Complexité - TD 1.1

Benjamin Bordais

15 Novembre 2021

On rappelle qu'une machine de Turing utilisant un espace $O(f)$ peut être modifié pour n'utilisé qu'un espace f (cela est donné par le théorème d'accélération en espace).

Question 1 *Rappeler quel est l'espace utilisé par une machine de Turing (en particulier, pour les cas où l'on considère des machines qui utilisent un espace $f(n) < n$ où n est la taille de l'entrée).*

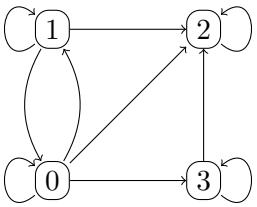
Solution 1 *Une machine de Turing possède $k+2$ rubans avec deux rubans distingués : le ruban d'entrée – en lecture seule – et le ruban de sortie – en écriture seule. Il faut noter que le ruban de sortie n'est pas pertinent lorsque l'on considère des machines de Turing traitant de problèmes de décision (étant donné que la sortie est alors oui ou non et est encodée dans les états de la machine). Les k autres rubans (avec k fixe, ne dépendant pas de l'entrée) sont les rubans de travail sur lesquels l'espace utilisé par la machine est compté : il s'agit alors du maximum (sur toute la durée de l'exécution) du nombre de cases mémoire utilisées cumulé sur toutes les bandes de travail.*

Représentation des graphes

Un graphe orienté est un couple (V, E), où $V = \{0, \dots, n-1\}$ est un ensemble fini de n sommets (*vertex*) et $E \subseteq V \times V$ est un ensemble fini d'arêtes (*edge*). Il existe deux façons standard de représenter un graphe avec un alphabet fini Σ contenant au moins $\{0, 1\}$, que nous décrivons ci-dessous. Étant donné un entier m, on note $\overline{m} \in \{0, 1\}^*$ son codage en base deux.

Liste d'adjacence : Une représentation en liste d'adjacence de G code le graphe comme une liste l de longueur n, où le u-ième élément de la liste est la liste de tous les sommets v tels que $(u, v) \in E$.

Matrice d'adjacence : Une représentation en matrice d'adjacence de G code le graphe comme une matrice M carrée à n lignes et n colonnes, et à valeurs dans $\{0, 1\}$, telle que $M[u][v] = 1$ si et seulement si $(u, v) \in E$.



Question 2 *Proposer un alphabet Σ pour représenter un graphe (que l'on pourrait utiliser pour les deux représentations).*

Solution 2 *On peut par exemple considérer l'alphabet* $\Sigma = \{\mathtt{0}, \mathtt{1}, /, \bullet, \#\}$. *Dans ce cas, le codage d'un graphe* G *par liste d'adjacence serait de la forme :*

$$l_G \stackrel{def}{=} k_0^0/\ldots/k_{m_1}^0 \bullet \cdots \bullet k_0^{n-1}/\ldots/k_{m_{n-1}}^{n-1}\#$$

Ainsi, on code les nœuds (représenter en tant qu'entiers) en binaire[1] *avec* $\mathtt{0}, \mathtt{1}$ *et* $/, \bullet$ *sont des séparateurs respectivement des entiers et des listes, tandis que* $\#$ *est le symbole de fin de mots.*

De la même manière, le codage par matrice d'adjacence est de la forme :

$$m_G \stackrel{def}{=} m_{0,0}m_{0,1}\ \ldots m_{0,n-1} \bullet \cdots \bullet m_{n-1,0}\ldots m_{n-1,n-1}\#$$

Question 3 *Donnez une représentation du graphe ci-dessus en liste d'adjacence et en matrice d'adjacence à l'aide de cet alphabet.*

Solution 3 *Dans le cas de graphe* G*, on a :*

$$l_G := \mathtt{0/01/10/11} \bullet \mathtt{0/01/10} \bullet \mathtt{10} \bullet \mathtt{10/11}\#$$

et

$$m_G := \mathtt{1111} \bullet \mathtt{1110} \bullet \mathtt{0010} \bullet \mathtt{0011}\#$$

Question 4 *Montrez qu'il existe une fonction logspace (i.e. qui peut être implémenté par une MT qui utilise un espace en* $\log(n)$*) qui à toute représentation d'un graphe* G *en liste d'adjacence associe une représentation de* G *en matrice d'adjacence.*

Réciproquement montrez qu'il existe une fonction logspace qui à toute représentation d'un graphe G *en matrice d'adjacence associe une représentation de* G *en liste d'adjacence.*

Solution 4 *Pour passer de la représentation par liste d'adjacence à la représentation par matrice d'adjacence, on considère deux compteurs* i *et* j *qui varient entre 0 et* $k-1$ *(où* k *est le nombre de symbole* $\bullet$ *plus 1) initialisés à 0 ainsi qu'un pointeur qui va parcourir (plusieurs fois) la liste d'indice* i*. Le compteur* i *est incrémenté à chaque fois qu'un symbole* $\bullet$ *est vu. Ensuite pour chaque valeur de* j *entre 0 et* $k-1$*, le pointeur va chercher dans la liste d'indice* i *un nœud égal à* j*. Si on en trouve un, un* $\mathtt{1}$ *est écrit sur la bande de sortie, sinon un* $\mathtt{0}$ *est écrit. Lorsque l'on a finit la boucle pour* $j = k-1$*, le symbole* $\bullet$ *est écrit et* j *est réinitialisé à* 0*. L'espace utilisé par cette algorithme est bien logarithmique car on considère deux compteurs et un pointeur qui, s'ils sont implémentés en binaire, prennent un espace logarithmique en la taille de l'entrée.*

Réciproquement, on considère un algorithme qui fait varier un compteur i *entre* 0 *et* $k-1$ *(où* k *est le nombre de symbole* $\bullet$ *plus 1). Ce compteur* i *est incrémenté dès qu'un bit* $\mathtt{0}$ *ou* $\mathtt{1}$ *est vu et est réinitialisé à chaque symbole* $\bullet$*. A chaque symbole* $\bullet$*, ce même symbole est écrit sur la bande de sortie et dès qu'un* $\mathtt{1}$ *est vu, le compteur* i *est copié sur la bande de sortie, précédé d'un symbole / si ce n'est pas la première fois que le compteur* i *est copié depuis qu'il a été réinitialisé. L'espace utilisé par cet algorithme correspond à la taille prise par ce compteur qui, s'il est implémenté en binaire, prend un nombre de bits logarithmique en la taille de l'entrée.*

1. La manière dont on représente les entiers, en binaire ou en unaire, peut changer radicalement la complexité d'un problème étant donné que celle-ci dépend de la taille de l'entrée qui est exponentiellement plus grande avec une représentation unaire plutôt qu'une représentation binaire. Pour le cas qui nous intéresse, on peut s'en sortir avec les deux représentations.

Fonction LOGSPACE et L

Question 5 *Montrer que toute machine de Turing déterministe qui calcule en espace logarithmique (c'est-à-dire telle que l'espace utilisé sur une entrée de taille n est au plus $\log n$), s'arrête après un nombre d'étapes polynomial. Qu'en est-il si l'espace pris est en $\log^k n$ pour un $k \geq 0$.*

Solution 5 *Considérons une machine de Turing M sur un alphabet Q et un alphabet Σ qui fonctionne en espace logarithmique (sans perte de généralité, on suposera une seule bande de travail), c'est-à-dire tel qu'à tout instant d'un calcul sur une entrée w de taille $n = |w|$, le nombre de cellules sur les bande de travail utilisé est inférieur à $k \cdot \log(n)$ pour un k donné. Alors, sur une entrée w de taille n, le nombre de configurations différentes est borné par $f(n) = |Q| \cdot |\Sigma|^{k \cdot \log(n)} \cdot (k \cdot \log n) \cdot n$ (les deux derniers termes faisant référence à l'emplacement de la tête d'écriture sur la bande de travail et de lecture sur la bande d'entrée). On a alors qu'il existe d tel que $f(n) = O(n^d)$. Étant donné que la machine est déterministe, celle-ci boucle nécessairement si elle passe deux fois par la même configuration. Ainsi, le nombre d'étapes pris par cette machine pour s'arrêter est au plus polynomial.*

Lorsque $k \geq 2$, le nombre d'étapes n'est plus borné par un polynôme. En effet, sur une entrée de taille n (par exemple n écrit en unaire), on peut réserver sur une bande de travail un espace de taille $\log^2 n$. On peut ensuite encoder un compteur en bianire sur cet espace qui va donc compter jusqu'à $2^{\log^2 n}$. On peut donc écrire $2^{\log^2 n}$ en unaire sur la bande de sortie. Or, $2^{\log^2 n} = n^{\log n}$, ce qui n'est borné par un polynôme.

Question 6 *Soient $h_1 : L_1 \mapsto L_2$ et $h_2 : L_2 \mapsto L_3$ deux fonctions calculables en espace logarithmique par des machines déterministes.*

Montrer que la fonction $h_2 \circ h_1 : L_1 \mapsto L_3$ peut aussi être calculée en espace logarithmique par une machine déterministe.

Solution 6 *La question précédente nous donne qu'il existe un certain d tel que, pour toute entrée w telle que $n = |w|$, on a $|h_1(w)| \leq n^d$. Ainsi, l'espace pris par un calcul de h_2 sur $h_1(w)$ pend un espace en $O(\log(n))$ et est donc logarithmique. Cependant, on doit ici remarquer qu'il n'est pas possible d'écrire la résultat du calcul de h_1 sur une bande de travail pour ensuite lancer directement h_2 dessus vu qu'on utiliserait alors un espace plus grand que logarithmique. Il faut donc que, lors du calcul de $h_2(h_1(w))$, lorsque h_2 doit lire le i-ème bit de $h_1(w)$ l'on effectue une nouvelle fois le calcul de $h_1(w)$ en ne renvoyant que le bit d'intérêt. Comme on considère la complexité en espace, et non en temps, on peut effectuer plusieurs fois le même calcul sans que cela affecte la complexité (à condition que l'on réutilise l'espace utilisé par les calculs précédents).*

Considérons un alphabet fini A. Pour une fonction partielle $f : A^* \to A^*$ défini sur son domaine $\mathsf{dom}\, f$, on associe le langage :

$$D_f = \{\langle x, i, a\rangle \in A^* \times \mathbb{N} \times A \mid x \in \mathsf{dom}\, f \text{ et } 0 < i \leq |f(x)| \text{ avec } a \text{ la } i\text{-eme lettre de } f(x)\}.$$

On suppose que i est représenté en binaire.

On dit qu'une fonction est calculable en espace logarithmique s'il existe une machine de Turing déterministe M_f qui, sur une entrée $x \in A^*$, termine et écrit $f(x)$ sur sa bande de sortie en utilisant un espace $\log x$. Si $f(x)$ n'est pas définit, la machine M_f doit atteindre un état rejetant.

Question 7 *Montrer qu'une fonction totale f est calculable en espace logarithmique si et seulement si $D_f \in \mathsf{L} = \mathsf{SPACE}(\log)$ et il existe un polynôme p tel que $|f(x)| \leq p(|x|)$ pour tout x de A^*.*

Solution 7 $(\Rightarrow)$ *: Par la question 5, on a que si f est calculable en espace logarithmique, alors le nombre d'étape pris est borné par un polynôme. Considérons maintenant une instance $\langle x, i, a\rangle$ de D_f. On peut simuler M_f sur l'entrée x en gardant un compteur sur ce qui est écrit en sortie. Lorsque ce compteur atteint i, on compare la lettre écrite sur la bande de sortie et a en acceptant si c'est la même et rejetant si c'est différent. Si la simulation termine avant le compteur n'atteigne la valeur i, on rejette (on a excédé la longueur de $p(|x|)$). L'espace pris correspond donc à celui utilisé pour la simulation plus un compteur écrit en binaire.*

$(\Leftarrow)$ *: Étant donné une machine fonctionnant en espace logarithmique décidant D_f, sur une entrée x, on peut calculer $f(x)$ en bouclant sur tous les entiers i entre 1 et $p(|x|)$ et toute les lettres de A pour déterminer quel triplet $\langle x, i, a\rangle$ appartient à D_f. L'espace pris est alors l'espace pris pour lancer D_f plus des compteurs sur i et a après avoir calculer $p(|x|)$. L'espace pris pour stocker a est constant puisque l'alhabet A est fini et fixé indépendamment de l'entrée x. De plus, l'espace pris par le compteur sur $p(|x|)$ est $\log p(|x|) = O(\log |x|)$.*

Question 8 *Peut-on retirer l'hypothèse de l'existence d'un tel polynôme p ?*

Solution 8 *On ne peut pas se passer de cette hypothèse. Par exemple, pour $A = \{a\}$ et $f(x) = a^{2^{|x|}}$, pour tout $x \in A^*$, on a f qui n'est pas calculable en temps polynomial puisque cela suppose d'écrire un mot de taille exponentielle. A fortiori, f n'est pas calculable en espace logarithmique. Cependant, D_f est bien dans L. En effet, un triplet $\langle x, i, a\rangle$ est dans D_f si et seulement si $1 \leq i \leq 2^{|x|}$. Cela est bien vérifiable en espace logarithmique puisque cela revient à comparer le nombre de bit utilisé pour écrire i et la taille de x.*

Question 9 *Peut-on retirer l'hypothèse que la fonction f est totale ?*

Solution 9 *On ne peut pas non plus. En effet, pour tout ensemble $P \subseteq A^*$, on définit f_P par $f_P(x) = \epsilon$ pour tout $x \in P$ et f_P non définit sinon. Dans ce cas, D_{f_P} est vide et $|f(x)| = 0$ pour tout $x \in \mathsf{dom} f = P$. Ainsi, D_{f_P} est dans L et la taille de f_P est bornée par un polynôme. Cependant, si P est un ensemble indécidable, alors f_P n'est pas calculable.*