Collège Sciences et technologies – Masters
Année universitaire 2016/2017, session d'automne décembre 2016
Mention : Informatique, Code UE : 4TIN704EX
Intitulé de l'épreuve : *Calculabilité et complexité*
Date : 8/12/16 Heure : 9h-12h Durée : 3h
Documents autorisés : poly et notes TD
Le barème est indicatif. A noter qu'il dépasse 20 points.

On attachera une grande importance à la clarté et à la concision des justifications. Vous devez rendre la première page du sujet avec votre copie.

Pour les exercices 1 et 2 seulement les justifications écrites dans l'encadré sur le sujet serons considérées. Pour l'exercice 3 (QCM) seulement les réponses écrites sur le sujet seront considérées.

Exercice 1 (2 points + 2 points) : ensembles dénombrables

Un *polynôme* est une fonction $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$ avec une inconnue x et des coefficients $a_n, \ldots, a_0$ **entiers**. Un entier x s'appelle *racine* du polynôme p si $p(x) = 0$.

1. (1 p.) Justifiez que l'ensemble $\mathcal{P}$ des polynômes est dénombrable.

   Correction : un polynôme $p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$ peut être identifié avec les tuplet de ses coefficients $(a_n, \ldots, a_0)$. Or on a montré en cours/TD que l'ensemble des séquences *finies* d'entiers est dénombrable.

2. (1 p) Justifiez que l'ensemble $\mathcal{R}$ des polynômes qui ont au moins une racine entière, est dénombrable.

   Correction : L'ensemble $\mathcal{R}$ est dénombrable car il est sous-ensemble de l'ensemble dénombrable $\mathcal{P}$.

3. (**2 p. additionnels**)
   Décrivez un **algorithme** qui énumère l'ensemble $\mathcal{R}$.
   Indication : Vous pouvez utiliser un sous-algorithme qui énumère $\mathcal{P} \times \mathbb{Z}$.

   Correction : On sait que $\mathcal{P}$ et $\mathbb{Z}$ sont dénombrables, donc $\mathcal{P} \times \mathbb{Z}$ est dénombrable. On considère un algorithme qui énumère $\mathcal{P} \times \mathbb{Z}$. Pour chaque couple $(p(x), n)$ on calcule $p(n)$. Si le résultat est 0, on ajoute p à la sortie.

Exercice 2 (3 points) : décidabilité

Une **2-réduction** de A vers B est un programme P qui travaille sur des entrées de A et qui s'arrête toujours ; sur entrée x le programme P calcule deux entrées y_1 et y_2 de B, avec la propriété suivante :

x est entrée positive de A si et seulement si
y_1 est entrée positive de B **et** y_2 est entrée négative de B

Justifiez : si B est décidable et s'il existe une 2-réduction de A vers B, alors A est décidable.

Correction : On suppose que B est décidable, donc il existe un algorithme P' (= programme qui termine toujours) qui décide sur entrée y si y est entrée positive de B.

Pour décider A on utilise l'algorithme suivant : sur entrée x on lance le programme P pour calculer y_1 et y_2. Ensuite, on lance P' d'abord sur y_1, ensuite sur y_2. Comme P' termine toujours, on aura les réponses aux deux questions $y_1 \stackrel{?}{\in} B$ et $y_2 \stackrel{?}{\in} B$. Si les deux réponses sont "oui", on répond "oui" pour x, sinon on répond "non".

Exercice 3 (2 points) : QCM

Répondez par "Vrai" ou "Faux" aux questions suivantes. Les réponses fausses sont comptées négativement.

		Vrai	Faux
1	L'ensemble des suites infinies d'entiers est dénombrable.	□	X
2	Tout ensemble d'entiers qui est dénombrable peut être énuméré en ordre croissant.	□	X
3	Tout programme en C peut s'écrire en langage WHILE.	X	□
4	Tout programme en C peut s'écrire en langage LOOP.	□	X
5	Il existe un algorithme qui prend en entrée un programme en Java et répond "oui" si le programme renvoie des données personnelles non-chiffrées, et "non" sinon.	□	X
6	Il existe un semi-algorithme qui prend en entrée un programme en Java et répond "oui" si le programme renvoie des données personnelles non-chiffrées.	X	□
7	Toute formule booléenne satisfaisable possède un certificat polynomial de satisfiabilité.	X	□
8	NP est la classes des problèmes qui ne peuvent pas être résolus en temps polynomial.	□	X

Exercice 4 (5 points) : NP

Dans cet exercice nous considérons la variante suivante du problème 3-colorabilité, appellé **3-COL-restreint** :

Entrée : Un graphe non-orienté $G = (V, E)$ et un entier k.
Question : existe-t-il un sous-ensemble de sommets $U \subseteq V$ de taille k tel que $G \setminus U$ est 3-colorable ?

Remarque : $G \setminus U$ est le graphe obtenu à partir de G en enlevant tous les sommets de U, ainsi que les arêtes incidentes.

1. Montrez que **3-COL-restreint** est dans NP.
   Quels sont les certificats ? comment marche la vérification ?

   Correction :
   — Un certificat pour **3-COL-restreint** est une paire (U, c), où $U \subseteq V$ est un ensemble de sommets et $c : (V \setminus U) \to \{1, 2, 3\}$ est une 3-coloration des sommets de $G \setminus U$.
   La taille du certificat est polynomiale (en $O(|V|)$. Le vérificateur prend une paire (U, c) en entrée et vérifie en temps polynomial si
   — U est de taille k (en temps $O(|V|)$), et
   — pour tous $(u, v) \in E$, $u, v \notin U$: $(c(u) \neq c(v)$ (en temps $O(|E|)$.
2. Montrez que **3-COL-restreint** est NP-difficile.

   Correction : on peut réduire 3-COL à 3-COL-restreint, en envoyant l'entrée G de 3-COL sur l'entrée $G, 0$ de 3-COL-restreint.
3. Décrivez une réduction polynomiale de **3-COL-restreint** vers **SAT**.
   Quelle est la signification des variables booléennes ? qu'expriment vos clauses ?

   Indication : en plus des variables utilisées dans la réduction pour 3-colorabilité, vous allez utiliser $n \cdot k$ variables additionnelles, une pour chaque paire sommet / entier dans $\{1, \ldots, k\}$.

   Correction : on utilise deux types de variables, pour décrire l'ensemble U ainsi que la coloration.
   Pour la coloration on se sert des variables vue en cours : $x_{u,i}$, pour chaque $u \in V$ et $i \in \{1, 2, 3\}$. Pour l'ensemble U on se sert de variables $y_{v,j}$, où $v \in V$, $j \in \{1, \ldots, k\}$.
   L'idée sous-jacente est que $x_{u,i}$ vrai signifie que la couleur de u est i, et $y_{v,j}$ vrai signifie que v est le j-ème sommet de U.
   Pour les clauses : on modifie les clauses de 3-COL pour tenir compte de U, et on rajoute des clauses pour U.
   — Clauses de 3-COL sur les variables $x_{u,i}$:
   (a) On remplace les clauses vues en cours pour 3-COL (exprimant le fait que chaque sommet a exactement une couleur associée)

$$\bigwedge_{u \in V} \big((x_{u,1} \vee x_{u,2} \vee x_{u,3}) \wedge \bigwedge_{1 \leq i \neq j \leq 3} (\neg x_{u,i} \vee \neg x_{u,j})\big)$$

par

$$\Phi_1 = \bigwedge_{u \in V} \big(\wedge_{1 \leq j \leq k} \neg y_{u,j} \Longrightarrow \phi \big)$$

où $\phi = (x_{u,1} \vee x_{u,2} \vee x_{u,3}) \wedge \bigwedge_{1 \leq i \neq j \leq 3} (\neg x_{u,i} \vee \neg x_{u,j})$. Cette partie dit que si $u \notin U$, alors u doit avoir exactement une couleur parmi $\{1, 2, 3\}$.

(b) On remplace les clauses vues en cours pour 3-COL (exprimant que deux sommets voisins ont toujours des couleurs différentes)

$$\bigwedge_{uv \in E, 1 \leq i \leq 3} (\neg x_{u,i} \vee \neg x_{v,i})$$

par

$$\Phi_2 = \bigwedge_{uv \in E, 1 \leq i \leq 3} \big(\wedge_{1 \leq j \leq k} (\neg y_{u,j} \wedge \neg y_{v,j}) \big) \Longrightarrow (\neg x_{u,i} \vee \neg x_{v,i})$$

Cette partie dit que si $u, v \notin U$ sont voisins, alors leurs couleurs sont différentes.

— Clauses pour U :

$$\Phi_3 = \bigwedge_{v \in V, 1 \leq i \neq j \leq k} (\neg y_{v,i} \vee \neg y_{v,j}) \wedge \bigwedge_{1 \leq i \leq k} \bigvee_{v \in V} y_{v,i}$$

Un sommet ne peut pas être affecté à deux "positions" différentes dans U, et à chaque "position" de U on a affecté un sommet.

La formule obtenue est $\Phi = \Phi_1 \wedge \Phi_2 \wedge \Phi_3$. Les formules Φ_1, Ψ_3 ne sont pas en CNF, mais peuvent être réécrites facilement en CNF en utilisant l'équivalence :

$$\Psi_1 \Longrightarrow \Psi_2 \quad \text{équivalent à} \quad \neg \Psi_1 \vee \Psi_2$$

La formule Ψ_3 est déjà en CNF.

On montre maintenant que $(G = (V, E), k) \mapsto \Phi$ est une réduction.

(a) Supposons que $(G = (V, E), k)$ est instance positive de **3-COL-restreint**. Il existe donc un $U \subseteq V$ t.q. $|U| = k$ est une coloration valide $c : V \backslash U \to \{1, 2, 3\}$. On obtient une valuation σ qui satisfait Φ en mettant

$$\sigma(x_{u,i}) = \begin{cases} 1 & \text{si } u \notin U \text{ et } c(u) = i \\ 0 & \text{sinon} \end{cases}$$

$$\sigma(y_{v,j}) = \begin{cases} 1 & \text{si } v \text{ est le } j\text{-ème sommet de } U \\ 0 & \text{sinon} \end{cases}$$

(b) Supposons mainetant que Φ est satisfaisable, donc qu'on a une valuation σ qui satisfait Φ. On met $U = \{v \mid \text{ il existe } j \text{ t.q. } \sigma(y_{v,j}) = 1\}$. Grâce à la sous-formule Φ_3, l'ensemble U est de taille k.

On définit maintenant pour tout $u \notin U$: $c(u) = i$ si $\sigma(x_{u,i}) = 1$. Grâce à la sous-formule Φ_1 la fonction c colorie chaque sommet de $V \setminus U$ par une et une seule couleur. Et grâce à la sous-formule Φ_2, le coloriage c est valide.

Exercice 5 (5 points) : NP

On considère la variante suivante de SAT :

SAT-restreint
Entrée : formule booléenne φ en forme normale conjonctive (conjonction de clauses), et paramètre k.
Question : est-ce que φ a une valuation qui la satisfait **et** qui met *au plus* k variables à *vrai* ?

1. Montrez que le problème **SAT-restreint** est dans NP.
2. Donnez une réduction de SAT à **SAT-restreint**.
3. Déduisez que **SAT-restreint** est NP-complet.

Exercice 6 (3 points + 2 points) : problème de l'arrêt

1. (3 p.) Soit P un programme WHILE avec variables x_0, x_1, qui prend en entrée un entier. On sait que P satisfait l'invariant suivant :

   > *Sur entrée n, les valeurs de x_0 et x_1 durant l'exécution de P sont toujours inférieures à n^2.*

   Décrivez un algorithme qui répond à la question suivante :

   Entrée : entier n.
   Question : est-ce que P s'arrête sur n ?

   Correction : Comme le programme satisfait l'invariant, on sait que le nombre de configurations possibles sur entré n est borné par $|P| \times n^4$ (une configuration consiste du numéro de l'instruction actuelle, plus les valeurs de x_0, x_1).
   Donc, pour décider si P s'arrête sur n il suffit de simuler P sur n pour au plus $|P| \times n^4$ pas - au-delà de cette valeur une configuration se répète, donc P ne termine pas.
2. (**2 p. additionnels**) Montrez que la question suivante est indécidable :

   Entrée : Programme WHILE P.
   Question : est-ce qu'il existe une entrée n de P tel que la valeur de x_0 dépasse n^2 au cours de l'exécution de P sur n ?

   Indication : vous pouvez réduire à partir de la question d'arrêt de programme sur entrée 0 (HALT_0).

   Correction : Réduction de HALT_0. On veut savoir si un programme P termine sur entrée 0.
   A partir de P on construit le programme P' suivant. Sur l'entrée n le programme P' simule P sur entrée 0 pour au plus n^2 pas. La variable x_0 n'est pas utilisée dans la simulation. Si la simulation s'arrête avant n^2 pas, P met $x_0 := n^2 + 1$ et s'arrête. Sinon, P s'arrête (et la valeur de x_0 reste 0).
   On voit que si P ne s'arrête pas sur 0 alors la valeur de x_0 est toujours 0, pour tous les calculs de P'. Par contre, si P s'arrête sur 0 alors il existe n t.q. la valeur de x_0 durant le calcul de P' sur n depasse n^2 : n est le nombre de pas du calcul de P sur 0.