \ldots, s_\ell\}$ (où ℓ est le nombre de sommets de G).

— On construit les $|E| + 1$ mots suivants :

$$u = s_1 \cdots s_\ell,$$
$$u_e = \underbrace{s_1 \cdots s_{i-1} s_{i+1} \cdots s_\ell}_{\text{tous les sommets sauf } s_i} \cdot \underbrace{s_1 \cdots s_{j-1} s_{j+1} \cdots s_\ell}_{\text{tous les sommets sauf } s_j} \text{ pour chaque arête } e \text{ entre } s_i \text{ et } s_j, \text{ avec } i < j.$$

— On choisit enfin $k = \ell - p$.

2) Montrer que cette transformation définit bien une réduction polynomiale de Couverture par sommets à Sous-mot commun. ∎