Chapitre VI

Complexité spatiale

On se pose le même problème que pour la complexité temporelle, mais, cette fois-ci, avec la quantité de mémoire (= le nombre de cases utilisées sur la bande) nécessaire à la résolution d'un problème.

Ce type de complexité est appelée la complexité spatiale.

Ceci va nous amener à poser la définition de nouvelles classes de complexité.

Puis, à rechercher les propriétés de ces classes entre-elles, et avec les classes de complexité temporelle.

1 Définitions

Définition : complexité spatiale pour une MT déterministe.

Soit M une MT déterministe qui s'arrête sur toutes ses entrées.

La complexité spatiale de M est une fonction $f : \mathbb{N} \to \mathbb{N}$ où $f(n)$ est le nombre maximum de cases différentes sur lesquelles M visitées sur n'importe quelle entrée de longueur n.

Définition : complexité spatiale pour une MT non déterministe.

Soit N une MT non-déterministe que s'arrête sur toutes ses entrées.

La complexité spatiale de N est la fonction $f(n)$ représentant le nombre de cases maximales utilisé lors de l'exécution de n'importe quelle branche de N pour n'importe quelle entrée de longueur n.

Remarques :

- Si la complexité spatiale de M est $f(n)$, on dit M s'exécute en espace $f(n)$.
- Défini ici en nombre de cases utilisées par la MT.
  Défini en octets sur un processeur.

Soit $f : \mathbb{N} \to \mathbb{R}$ une fonction.

On définit alors :

Définition VI.1 ($\texttt{SPACE}(f(n))$)

$\texttt{SPACE}(f(n))$ est l'ensemble des langages décidés par une MTD M qui s'exécute en espace $f(n)$.

Définition VI.2 ($\texttt{NSPACE}(f(n))$)

$\texttt{NSPACE}(f(n))$ est l'ensemble des langages décidés par une MTND M qui s'exécute en espace $f(n)$.

Note :

Dans un premier temps, on considérera que $f(n) \geq n$, car la bande d'entrée contient l'entrée qui elle-même est de taille n. Donc, si $f(n) < n$ cela signifierait que l'entrée n'a pas été lue en entier.

2 Premières propriétés

Théorème VI.1 (Savitch)

soit une fonction $f : \mathbb{N} \to \mathbb{R}$ telle que $f(n) \geq n$, alors $\texttt{NSPACE}(f(n)) \subseteq \texttt{SPACE}(f(n)^2)$.

Autrement dit, toute MTND qui s'exécute en espace $f(n)$ peut s'exécuter en espace $\texttt{SPACE}(f(n)^2)$ sur une MTD.

Démonstration: (idée principale)

On peut simuler une MTND en espace $f(n)$ sur une MTD en parcourant l'arbre d'exécution de manière préfixe (toutes les branches s'arrêtent). Une branche d'espace $f(n)$ s'exécute au plus en temps $f(n)$ (on ne visite pas plus de case que l'on a de temps). On a alors besoin de $f(n)$ espace de stockage sur une profondeur d'exécution $f(n)$, d'où une simulation en espace déterministe au plus de $f(n)^2$. □

Définition VI.3 (PSPACE)

PSPACE est la classe des langages qui sont décidables en espace polynomial par une MTD, à savoir : **PSPACE** $= \cup_k \texttt{SPACE}(n^k)$

Définition VI.4 (NPSPACE)

NPSPACE est la classe des langages qui sont décidables en espace polynomial par une MTND, à savoir : **NPSPACE** $= \cup_k \texttt{NSPACE}(n^k)$

Théorème VI.2

PSPACE = **NPSPACE**

Démonstration:

Par le théorème de Savitch, $\texttt{NSPACE}(f(n)) \subseteq \texttt{SPACE}(f(n)^2)$. Si $f(n)$ est un polynôme, $f(n)^2$ aussi. □

Proposition VI.1

SPACE = *co***SPACE**

Démonstration:

$\forall f(n), \texttt{SPACE}(f(n)) = \texttt{coSPACE}(f(n))$. Même idée que pour **P** = *co***P**. □

Proposition VI.2

PSPACE = co**NPSPACE**

Démonstration:

PSPACE = **NPSPACE**, donc *co***PSPACE** = *co***NPSPACE**. Conclure avec **SPACE** = *co***SPACE**. □

On a donc : **PSPACE** = co**PSPACE** = **NPSPACE** = co**NPSPACE**

3 Lien avec la complexité temporelle

Théorème VI.3
P ⊆ PSPACE

DÉMONSTRATION:

Si un langage est décidé par une MTD M en temps $f(n)$, alors M visite au plus $f(n)$ cellules différentes. Donc, $\texttt{TIME}(f(n)) \subseteq \texttt{SPACE}(f(n))$, et par conséquent **P** ⊆ **PSPACE**. □

Théorème VI.4
NP ⊆ PSPACE

DÉMONSTRATION:

De la même façon, $\texttt{NTIME}(f(n)) \subseteq \texttt{NSPACE}(f(n))$. Donc, **NP** ⊆ **NPSPACE**. Or, on a montré que **PSPACE** = **NPSPACE**. En conséquence, **NP** ⊆ **PSPACE**. □

Lemme VI.1
une MTD M en espace $f(n)$ a au plus $2^{O(f(n))}$ configurations différentes.

DÉMONSTRATION:

Notons $|Q|$ le nombre d'états de M et $|\Gamma|$ le nombre de symboles de son alphabet (de bande). Si M est en espace $f(n)$, le nombre de bandes différentes est $|\Gamma|^{f(n)}$ et le nombre de couples (position,état) différents sur la bande est $f(n).|Q|$.
En conséquence, le nombre de configurations différentes est donc de : $f(n).|Q|.|\Gamma|^{f(n)} = f(n).2^{O(f(n))} = 2^{O(f(n))}$. □

Théorème VI.5
PSPACE ⊆ EXPTIME

DÉMONSTRATION:

Une MT qui ne boucle pas avec $2^{O(f(n))}$ configurations différentes est au plus en temps $2^{O(f(n))}$. Ceci et le lemme précédent implique $\texttt{SPACE}(f(n)) \subseteq \texttt{TIME}(2^{O(f(n))})$. D'où **PSPACE** ⊆ **EXPTIME**. □

Donc, un programme qui utilise une taille de mémoire polynomiale s'exécute au pire en temps exponentiel.

4 Espace logarithmique

Pour l'instant nous avons toujours considéré que $f(n) \geq n$ à savoir, la bande doit au moins contenir l'entrée.

En revanche, il pourrait être intéressant de savoir ce qu'un décideur utilise comme mémoire **en plus** de l'entrée w. Il devrait alors être possible de passer en sub-linéaire.

On considère à partir de maintenant les exécutions sur la MT suivante à 2 bandes :

- la première bande est la bande d'entrée (contient l'entrée de la MT et est en lecture seule)
- la seconde bande est la bande de travail (en lecture/écriture)

Afin que :

- Seul l'espace utilisé sur la bande de travail est compté.
- Une **configuration** contient l'état de la MT, la bande de travail, les positions des pointeurs sur chaque bande.

Remarque : cette approche n'a pas de sens dans le cadre de la complexité temporelle (temps de lecture de la bande + accès séquentiel aux données).

Définition VI.5 (classe **L**)

L est la classe des langages qui sont décidables en espace logarithmique sur une MTD.
$\mathbf{L} = \texttt{SPACE}(\log n)$

Définition VI.6 (classe **NL**)

NL est la classe des langages qui sont décidables en espace logarithmique sur une MTND.
$\mathbf{NL} = \texttt{NSPACE}(\log n)$

Ces espaces nous intéressent à plusieurs titres :

- ils sont assez gros pour permettre la résolution de problèmes intéressants.
- ils sont robustes (indépendant du modèle de machines ou de la méthode de codage de l'entrée, pour peu qu'elle reste raisonnable).

Pour faire court, les langages à espace logarithmique sont ceux pour lesquels l'algorithme que le reconnaît peut fonctionner avec $O(n)$ variables binaires contenant $O(\log(n))$ (= un nombre fixe de variables numériques de $k.\log_2 n$ bits).
Le langage $A = \left\{0^k 1^k \mid k > 0\right\} \in \mathbf{L}$. En effet, après vérification de syntaxe (en $SPACE(1)$), utiliser deux variables n_0 et n_1 pour calculer et stocker le nombre de 0 et de 1 (en $2.\texttt{SPACE}(\log_2 n)$) pour une entrée de taille n). Comparer n_0 et n_1 et conclure.

Le langage $\texttt{PATH} \in \mathbf{NL}$. En effet, Utiliser deux variables : l'identificateur du sommet courant x et la longueur i du chemin de la manière suivante :

$N(\langle G, s, t\rangle) =$

```
1 Initialisation : x = s, i = 0, nSommets = nombre de sommets de G
2 tant que i < nSommets
3 |   choix non déterministe d'une arête (x, u) dans G
4 |   si u = t alors accepter sinon x = u
5 |   incrémenter i
6 rejeter
```

On utilise en espace 3 variables $\log_2 n$ bits et un algorithme non déterministe. Donc on est en $\texttt{NSPACE}(\log_2 n)$, donc dans **NL**.

De façon similaire à la question de savoir si **P** = **NP**, il se pose la question de savoir si **L** = **NL**.
Pour ce faire, on veut définir une **NL**-complétude et si l'un des langages *NL*-complet est dans *L*, alors on pourra conclure **L** = **NL**.
On a besoin de définir trois chosess :

- les fonctions calculables en espace logarithmique
- une réduction en espace logarithmique
- la **NL**-complétude

On utilisera esp.log. pour espace logarithmique.

Définition VI.7 (transducteur)

Un transducteur est une MT avec une bande d'entrée en lecture seule, une bande de travail et une bande de sortie en écriture seule.

Définition VI.8 (fonction calculable en espace logarithmique)

Une fonction $f(w)$ calculable en esp.log. est :

- un transducteur qui prend en entrée $\langle w \rangle$, avec $n = |\langle w \rangle|$.
- dont la bande de travail contient au plus $O(\log n)$ symboles,
- et qui renvoie la valeur stockée sur sa bande de sortie au moment où le transducteur s'arrête.

Définition VI.9 (réduction en espace logarithmique)

Un langage A est dit réductible en esp.log. à un langage B si il existe une réduction de A à B par une fonction calculable en esp.log..
On note : $A \leq_{\mathbf{L}} B$
Rappel : réduction = fonction calculable tq $\forall w, w \in A \Leftrightarrow f(w) \in B$.

Définition VI.10 (**NL**-complétude)

Un langage B est dit **NL**-complet si :

- $B \in \mathbf{NL}$
- $\forall A \in \mathbf{NL}$, $A \leq_{\mathbf{L}} B$, à savoir tout autre langage de **NL** peut se réduire à B.

Théorème VI.6

Si $A \leq_{\mathbf{L}} B$ et $B \in \mathbf{L}$ alors $A \in \mathbf{L}$.

Démonstration:
Pour décider A en esp.log., effectuer la transduction de A vers B (donc en esp.log.), décider B (en esp.log.). □

Corollaire VI.1

si n'importe quel langage **NL**-complet est dans **L**, alors **L** = **NL**

Démonstration:
si B est **NL**-complet alors $\forall A \in \mathbf{NL}$, $A \leq_{\mathbf{L}} B$.
L'existence de cette réduction implique que $\forall w, w \in A \Leftrightarrow f(w) \in B$.
Si $B \in \mathbf{L}$, le décideur de B permet donc de décider $f(A)$en espace L.
Donc, $A \in \mathbf{L}$ (et pour tout $A \in \mathbf{NL}$). D'où **L** = **NL**. □

Par ailleurs, on peut démontrer que :

Proposition VI.3

- PATH est **NL**-complet.
- $\overline{\texttt{PATH}}$ est **NL**-complet.

Ce qui pour conséquence que :

Théorème VI.7

NL = co**NL**

DÉMONSTRATION:

1. $\forall x \in \mathbf{NL}$, nous avons $x \leq_{\mathbf{L}} \texttt{PATH}$ puisque PATH est **NL**-complet. Alors, par la même fonction de réduction, nous avons $\overline{x} \leq_{\mathbf{L}} \overline{\texttt{PATH}}$. Donc, $\overline{x} \in \mathbf{NL}$ puisque $\overline{\texttt{PATH}} \in \mathbf{NL}$. En conséquence, $\mathbf{NL} \subseteq \texttt{co}\mathbf{NL}$.
2. $\forall x \in \texttt{co}\mathbf{NL}$, on a $\overline{x} \in \mathbf{NL}$. De façon similaire, $\overline{x} \leq_{\mathbf{L}} \texttt{PATH}$ et $x \leq_{\mathbf{L}} \overline{\texttt{PATH}}$. Encore une fois, puisque $\overline{\texttt{PATH}} \in \mathbf{NL}$, on a aussi $x \in \mathbf{NL}$. Donc, $\texttt{co}\mathbf{NL} \subseteq \mathbf{NL}$.

□

Corollaire VI.2

NL $\subseteq$ **P**

DÉMONSTRATION:

- **démonstration 1 :** le nombre de configuration d'un MTD sur une bande de taille $f(n)$ a au plus $c(n) = 2^{O(f(n))}$ configurations différentes. En eps.log., $f(n) = O(\log(n))$, d'où $\exists k$, $c(n) \leq 2^{log(n^k)} = n^k$. En conséquence, cette MTD est en temps $\texttt{TIME}(n^k) \subset \mathbf{P}$.
- **démonstration 2 :** PATH est **NL**-complet. Avec le même argument que la démonstration précédente, la réduction en esp.log. vers PATH est dans **P**. Comme $\texttt{PATH} \in \mathbf{P}$ et PATH **NL**-complet, tout langage de **NL** peut être décidé par PATH en temps polynomial.

□

Définition VI.11 (Fonction constructible en espace)

Une fonction $f(n) \geq n$, non décroissante, est constructible en espace si il existe une fonction calculable F qui calcule $F : 1^n \mapsto 1^{f(n)}$ en espace $O(f(n))$.

Théorème VI.8 (hiérarchie en espace)

Si g est constructible en espace et $f(n) = o(g(n))$, alors $\texttt{SPACE}(f(n)) \subsetneq \texttt{SPACE}(g(n))$.

Corollaire VI.3

NL $\subsetneq$ **PSPACE**

DÉMONSTRATION:

Par le théorème de hiérarchie en espace, on a **NL** $\subsetneq$ **NPSPACE**. Or **PSPACE** = **NPSPACE**. D'où **NL** $\subsetneq$ **PSPACE**. □

5 Synthèse des espaces de complexité

$$\mathbf{L} \subseteq \mathbf{NL} = \mathbf{coNL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME}$$

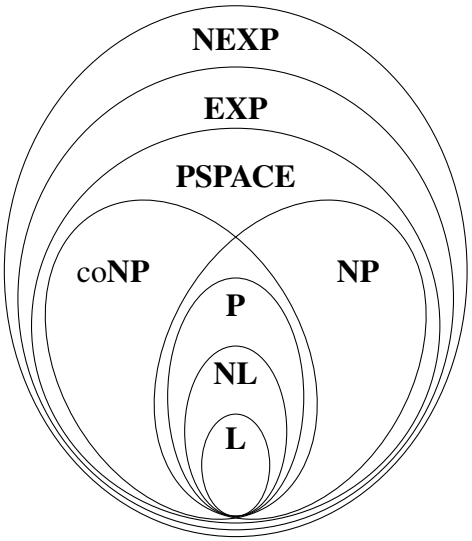
En conséquence,

- soit $\mathbf{coNL} \subsetneq \mathbf{P}$.
- soit $\mathbf{P} \subsetneq \mathbf{PSPACE}$.

Malheureusement, on ne sait pas laquelle des deux est vraie ou si les deux sont vraies (on pense qu'il s'agit de ce dernier cas).
En résumé,

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} = \mathbf{NPSPACE} \subseteq \mathbf{EXPTIME}$$

Donc, au moins une des inclusions est stricte.



Remarques :

- actuellement, on ne sait pas laquelle est stricte.
- beaucoup pensent qu'elles le sont toutes.