IFT503/711 – Théorie du calcul

Université de Sherbrooke

Devoir 3

Enseignant:	Manuel Lafond
Date de remise:	jeudi 4 mars 2021 avant 11h59 AM
À réaliser:	individuellement ou à deux au 1er cycle individuellement aux cycles supérieurs
Modalités:	à remettre via turnin
Pointage:	sur 40 points au 1er cycle (+ 5pts bonus pour ★) sur 50 points aux cycles supérieurs

Question 1.
Répondez à ces questions d'échauffement:

(a) Montrez que le langage UPREM $= \{1^n : n$ est un nombre premier$\}$ est dans P. 2 pts

Idée. Pour chaque entier i entre 1 et n, vérifier si i divise n, accepter si et seulement si ce n'est jamais le cas. Ceci demande n itérations, ce qui est polynomial en la taille de l'entrée (n bits).

(b) Montrez que le langage COMP $= \{n : n$ encode un nombre non-premier en binaire$\}$ est dans NP. 2 pts

Idée. Ici on n'a pas le luxe de tester une division avec n entiers. Ceci est parce que l'entrée est de taille $O(\log n)$ en binaire, et n itérations serait exponentiel. Mais on peut utiliser $\langle n, d\rangle$ en guise de certificat, où d est un diviseur de n certificant que n n'est pas premier. Même encodé en binaire, on peut vérifier en temps polynomial que d divise n.

(c) Montrez que le langage PREM $= \{n : n$ encode un nombre premier en binaire$\}$ est dans co-NP. 2 pts

Idée. Le complément de PREM est COMP, qui est dans NP, donc PREM est dans co-NP.

(d) Donnez un exemple de langage qui n'est pas dans NP $\cup$ co-NP. Justifiez votre réponse. 2 pts

Idée. Un langage indécidable, disons A_{TM}. On a vu qu'un langage de NP pouvait être décidé en temps exponentiel. Donc un langage indécidable ne peut pas être dans NP. Aussi, le complément d'un langage indécidable est aussi indécidable. C'est-à-dire, $\overline{A_{TM}}$ est indécidable, car sinon on pourrait aussi décider A_{TM}. Donc $\overline{A_{TM}}$ n'est pas dans NP, et donc son complément A_{TM} n'est pas dans co-NP.

Question 2.
Montrez que les deux langages décrits ci-bas sont NP-complet.

(a) Dans le problème du NO-BAD-SUM, on reçoit un ensemble I d'entiers et un ensemble S de sommes à éviter, puis on cherche un sous-ensemble $I' \subseteq I$ de taille maximum tel que toute paire de I' ne somme pas à un élément de S. Tous les entiers de I et S sont encodés en binaire. En terme de langage, on a: 8 pts

$$\text{NO-BAD-SUM} = \{\langle I, S, k\rangle : \text{il existe } I' \subseteq I \text{ avec } |I'| \geq k \text{ tel que pour tout } i, j \in I',\ i + j \notin S\}$$

Montrez que NO-BAD-SUM est NP-complet.
Suggestion: IND-SET.

Idée. NO-BAD-SUM est dans NP car $I' \subseteq S$ peut servir de certificat. Pour la réduction, on peut réduire de IND-SET. Pour une instance $\langle G, k\rangle$, on suppose que les sommets sont numérotés $v_1, \ldots, v_n$. Pour chaque v_i, on génère l'entier 2^i qu'on ajoute à S (donc le i-ème bit à 1. Pour chaque arête $v_iv_j \in E(G)$, on ajoute à $2^i + 2^j$ à I.

Je vous laisse argumenter que si on a un ensemble indépendant I' de G, alors $S' = \{2^i : v_i \in I'\}$ est une solution pour NO-BAD-SUM. La partie **hyper-importante** ici est d'argumenter qu'on ne peut pas avoir inclut deux entiers dans S' de somme interdite par accident. C'est pour ça qu'on a pris des puissances 2^i. Si on avait ajouté à S les entiers $\{1, 2, \ldots, n\}$, on aurait eu ce problème. Je vous laisse aussi argumenter que si on a $S' \subseteq S$ qui évite les sommes interdites, alors $\{v_i : 2^i \in S'\}$ est un ensemble indépendant.

(b) Dans le problème de *l'intersection bornée*, on reçoit des ensembles $A_1, \ldots, A_n$ et $B_1, \ldots, B_m$. On veut savoir s'il existe un ensemble X tel que $|X \cap A_i| \geq 1$ pour tout $i \in \{1, \ldots, n\}$, et $|X \cap B_i| \leq 1$ pour tout $i \in \{1, \ldots, m\}$. 8 pts

$$\begin{aligned}\text{INTER-BORNE} = \{\langle A_1, \ldots, A_n, B_1, \ldots, B_m\rangle : \exists X\ (&\forall i \in \{1, \ldots, n\}, |X \cap A_i| \geq 1 \wedge \\ &\forall i \in \{1, \ldots, m\}, |X \cap B_i| \leq 1)\}\end{aligned}$$

Montrez que INTER-BORNE est NP-complet.

Suggestion: 3-SAT.

Idée. INTER-BORNE est dans NP car X peut servir de certificat. Pour la réduction via 3-SAT et une instance ϕ, on génère des ensembles sur univers $\{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$. Oour chaque clause C_i, on ajoute un ensemble A_i contenant les trois litéraux de C_i (i.e. les 3 variables et leur type, positif ou négatif). Pour chaque variable x_i, on ajoute un ensemble $B_i = \{x_i, \overline{x}_i\}$.

Je vous laisse argumenter que si une assignation satisfait ϕ, alors X correspondant aux valeurs choisies par l'assignation intersectera au moins un élément de chaque A_i, et exactement un élément de chaque B_i. Conversément, si on trouve un ensemble X à l'instance INTER-BORNE, les B_i forcent le choix d'au plus une valeur par variable, et intersecter avec les A_i force de satisfaire chaque clause.

Question 3.
On écrit $\langle \phi, n\rangle$ pour dénoter une formule booléenne qui utilise n variables $x_1, \ldots, x_n$. On dénote par $\#sat(\phi, n)$ le nombre d'assignations de ces n variables qui satisfont ϕ.

(a) Considérez le langage UNIQUESAT $= \{\langle \phi, n\rangle : \#sat(\phi, n) = 1\}$. Dites pourquoi l'argument suivant qui prétend que UNIQUESAT $\in$ NP est erroné. 3 pts

Soit $\langle \phi, n\rangle$ une formule à n variables. En guise de certificat vérifiant que $\langle \phi, n\rangle \in$ UNIQUESAT, on prend une assignation A qui satisfait ϕ, ce qui est facile à vérifier en temps polynomial. Il existe donc un vérificateur pour UNIQUESAT, et le langage est dans NP.

Idée. Le certificat ne garantit pas que l'assignation est la seule et unique pouvant satisfaire ϕ.

(b) Montrez que si vous avez un oracle pour UNIQUESAT, alors vous pouvez décider SAT en temps polynomial. 4 pts

Idée. On créé une nouvelle variable z et on donne à l'oracle $\phi' = \phi \vee (x_1 \wedge \ldots \wedge x_n)$. Notez que ϕ' est toujours satisfaisable par au moins une assignation. Si l'oracle rejette ϕ', alors il existe au moins deux façons de satisfaire ϕ'. Donc, il doit y avoir une façon de satisfaire ϕ et on peut accepter ϕ. Si l'oracle accepte, $x_1 = T, \ldots, x_n = T$ est l'unique façon de satisfaire ϕ'. On doit encore déterminer si cette assignation peut satisfaire ϕ. Pour ce faire, on appelle encore l'oracle avec $\phi'' = \phi \vee (\overline{x}_1 \wedge \ldots \wedge \overline{x}_n)$. Si l'oracle rejette, comme avant il faut que ϕ soit satisfaisable. Si l'oracle accepte, alors $x_1 = F, \ldots, x_n = F$ est l'unique assignation possible. Comme l'oracle avait accepté ϕ', on déduit que ϕ ne peut pas être satisfaisable, et donc on peut rejeter ϕ.

(c) Soit le langage MAJSAT $= \{\langle \phi, n\rangle : \#sat(\phi, n) \geq 2^n/2\}$, i.e. les formules satisfaites par au moins la moitié des assignations. Dites pourquoi l'argument suivant qui prétend que MAJSAT $\in$ NP est erroné. 3 pts

Soit $\langle \phi, n\rangle$ une formule à n variables. Soient $A_1, A_2, \ldots, A_k$ la liste des assignations qui satisfont ϕ, qui nous serviront de certificat. Pour chaque A_i, $1 \leq i \leq k$, on peut vérifier en temps polynomial que A_i satisfait bel et bien ϕ. Une fois que c'est fait, il suffit de vérifier que $k \geq 2^n/2$. Il existe donc un vérificateur pour MAJSAT, et le langage est dans NP.

Idée. Ceci prend un temps exponentiel à vérifier.

(d) Montrez que MAJSAT est NP-difficile. 6 pts

Indice: étant donné une formule booléenne ϕ sur variables $x_1, \ldots, x_n$, considérez

$$(\overline{x_0} \wedge \phi) \vee (x_0 \wedge (x_1 \vee \ldots \vee x_n))$$

où x_0 est une nouvelle variable.

Idée. L'indice disait tout. Étant donné une instance ϕ de SAT, on la transforme en $\phi' = (\overline{x_0} \wedge \phi) \vee (x_0 \wedge (x_1 \vee \ldots \vee x_n))$ pour notre instance de MAJSAT. Notez que ϕ' a $n+1$ variables. Si ϕ est satisfaisable, alors on peut satisfaire ϕ' en satisfaisant $(\overline{x}_0 \wedge \phi)$, ou bien en choisissant une des $2^n - 1$ assignations satisfaisant $(x_0 \wedge (x_1 \vee \ldots \vee x_n)$. Ceci donne au moins $2^n - 1 + 1 = 2^n = 2^{n+1}/2$ assignations pour ϕ', qui est donc dans MAJSAT. Conversément, si ϕ' a au moins $2^{n+1}/2 = 2^n$ assignations satisfaisantes, exactement $2^n - 1$ peuvent satisfaire la partie droite. Donc, au moins une assignation satisfait $(\overline{x}_0 \wedge \phi)$, ce qui implique que ϕ est satisfaisable.

★ Question 4. (cycles supérieurs)
Soient EXPTIME $= \bigcup_{k\in\mathbb{N}} DTIME(2^{n^k})$ et NEXPTIME $= \bigcup_{k\in\mathbb{N}} NTIME(2^{n^k})$ les classes des langages décidables en temps exponentiel, respectivement avec une MT déterministe et une MT non-déterministe. 10 pts

Montrez que si EXPTIME $\neq$ NEXPTIME, alors P $\neq$ NP.

Suggestion: Prenez un langage L dans NEXPTIME, puis montrez que $\{pad(w, f(w)) : w \in L\}$ est dans NP pour un certain k et une fonction $f(w)$ appropriée, où $pad(w, f(w))$ est un mot formé de w suivi de $1^{f(w)}$. Faites ensuite une preuve par contraposition.

Idée. Tout était dans la suggestion. Soit $L \in NEXPTIME$ et M qui décide L en temps exponentiel non-déterministe. Supposons que le temps de M est borné par $2^{|w|^k}$ sur entrée w, où k est une constante. Soit $L' = \{pad(w, 1^{2^{|w|^k}} - |w| : w \in L\}$. Pour montrer que L' est dans NP, on conçoit une MT non-déterministe M' qui vérifie que son entrée est un mot w suivi de $2^{|w|^k} - |w|$ symboles 1. On remarque que si c'est le cas, w est unique. En effet, si l'entrée contient m symboles, il n'y a qu'une solution dans les entiers pour n à l'égalité $m = n + (2^{n^k} - n)$. Une fois que M' a extrait w, M' simule M sur entrée w en temps non-déterministe $O(2^{|w|^k})$, ce qui est polynomial par rapport à l'entrée donnée à M'. Puisque le résultat de M' et M est le même par rapport aux entrées w possibles, on déduit que $L' \in$ NP.

Si on suppose que P = NP, alors L' est décidable en temps polynomial par une MT que l'on appellera R. Soit w une instance de L. On conçoit une MT déterministe R' pour décider si $w \in L$. Il suffit d'ajouter $2^{|w|^k} - |w|$ symboles 1 à w et l'envoyer à R. Ceci décidera si $w \in L$ en temps polynomial par rapport à $2^{|w|^k}$, ce qui est un temps exponentiel déterministe par rapport à $|w|$. Ceci montre que $L \in EXPTIME$.

Puisque L était un langage arbitraire de NEXPTIME, nos suppositions mènent EXPTIME = NEXPTIME. On a donc montré que si P = NP, alors EXPTIME = NEXPTIME, ce qui est équivalent à l'énoncé de la question.