Calculabilité / Complexité (L3)
Examen "Complexité"

12 jan. 2017 / 14h00–17h00 / Salle C315

Exercice (6 points)

Pour un entier $k > 0$ et un alphabet fini A, une fonction totale $f : A^* \to A^*$ est dite $\log^k$space s'il existe une machine de Turing déterministe qui calcule $f(x)$ pour tout $x \in A^*$ en utilisant un espace de travail $O(\log^k n)$, c.-à-d. $\leq c(\log |x|)^k$, pour un entier c. On dit que f est "logspace" quand $k = 1$, et polylogspace si elle est $\log^k$space pour un $k \in \mathbb{N}$.

Note: pour simplifier les calculs, on parle ici d'une fonction $\log : \mathbb{N} \to \mathbb{N}$ définie par $\log(0) = 0$ et, pour $n > 0$, $\log(n) = \lfloor \log_2 n \rfloor$. Dans le même esprit, on convient que l'adjectif "logspace" et ses dérivés sont invariables.

Question 1. Est-ce que les fonctions polylogspace sont calculables en temps polynomial? Justifiez.

Question 2. Montrez que si f est $\log^k$space alors $|f(x)|$ est en $|x|^{O(\log^{k-1} |x|)}$.

Question 3. Est-ce que la composition de deux fonctions $\log^k$space est elle-même $\log^k$space? Justifiez.

Problème (14 points)

On dit qu'un mot u est sous-mot d'un mot v, noté $u \preceq v$, si u est une sous-suite de v, c.-à-d., obtenue en retirant un nombre arbitraire de lettres. P.ex. abba $\preceq$ abracadabra. En particulier, $\epsilon \preceq u \preceq u$ pour tout u, où ϵ dénote le mot vide.

On s'intéresse au problème LCS (pour "Longest Common Subword"). Il prend en entrée un alphabet A, une liste $u_1, \ldots, u_k$ de mots dans A^*, ainsi qu'un entier N. La question à résoudre est "existe-t-il un mot v tel que $|v| = N$ et $v \preceq u_i$ pour $i = 1, \ldots, k$?"

Question 1. Est-ce que l'instance ($A = \{a, b, c\}, u_1 = aabc, u_2 = bbca, u_3 = ccab, N = 2$) est positive? Est-ce que l'instance

$$(A = \{0, 1, \ldots, 9\}, u_1 = 31415926535897932384 6, u_2 = 14142135623730950488 0, N = 8)$$

est positive?

Justifiez brièvement dans les deux cas.

Question 2. On suppose que dans une instance de LCS, l'entier N est donné en base 1, p.ex. sous la forme $\mathtt{I}^N = \mathtt{II} \ldots \mathtt{I}$, et donc une instance a une longueur $O(N + L \log L)$ si $L = |u_1| + \cdots + |u_k|$. (NB: le facteur $\log L$ compense le fait que les u_i peuvent utiliser jusqu'à L lettres différentes.)

Montrez que LCS est dans NP.

Question 3. Soit LCS_b le problème qui est comme LCS sauf que N est écrit en base 2, de sorte que la taille d'une instance est $O(\log N + L \log L)$.

3a. Donnez une réduction logspace de LCS à LCS_b.

3b. Donnez ensuite une réduction logspace de LCS_b à LCS .

Pour ces deux questions on justifiera la correction et on détaillera (sans forcément écrire un programme) les algorithmes utilisés par les réductions de façon à bien comprendre comment un espace logarithmique est suffisant.

Question 4. On veut montrer que LCS est NP–difficile. Pour cela on part du problème NODECOVER vu en TD et connu pour être NP-complet. On rappelle qu'une instance de NODECOVER est constituée d'un graphe simple (i.e., non orienté, sans arêtes multiples ni boucles) $G = (V, E)$ et d'un entier K et qu'on se demande si G admet un recouvrement de cardinal au plus K, sachant qu'un recouvrement est un ensemble de sommets $C \subseteq V$ tel que chaque arête de E a une (au moins) de ses extrémités dans C.

Soit une instance $G = (V, E), K$ avec $|V| = \ell$ sommets et $|E| = m$. On va considérer des mots $u_0, u_1, \ldots, u_m$ sur l'alphabet V. On pose d'abord $u_0 = v_1 v_2 \ldots v_l$ en fixant une énumération de V. On fixe ensuite une énumération $e_1, e_2, \ldots, e_m$ des arêtes et pour, $i \in \{1, \ldots, m\}$, on pose $e_i = \{v_r, v_s\}$ de sorte que $r < s$. On pose alors

$$u_i = v_1 v_2 \ldots v_{r-1} v_{r+1} \ldots v_l v_1 v_2 \ldots v_{s-1} v_{s+1} \ldots v_l$$

4a. Prouvez que G admet un recouvrement de taille $\ell - N$ ssi il existe un mot w de longueur N qui soit sous-mot de chacun des u_i pour $i = 0, \ldots, m$.

4b. Donnez une réduction logspace de NODECOVER à LCS. Justifiez soigneusement (sans écrire de code) que votre réduction est bien calculable en espace logarithmique.

4c. Conclure en donnant la complexité de LCS et celle de LCS_b.

Question 5. On s'intéresse à deux versions de LCS. Pour deux mots u, v, on note $u \preceq_{\text{no}} v$, et on dit que "u est un sous-mot non orienté de v", ssi $u \preceq v$ ou $\widetilde{u} \preceq v$. Ici $\widetilde{u}$ est le mot miroir de u: p.ex. $\widetilde{\mathtt{aab}} = \mathtt{baa}$. De même on note $u \preceq_{\text{cy}} v$, et on dit que "u est cycliquement sous-mot de v", si $u_2 u_1 \preceq v$ pour une factorisation $u = u_1 u_2$ de u.

5a. Est-ce que $\preceq_{\text{no}}$ est un préordre, c.-à-d., une relation réflexive et transitive, sur les mots? Même question pour $\preceq_{\text{cy}}$. Justifiez.

5b. Le problème LCS_{no} est une version de LCS où on demande s'il existe un sous-mot non orienté v de longueur N tel que $v \preceq_{\text{no}} u_i$ pour $i = 1, \ldots, k$?

Montrez que LCS_{no} est NP-complet.

5c. Même question pour le problème LCS_{cy} qui demande si les u_i admettent un sous-mot commun de longueur N au sens de $\preceq_{\text{cy}}$.

Question 6. On note LCS_2 la restriction de LCS où les instances contiennent exactement deux mots, i.e., $k = 2$.

6a. Donnez un algorithme en PTIME qui résout LCS_2. Justifiez sa correction et votre analyse de complexité.

6b. Plus généralement, pour $k \in \mathbb{N}$ fixé, le problème LCS_k est la restriction de LCS où les instances contiennent exactement k mots $u_1, \ldots, u_k$.

Soit $k \in \mathbb{N}$. Est-ce que LCS_k est dans PTIME ?