| université de BORDEAUX | **Année Universitaire 2019/2020, Session 1 d'automne**<br>Master CSI, UE 4TCY701U, Théorie de la complexité.<br>Lundi 18 novembre 2019. 9h–12h Durée : 3h.<br>**Documents non autorisés.** | **Collège Sciences et Technologies** |
|---|---|---|

**Toute réponse non justifiée n'apportera pas de point. Le barème est indicatif.**
**Vous pouvez utiliser une question précédente même si elle n'a pas été traitée.**

## Exercice 1 : Questions d'application du cours — 3 Points

Répondre aux questions ci-dessous en **justifiant brièvement mais précisément** (seules les réponses correctement justifiées apportent un point) :

**1)** Le problème **Pas d'impair** suivant est-il décidable ?
- **Entrée** : Le code d'une machine de Turing $M$.
- **Question** : Est-il vrai que $M$ n'accepte aucun mot de longueur impaire ?

**2)** Le problème **Arrêt borné** suivant est-il décidable ?
- **Entrée** : Un entier $k$ et le code d'une machine de Turing $M$.
- **Question** : Est-il vrai que sur toute entrée, $M$ s'arrête en au plus $k$ étapes de calcul ?

**3)** Est-il vrai que tout problème décidable se réduit au problème de correspondance de Post ?

## Exercice 2 : P vs. NP-complet — 5 Points

Dans cet exercice, on suppose que $\mathbf{P} \neq \mathbf{NP}$. Pour chacun des problèmes suivants, dites si il est dans **P** ou **NP**-complet. **Attention** : les réponses doivent être **prouvées**. Si vous répondez qu'un problème est dans **P**, vous devez décrire en français un algorithme polynomial qui le résout et expliquer pourquoi il est polynomial. Si vous répondez qu'un problème est **NP**-complet, vous devez prouver qu'il est dans **NP** *et* qu'il est **NP**-difficile. Pour cela, vous pouvez utiliser n'importe quel problème **NP**-complet parmi ceux vus en cours ou présents dans une feuille de TD, et seulement ceux-ci.

***Remarque** : les réductions sont faciles, la principale difficulté est de trouver le bon problème à réduire.*

Dans l'exercice, les graphes sont non orientés. Un ***arbre*** est un graphe connexe (c'est-à-dire que pour tous sommets $s, t$, il existe un chemin de $s$ à $t$) et sans cycle (c'est-à-dire qu'il n'y a pas de suite d'au moins 2 arêtes distinctes deux à deux de la forme $s_1 - s_2 - \cdots - s_n - s_1$). Enfin, le ***degré*** d'un sommet est le nombre d'arêtes adjacentes à ce sommet.

**1) Arbre couvrant de degré borné**
- **Entrée** : Un entier $k$ et un graphe $G = (S, A)$. Son ensemble de sommets est $S$, son ensemble d'arêtes est $A$.
- **Question** : Existe-t-il $B \subseteq A$ tel que $(S, B)$ soit un arbre dans lequel chaque sommet a un degré au plus $k$ ?

**2) Grand ensemble indépendant**
- **Entrée** : Un graphe $G = (S, A)$ à $n \geqslant 43$ sommets (ensemble de sommets : $S$, et ensemble d'arêtes : $A$).
- **Question** : Existe-t-il un sous-ensemble $S'$ de $S$ contenant $n - 42$ sommets, tel qu'aucune arête de $A$ n'a ses deux extrémités dans $S'$ ?

**3) SAT modifié**
- **Entrée** : Une formule propositionnelle en forme DNF (disjonction de conjonctions de littéraux).
- **Question** : Existe-t-il une affectation des variables qui rend la formule vraie et une autre affectation des variables qui rend la formule fausse ?

**4) Système d'équations quadratiques**
- **Entrée** : Un ensemble de $n$ variables $x_1, \ldots, x_n$ et un système fini d'équations, chacune de la forme $\sum_{1 \leqslant i,j \leqslant n} a_{i,j} x_i x_j + \sum_{1 \leqslant k \leqslant n} b_k x_k + c = 0$ où les constantes $a_{i,j}, b_k, c$ valent 0 ou 1.
- **Question** : Le système a-t-il une solution dans $\mathbb{Z}/2\mathbb{Z}$ ?

**5) Sac à dos**
- **Entrée** : Des entiers naturels $p_1, \ldots, p_n$ et $v_1, \ldots, v_n$, ainsi que deux entiers naturels $P$ et $V$.
- **Question** : Existe-t-il un sous-ensemble $I$ de $\{1, \ldots, n\}$ tel que $\sum_{i \in I} p_i \leqslant P$ et $\sum_{i \in I} v_i \geqslant V$ ?

## Exercice 3 : Inversion de marche — 5 Points

À une machine de Turing déterministe $M$ et un mot d'entrée $w$ pour $M$, on associe un nombre $n \in \mathbb{N} \cup \{\infty\}$. Intuitivement, $n$ compte le nombre de fois où la machine change la direction dans laquelle sa tête de lecture est déplacée (on parle d'inversion de marche).

Pour toute étape $i$ du calcul de $M$ sur $w$, on note $D_i \in \{\triangleleft, \triangleright, \triangledown\}$, la direction dans laquelle la tête est déplacée lors de cette étape. On dit qu'une ***inversion de marche*** se produit à l'étape $i$ si il existe une étape précédente $j < i$ telle que les deux conditions suivantes sont satisfaites :

— pour toute étape intermédiaire $k$ (*i.e.*, telle que $j < k < i$), on a $D_k = \triangledown$ et,

— soit $D_j = \triangleleft$ et $D_i = \triangleright$, soit $D_j = \triangleright$ et $D_i = \triangleleft$.

Le ***nombre d'inversions de marche*** de l'exécution de $M$ sur $w$ est le nombre $n \in \mathbb{N} \cup \{\infty\}$ d'étapes auxquelles une inversion de marche se produit lors du calcul de $M$ sur $w$. On remarquera que ce nombre est potentiellement infini (s'il est infini, l'exécution de $M$ sur $w$ ne termine pas).

On considère trois problèmes de décision. Pour chacun d'entre eux, dire si celui-ci est décidable ou indécidable. Dans chaque cas, la réponse devra être **prouvée**.

**1)** **NON-INVERSION**

**Entrée** : Le code d'une machine de Turing $M$ et un mot d'entrée $w$.
**Question** : Est-ce que le nombre d'inversions de marche de l'exécution de $M$ sur $w$ est égal à 0 ?

**2)** **BORNE FIXÉE**

**Entrée** : Le code d'une machine de Turing $M$, un mot d'entrée $w$ et un entier $k \in \mathbb{N}$.
**Question** : Est-ce que le nombre d'inversions de marche de l'exécution de $M$ sur $w$ est inférieur à $k$ ?

**3)** **INVERSION BORNÉE**

**Entrée** : Le code d'une machine de Turing $M$ et un mot d'entrée $w$.
**Question** : Est-ce qu'il existe $k \in \mathbb{N}$ tel que le nombre d'inversions de marche de l'exécution de $M$ sur $w$ est inférieur à $k$ ?

## Exercice 4 : Grammaires — 10 Points

Dans cet exercice, on considère plusieurs variations du formalisme de *grammaires*. Une grammaire $G$ permet de définir un langage de mots $L(G)$. Le premier formalisme est celui des grammaires hors-contexte (utilisé, en particulier, pour définir la syntaxe des langages de programmation).

### Partie I : Grammaires hors-contexte.

Une ***grammaire hors-contexte*** $G$ est donnée par 4 composantes, $G = (\Sigma, \mathcal{V}, S, \mathcal{R})$ où :

— $\Sigma$ est un ***alphabet*** (la grammaire définit un langage $L(G) \subseteq \Sigma^*$).

— $\mathcal{V} = \{X, Y, Z, \dots, \}$ est un ensemble fini de ***variables***.

— $S \in \mathcal{V}$ est une variable, appelée ***variable de départ***.

— $\mathcal{R}$ est un ensemble de ***règles*** de réécriture. Chaque règle s'écrit de la façon suivante :

$$X \to w \qquad \text{avec } X \in \mathcal{V} \text{ et } w \in (\mathcal{V} \cup \Sigma)^*.$$

Par exemple $G = (\{a, b\}, \{X\}, X, \mathcal{R})$ avec $\mathcal{R} = \{X \to ab,\ X \to aXb\}$ est une grammaire hors-contexte à deux règles.

À toute grammaire hors-contexte $G = (\Sigma, \mathcal{V}, S, \mathcal{R})$, on associe un langage $L(G) \subseteq \Sigma^*$ de la façon suivante. Soient $u, v$ deux mots dans $(\Sigma \cup \mathcal{V})^*$ (leurs lettres sont soient des variables de $\mathcal{V}$ soit des lettres de $\Sigma$). On note :

$$u \to v$$

si il existe une règle de réécriture $X \to w$ dans $\mathcal{R}$, et $u_1, u_2 \in (\Sigma \cup \mathcal{V})^*$ tels que :

$$u = u_1 X u_2 \quad \text{et} \quad v = u_1 w u_2.$$

En d'autres termes, on obtient $v$ à partir de $u$ en remplaçant la variable $X$ par le mot $w$ (*i.e.*, en appliquant la règle $X \to w$). On note $\xrightarrow{*}$ la clôture transitive de $\to$, c'est-à-dire que $u \xrightarrow{*} v$ si et seulement si il existe $u_0, \dots, u_n$ tels que,

$$u = u_0 \to u_1 \to u_2 \to \cdots \to u_n = v.$$

Par exemple, pour la grammaire $G$ ci-dessus, on a $X \to aXb \to aabb$, donc $X \xrightarrow{*} aabb$.

On peut maintenant introduire le langage défini par la grammaire $G$ :

$$L(G) = \{v \in \Sigma^* \mid S \xrightarrow{*} v\}.$$

Remarquez que $L(G)$ est un langage de mots de $\Sigma^*$, c'est-à-dire qu'il n'y a aucune lettre de $\mathcal{V}$ dans les mots de $L(G)$. Un langage de la forme $L(G)$ où $G$ est une grammaire hors-contexte s'appelle un ***langage hors-contexte***.

**1)** On considère la grammaire $G = (\{a, b\}, \{X\}, X, \mathcal{R})$ où $\mathcal{R} = \{X \to ab, X \to aXb\}$. Quel est le langage défini par $G$ ?

**2)** Montrer que tout langage hors-contexte est décidable.

## Partie II : Problèmes de décision pour grammaires hors-contexte.

Dans cette partie, on considère plusieurs problèmes de décision pour les grammaires hors-contexte.

On considère le problème **ACCEPTATION D'UN MOT** suivant.

**Entrée** : Une grammaire hors-contexte $G = (\Sigma, \mathcal{V}, S, \mathcal{R})$ sans règle de la forme $X \to \varepsilon$ et un mot $w \in \Sigma^*$.
**Question** : A-t-on $w \in L(G)$ ?

**1)** Montrer que le problème **ACCEPTATION D'UN MOT** est décidable. Quelle est la plus petite classe de complexité en temps vue en cours contenant ce problème ?

On considère maintenant le problème **LANGAGE VIDE** suivant.

**Entrée** : Une grammaire hors-contexte $G = (\Sigma, \mathcal{V}, S, \mathcal{R})$.
**Question** : A-t-on $L(G) = \emptyset$ ?

**2)** Montrer que le problème **LANGAGE VIDE** est dans **P**. Quelle est la plus petite classe de complexité en espace que vous connaissez contenant ce problème ?

**3)** On considère le problème d'intersection non vide pour les langages hors-contexte :

**Entrée** : Deux grammaires hors-contexte $G, H$.
**Question** : A-t-on $L(G) \cap L(H) \neq \emptyset$ ?

Montrer que ce problème est indécidable.

**Indication**

Réduisez le problème de correspondance de Post à ce problème : étant donnée une instance de PCP, construisez deux langages hors-contexte $L_1, L_2$ tels que $L_1 \cap L_2 \neq \emptyset$ si et seulement si l'instance du PCP a une solution.

**4)** (Difficile) On considère le problème **UNIVERSEL** suivant pour les langages hors-contexte :

**Entrée** : Une grammaire hors-contexte $G = (\Sigma, \mathcal{V}, S, \mathcal{R})$.
**Question** : A-t-on $L(G) = \Sigma^*$ ?

Montrer que ce problème est indécidable.

## Partie III : Grammaires hors-contexte simples.

Une grammaire hors-contexte est ***simple*** si ses règles sont de la forme $X \to a\,Y$ ou $X \to \varepsilon$ (où $X, Y \in \mathcal{V}$ et $a \in \Sigma$).

**1)** Montrer que le problème **UNIVERSEL** dont on restreint l'entrée à une grammaire *simple* est décidable.

On dit qu'une machine de Turing de décision est ***unidirectionnelle*** si elle déplace toujours sa tête de lecture vers la droite et si elle s'arrête dès qu'elle lit le caractère "□" (c'est-à-dire dès qu'elle a fini de lire son entrée). Formellement, une machine (potentiellement non-déterministe) $M = (\Sigma, \Gamma, Q, q_i, q_a, q_r, \delta)$ est ***unidirectionnelle*** si son ensemble de transitions $\delta \subseteq (Q \setminus \{q_a, q_r\}) \times \Gamma \times Q \times \Gamma \times \{\triangleleft, \triangleright, \nabla\}$ satisfait ces deux conditions :

— Pour toute transition $(q, a, q', b, D) \in \delta$, on a $D = \triangleright$.
— Pour toute transition $(q, a, q', b, D)$, si $a = \square$, alors $q' = q_a$ ou $q' = q_r$.

**2)** Montrer que si $G$ est simple, alors $L(G)$ est décidé par une machine unidirectionnelle.

**Indication**

Vous pouvez coder chaque règle de la grammaire par une transition d'une machine unidirectionnelle.

**3)** Inversement, montrer que si $M$ est une machine unidirectionnelle, $L(M)$ est défini par une grammaire simple.

Étant donné un mot $w \in \Sigma^*$, on note **alph**$(w) \subseteq \Sigma$ l'ensemble des lettres de $\Sigma$ apparaissant dans $w$. Par exemple, si $\Sigma = \{a, b, c, d\}$, **alph**$(a\,a\,b\,b\,a\,d\,d\,b\,a) = \{a, b, d\}$. On considère le problème de décision suivant :

**Entrée** : Un alphabet $\Sigma$ et deux grammaires hors-contexte simples $G_1$ et $G_2$, d'alphabet commun $\Sigma$.
**Question** : Existe-t-il $w_1 \in L(G_1)$ et $w_2 \in L(G_2)$ tels que **alph**$(w_1) =$ **alph**$(w_2)$ ?

**4)** Montrer que ce problème est dans **NP** et qu'il est **NP**-difficile.

## Partie IV : Grammaires générales.

On considère maintenant des grammaires plus générales. La seule différence avec les grammaires hors-contexte est que les règles sont de la forme $u \to v$, avec toujours $v \in (\Sigma \cup \mathcal{V})^*$ (comme dans les grammaires hors-contexte), mais le membre gauche $u$ est un *mot non vide* sur l'alphabet $\mathcal{V}$. Dans le cas des grammaires hors-contexte, $u$ devait être une lettre de l'alphabet $\mathcal{V}$.

**1)** Montrer que le problème **LANGAGE VIDE** pour ces grammaires générales est indécidable.