Théorème de Savitch

Benjamin Hellouin

Théorème 1 (de Savitch)**.**
Pour toute fonction $S(n) \geq \log(n), \mathbf{NSPACE}(S(n)) \subseteq \mathbf{SPACE}(S(n)^2)$.

On utilisera la vision de "deviner nondéterministiquement" pour avoir une meilleure intuition, mais on passe facilement à la vision avec certificat.

Définition 1. *Le problème décisionnel* ***PATH*** *est défini comme suit :*
Entrée : un graphe ***orienté*** $G = (V, E)$ *et deux sommets* s *(source) et* c *(cible) ;*
Sortie : OUI s'il existe un chemin de s *à* c *dans* G*, NON sinon.*

L'entrée est supposée sous forme de matrice d'adjacence (mais ça n'a pas d'importance) et n est le nombre de sommets ($\sqrt{}$ de taille de l'entrée, mais ça ne change rien).

Lemme 1. ***PATH*** *est* ***NL****-complet.*

Voici un algorithme non déterministe qui décide **PATH** en espace logarithmique :

$v \leftarrow s$
Tant que $v \neq c$ **faire**
  deviner le prochain sommet w
  Si $(v, w) \in E$ **alors**
    $v \leftarrow v'$
Sortie :

Soit P un problème de **NL** accepté par une machine M. Si on suppose que M n'a qu'une seule configuration acceptante (nettoyer le ruban et amener la tête à l'origine avant d'accepter), on voit que le problème d'acceptation d'une instance par M est équivalent à la recherche d'un chemin dans son graphe des configurations. De plus, une machine de Turing travaillant en espace logarithmique a un graphe de taille $2^{O(\log(n))} = P(n)$, donc **PATH** sera bien en espace logarithmique.

Lemme 2. ***PATH*** $\in \mathbf{SPACE}((\log \mathbf{n})^{\mathbf{2}})$.

Sur une instance de taille n, supposons que les sommets soient numérotés $\{1 \dots n\}$. On introduit la algorithme récursif **ATTEIGNABLE** qui étant donné deux sommets s et c et un entier i, décide si il existe un chemin de v vers s de taille 2^i.

Entrée : s, c, i
  $b \leftarrow$ **faux** (réponse)
  Pour tout $z \in V$ **faire**
    $b_0 = ATTEIGNABLE(s, z, i-1)$
    $b_1 = ATTEIGNABLE(z, c, i-1)$

Si $b_0 = b_1$ =**vrai alors**
$b \leftarrow$**vrai**
Sortie : b

Etudions la complexité spatiale de cet algorithme : hors des rappels récursifs, il utilise au plus un espace $O(\log n)$ (b, b_0, b_1 prennent un espace constant, z et les entrées un espace logarithmique). De plus, l'espace utilisé par le premier appel récursif peut être réutilisé (à l'exception de la réponse elle-même). On en déduit que le coût $S(i)$, indépendamment de s et c, vérifie $S(i) = \log n + S(i-1)$. On a donc $i = \lceil \log n \rceil$ appels récursifs, pour une complexité totale en espace $O(\log^2 n)$.