Complexité : TD 1

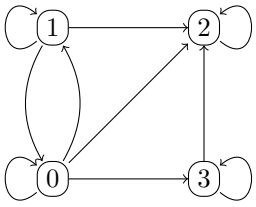
Chargé de TD : Adrien Koutsos

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 1 *Donnez une représentation du graphe ci-dessus en liste d'adjacence et en matrice d'adjacence.*

Question 2 *Donnez deux fonctions injectives τ_l et τ_m de l'ensemble des graphes dans Σ^* telles que pour tout G, $\tau_l(G)$ et $\tau_m(G)$ soient des représentations en, respectivement, liste d'adjacence et matrice d'adjacence de G.*

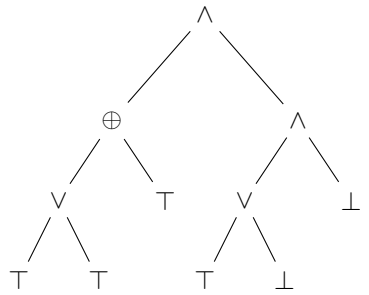
Question 3 *Montrez qu'il existe une fonction logspace qui à toute représentation d'un graphe G en liste d'adjacence associe une représentation de G en matrice d'adjacence.*

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.

Représentation des termes

Un terme booléen est un arbre binaire étiqueté par $\wedge, \vee, \neg, \top, \bot$, tel que un noeud étiqueté par $\wedge$ ou $\vee$ ait deux successeurs, un noeud étiqueté par $\neg$ ait un successeur et un noeud étiqueté par $\top$ ou $\bot$ ait zéro successeur.

Un terme booléen peut être représenté par un graphe étiqueté (le graphe étant représenté avec une liste d'adjacence), ou comme un mot sur l'alphabet $\Sigma = \{\wedge, \vee, \oplus, \top, \bot, (,)\}$.



$\wedge(\oplus(\vee(\top,\top),\top),\wedge(\vee(\top,\bot),\bot))$

Question 4 *Montrer qu'il existe deux fonctions logspace permettant de passer d'une représentation à l'autre.*

On peut naturellement définir la fonction eval qui évalue un terme booléen :

$$\mathsf{eval}(\top) = 1 \qquad \mathsf{eval}(\bot) = 0 \qquad \mathsf{eval}(\wedge(t_1, t_2)) = \mathsf{eval}(t_1).\mathsf{eval}(t_2)$$

$$\mathsf{eval}(\vee(t_1, t_2)) = 1 - (1 - \mathsf{eval}(t_1)).(1 - \mathsf{eval}(t_2))$$

$$\mathsf{eval}(\oplus(t_1, t_2)) = (1 - \mathsf{eval}(t_1)).\mathsf{eval}(t_2) + \mathsf{eval}(t_1).(1 - \mathsf{eval}(t_2))$$

Question 5 *Montrer que le problème suivant est dans* L *:*
Donnée : *Le codage en liste d'adjacence d'un terme booléen t.*
Question : *$\mathsf{eval}(t) = 1$?*
(Dans un premier temps, chercher un algorithme en SPACE$(log^2(n))$*)*

Transitivité des fonctions LOGSPACE

Question 6 *Montrer que toute machine de Turing déterministe qui calcule en espace logarithmique, s'arrête après un nombre d'étapes polynomial.*

Question 7 *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.

Question 8 *En déduire que le problème suivant est dans* L *:*
Donnée : *Un mot sur $\Sigma = \{\wedge, \vee, \oplus, \top, \bot, (,)\}$ codant un terme booléen t.*
Question : *$\mathsf{eval}(t) = 1$?*

Accélération linéaire

Question 9 *Montrer que pour tout $\epsilon > 0$,*

$$TIME(f(n)) \subseteq TIME(\epsilon f(n) + 2.n + C)\,.$$

Ou C est une constante. Évaluer le nombre d'états et de transitions de la nouvelle machine de Turing.

Machines alternantes

Une *machine de Turing alternante* est une machine de Turing dont les états sont étiquetés par les symboles booléens $\wedge, \vee, \top, \bot$. Lorsqu'on arrive dans un état $\top$, la machine accepte ; lorsqu'on arrive dans un état $\bot$, la machine rejette ; lorsqu'on arrive dans un état $\wedge$, la machine explore toutes les branches et accepte si toutes les branches acceptent ; lorsqu'on arrive dans un état $\vee$, la machine explore toutes les branches et accepte si l'une des branches accepte.

Formellement, une machine de Turing alternante à k bandes est un 6-uplet $(Q, q_0, \Sigma, \Gamma, \delta, \lambda)$ où :

- Q est un ensemble fini d'états,
- $q_0 \in Q$ est l'état initial,
- Σ est l'alphabet d'entrée, on y ajoute un marqueur de fin $\$ \notin \Sigma$,
- Γ est l'alphabet de travail, on y ajoute un caractère blanc $\# \notin \Gamma$,
- $\delta \subseteq Q \times (\Gamma \cup \{\#\})^k \times (\Sigma \cup \{\$\}) \times Q \times \Gamma^k \times \{\textbf{left}, \textbf{right}\}^k \times \{\textbf{left}, \textbf{right}\}$ est la fonction de transition,
- $\lambda : Q \longrightarrow \{\wedge, \vee, \top, \bot\}$ est la fonction d'étiquetage des états.

Une transition $(q, (a_1, \ldots, a_k), c, q', (b_1, \ldots, b_k), (m_1, \ldots, m_k), m) \in \delta$ indique que la machine peut passer de l'état q à l'état q' en lisant $(a_1, \ldots a_k)$ sur les bandes de travail et c sur l'entrée, et qu'elle écrit alors $(b_1, \ldots, b_k)$ sur les bandes de travail et se déplace dans la direction m_i sur chaque bande i, et dans la direction m sur l'entrée.

Comme pour les machines de Turing non déterministes, une machine de Turing alternante accepte une entrée en temps t si elle l'accepte en n'explorant que des configurations accessibles en au plus t étapes depuis la configuration initiale.

Une machine de Turing alternante accepte une entrée en espace s si elle l'accepte en n'explorant que des configurations dans lesquelles au plus s caractères sont écrits sur les bandes.

On note $ATIME(f(n))$ (respectivement $ASPACE(f(n))$) l'ensemble des langages acceptés par une machine de Turing alternante en temps (respectivement en espace) $f(n)$, n étant la taille de l'entrée.

Question 10 *Montrer que*

1. *pour $f(n) \geq n$, on a $ATIME(f(n)) \subseteq DSPACE(f(n))$.*
2. *pour $f(n) \geq \log(n)$, on a $ASPACE(f(n)) \subseteq \bigcup_{c>0} DTIME(c^{f(n)})$.*
3. *pour $f(n) \geq n$, on a $NSPACE(f(n)) \subseteq \bigcup_{c>0} ATIME(cf(n)^2)$.*
4. *pour $f(n) \geq n$ et $c > 0$, on a $DTIME(f(n)) \subseteq ASPACE(c\log(f(n)))$.*

Automates alternants

Une machine alternante qui n'a pas de bande de travail ($k = 0$) et qui ne se déplace que vers la droite sur la bande de lecture, est appelée un *automate alternant.*

Question 11 *Montrer que pour tout automate alternant à n états, il existe un automate non déterministe à 2^n états qui accepte le même langage.*