

Cours 4.2: Complexité en espace.

Olivier Bournez

bournez@lix.polytechnique.fr
LIX, Ecole Polytechnique

INF561

Algorithmes et Complexité

1

Au menu

Classes de complexité en espace

Relations entre espace et temps

Quelques résultats & résumé

2

Sous menu

Classes de complexité en espace

Mesure de l'espace d'un calcul

Classe PSPACE

Espace et structures non-finies

Espace non-déterministe

3

Taille d'une structure

- Considérons une structure du premier ordre sur la signature $(f_1, \dots, f_u, r_1, \dots, r_v)$.
- On rappelle que l'on dit qu'un emplacement (f, \mathfrak{M}) est *utile*, si f_i est un symbole de fonction dynamique, et son contenu $\llbracket (f, \mathfrak{M}) \rrbracket$ n'est pas **undef**.
- La *taille (mémoire)* de la structure \mathfrak{M} , notée $\text{size}(X)$ est le nombre d'emplacements utiles de X .

4

Espace de calcul

- Soit A un algorithme.
- A chaque entrée w est associée une exécution X_0, X_1, \dots, X_t du système de transitions associé à A où $X_0 = X[w]$ est un état initial qui code w , et chaque X_i est un état, et $X_{i+1} = \tau_A(X_i)$ pour tout i .
- Supposons que w soit acceptée, c'est-à-dire qu'il existe un entier t avec X_t terminal :
 - ▶ Rappel : le *temps de calcul* est l'entier t .
 - ▶ L'*espace (mémoire) de calcul sur l'entrée w* , noté $\text{SPACE}(A, w)$, est défini comme

$$\text{SPACE}(A, w) = \max_{0 \leq i \leq t} \text{size}(X_i)$$

5

- Soit A un algorithme qui termine sur toute entrée.

- L'*espace (mémoire)* de l'algorithme A est la fonction $f : \mathbb{N} \rightarrow \mathbb{N}$ telle que

$$f(n) = \max_{|w|=n} \text{SPACE}(A, w).$$

6

- Soit $t : \mathbb{N} \rightarrow \mathbb{N}$ une fonction.
- On définit la classe

$$\text{SPACE}(t(n)) = \{L \mid L \text{ est un langage décidé par un algorithme en espace } \mathcal{O}(t(n))\}.$$

7

Conventions

- Pour pouvoir parler d'espace $f(n) \leq n$,
- ... on ne compte pas l'entrée et la sortie dans l'espace :
 - ▶ l'entrée est accessible en *lecture uniquement*,
 - ▶ et la sortie en *écriture uniquement*

8

Classes de complexité en espace

Mesure de l'espace d'un calcul

Classe PSPACE

Espace et structures non-finies

Espace non-déterministe

- $$\text{PSPACE} = \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k).$$

- $$\text{LOGSPACE} = \text{SPACE}(\log(n)).$$

Classes de complexité en espace

Mesure de l'espace d'un calcul

Classe PSPACE

Espace et structures non-finies

Espace non-déterministe

- Sur \mathbb{N} , tout algorithme peut se coder par une machine à 2 compteurs et donc en espace $\mathcal{O}(1)$ (au prix d'une perte de temps qui peut être exponentielle).
- Sur la structure $(\mathbb{R}, 0, 1, +, -, *, =, <)$ tout algorithme peut être réalisé par un algorithme qui fonctionne en espace constant (au prix d'une perte de temps qui peut être exponentielle).

On ne parlera que des structures finies dans la suite

Classes de complexité en espace

Mesure de l'espace d'un calcul

Classe PSPACE

Espace et structures non-finies

Espace non-déterministe

- Soit M un alphabet fini.
- Un problème $L \subset M^*$ est dans NPSPACE s'il existe un polynôme $p : \mathbb{N} \rightarrow \mathbb{N}$ et un problème A (appelé *vérificateur pour L*) tels que
 - ▶ pour tout mot w ,
 $w \in L$ si et seulement si $\exists u \in M^*$ tel que $\langle w, u \rangle \in A$,
 - ▶ et tels que déterminer si $\langle w, u \rangle \in A$ admette un algorithme en espace $p(|w|)$.

- Soit M un alphabet fini.
- **Théorème**
Un problème $L \subset M^$ est dans NPSPACE si et seulement s'il est décidé en espace non-déterministe polynomial.*

Classes de complexité en espace

Relations entre espace et temps

Quelques résultats & résumé

Sous menu

Relations entre espace et temps

Relations triviales

- Temps non-déterministe vs déterministe
- Temps non-déterministe vs espace
- Espace non-déterministe vs temps
- Espace non-déterministe vs espace déterministe
- Espace logarithmique non-déterministe

17

Relations triviales

■ Théorème

$$\text{SPACE}(f(n)) \subset \text{NSPACE}(f(n)).$$

■ Théorème

$$\text{TIME}(f(n)) \subset \text{SPACE}(f(n)).$$

18

Preuve

- Un problème déterministe étant un problème non-déterministe particulier.
- Un algorithme écrit au plus un nombre fini d'emplacements (mémoire) à chaque étape.

19

Sous menu

Relations entre espace et temps

Relations triviales

Temps non-déterministe vs déterministe

- Temps non-déterministe vs espace
- Espace non-déterministe vs temps
- Espace non-déterministe vs espace déterministe
- Espace logarithmique non-déterministe

20

Théorème

Pour tout langage de $\text{NTIME}(f(n))$, il existe un entier c tel que ce langage soit dans $\text{TIME}(c^{f(n)})$. Si l'on préfère :

$$\text{NTIME}(f(n)) \subset \bigcup_{c \in \mathbb{N}} \text{TIME}(c^{f(n)})$$

21

Preuve

- Soit L un problème de $\text{NTIME}(f(n))$.
- On sait qu'il existe un problème A tel que pour déterminer si un mot w de longueur n est dans L , il suffit de savoir s'il existe $u \in M^*$ avec $\langle w, u \rangle \in A$, ce dernier test pouvant se faire en temps $f(n)$, où $n = |w|$.
- On peut se limiter aux mots u de longueur $f(n)$.
- Tester si $\langle w, u \rangle \in A$ pour tous les mots $u \in M^*$ de longueur $f(n)$ se fait facilement en temps $\mathcal{O}(c^{f(n)} + \mathcal{O}(f(n))) = \mathcal{O}(c^{f(n)})$, où $c > 1$ est le cardinal de M .

22

Remarque

- Pour écrire un mot u de longueur $f(n)$, il faut que f soit calculable.
- En fait, on se limite aux bornes sur le temps et l'espace qui sont de complexité propre :
 - ▶ on suppose que la fonction $f(n)$ est non-décroissante, c'est-à-dire que $f(n+1) \geq f(n)$,
 - ▶ et qu'il existe un algorithme qui prend en entrée w et qui produit en sortie $1^{f(n)}$ en temps $\mathcal{O}(n + f(n))$
 - ▶ et en espace $\mathcal{O}(f(n))$, où $n = |w|$.
- Toutes les fonctions usuelles non-décroissantes, comme $\log(n)$, n , n^2 , \dots , $n \log n$, $n!$ vérifient ces propriétés.
- Cette hypothèse sera implicite dans la suite.

23

Sous menu

Relations entre espace et temps

Relations triviales

Temps non-déterministe vs déterministe

Temps non-déterministe vs espace

Espace non-déterministe vs temps

Espace non-déterministe vs espace déterministe

Espace logarithmique non-déterministe

24

Temps non-déterministe vs espace

Théorème

$$\text{NTIME}(f(n)) \subset \text{SPACE}(f(n)).$$

25

Preuve

- On utilise exactement le même principe que dans la preuve précédente, si ce n'est que l'on parle d'espace.
- Soit L un problème de $\text{NTIME}(f(n))$.
- On sait qu'il existe un problème A tel que pour déterminer si un mot w de longueur n est dans L , il suffit de savoir s'il existe $u \in M^*$ de longueur $f(n)$ avec $\langle w, u \rangle \in A$.
- On utilise un espace $\mathcal{O}(f(n))$ pour générer un à un les mots $u \in M^*$ de longueur $f(n)$ puis on teste pour chacun si $\langle w, u \rangle \in A$, ce dernier test se faisant en temps $f(n)$, donc espace $f(n)$.
- Le même espace pouvant être utilisé pour chacun des mots u , au total cela se fait au total un espace $\mathcal{O}(f(n))$ pour générer les $u + \mathcal{O}(f(n))$ pour les tests, soit $\mathcal{O}(f(n))$.

26

Sous menu

Relations entre espace et temps

Relations triviales

Temps non-déterministe vs déterministe

Temps non-déterministe vs espace

Espace non-déterministe vs temps

Espace non-déterministe vs espace déterministe

Espace logarithmique non-déterministe

27

Lien fondamental avec le problème REACH

- Le problème de décision REACH,
 - ▶ on se donne
 1. un graphe orienté $G = (V, E)$,
 2. deux sommets u et v ,
 - ▶ et on cherche à décider s'il existe un chemin entre u et v dans G .
- jouera un grand rôle.

28

Lien fondamental avec le problème REACH

- A tout algorithme A (déterministe ou non) sur la structure $\Sigma = (M, f_1, \dots, f_s, r_1, \dots, r_r)$ est associé un système de transition, qui peut être vu comme un graphe.
- chaque configuration X peut se décrire par un mot $[X]$ sur l'alphabet M longueur $O(\text{size}(X))$ qui en décrit les emplacements utiles :
 - ▶ si on fixe l'entrée w de longueur n , pour un calcul en espace $f(n)$, il y a donc moins de $O(c^{f(n)})$ sommets dans ce graphe G_w , où $c > 1$ est le cardinal de l'alphabet M .
- On peut construire un circuit C_w de taille $O(f(n))$ tel que $C_w([X], [X']) = 1$ si et seulement si X' est une configuration successeur de la configuration X .
- Un mot w est accepté par l'algorithme A si et seulement s'il y a un chemin dans ce graphe G_w entre l'état initial $X[w]$ codant l'entrée w , et un état acceptant X^* , que l'on peut supposer unique.

29

Lien fondamental avec le problème REACH

- Autrement dit :

$w \in L$ si et seulement si $\langle G_w, X[w], X^* \rangle \in \text{REACH}$

- On va écrire ce que l'on sait sur le problème REACH.

30

- REACH se résout par exemple en temps et espace $O(n^2)$, où n est le nombre de sommets, par un parcours en profondeur.

31

Conséquence

Théorème

Si $f(n) \geq \log n$, alors

$$\text{NSPACE}(f(n)) \subset \bigcup_{c \in \mathbb{N}} \text{TIME}(c^{f(n)}).$$

32

Sous menu

Relations entre espace et temps

Relations triviales

Temps non-déterministe vs déterministe

Temps non-déterministe vs espace

Espace non-déterministe vs temps

Espace non-déterministe vs espace déterministe

Espace logarithmique non-déterministe

33

Proposition

$\text{REACH} \in \text{SPACE}(\log^2 n)$.

34

Preuve

- Soit $G = (V, E)$ le graphe orienté en entrée.
- Étant donné deux sommets x et y de ce graphe, et i un entier, on note $\text{CHEMIN}(x, y, i)$ si et seulement s'il y a un chemin de longueur inférieure à 2^i entre x et y .
- On a $a < G, u, v > \in \text{REACH}$ si et seulement si $\text{CHEMIN}(u, v, \log(n))$, où n est le nombre de sommets.
- L'astuce est de calculer $\text{CHEMIN}(x, y, i)$ récursivement en observant que l'on a $\text{CHEMIN}(x, y, i)$ si et seulement s'il existe un sommet intermédiaire z tel que $\text{CHEMIN}(x, z, i-1)$ et $\text{CHEMIN}(z, y, i-1)$. On teste alors à chaque niveau de la récursion chaque sommet possible z .
- Pour représenter chaque sommet, il faut $O(\log(n))$ bits. Cela donne une récurrence de profondeur $\log(n)$, chaque étape de la récurrence nécessitant uniquement de stocker un triplet x, y, i et de tester chaque z de longueur $O(\log(n))$. Au total, on utilise donc un espace $O(\log(n)) + O(\log(n)) = O(\log^2(n))$.

35

Conséquence

Théorème (Savitch)

Si $f(n) \geq \log(n)$, alors

$\text{NSPACE}(f(n)) \subset \text{SPACE}(f(n)^2)$.

36

Preuve

- On utilise l'algorithme précédent pour déterminer s'il y a un chemin dans le graphe G_w entre $X[w]$ et X^* .
- On remarquera que l'on a pas besoin de construire explicitement le graphe G_w mais que l'on peut utiliser l'algorithme précédent à la volée.

37

Corollaire

$PSPACE = NPSPACE.$

38

Sous menu

Relations entre espace et temps

Relations triviales

Temps non-déterministe vs déterministe

Temps non-déterministe vs espace

Espace non-déterministe vs temps

Espace non-déterministe vs espace déterministe

Espace logarithmique non-déterministe

39

- On note

$NLOGSPACE = NSPACE(\log(n)).$

- **Théorème**

$REACH \in NLOGSPACE.$

40

Preuve

- Pour déterminer s'il y a un chemin entre u et v dans un graphe G , on devine le chemin arc par arc.
- Cela nécessite uniquement de garder le sommet que l'on est en train de visiter en plus de u et de v .

41

Technique du comptage inductif

- Soit C une classe de complexité. On note $\text{co-}C$ pour la classe des langages dont le complémentaire est dans la classe C .
- **Théorème**
Le problème REACH est aussi dans coNLOGSPACE .

42

L'idée

- Supposons que l'on veuille savoir si t n'est pas accessible à partir d'un sommet s dans un graphe G à n sommets.
- Supposons que l'on connaisse le nombre c sommets accessibles à partir d'un sommet s .
- L'algorithme suivant résout le problème :

```
d ← 0
for u sommet de G do
  ran ← guess r ∈ {0,1}
  if ran then
    [ suivre de façon non déterministe un chemin
      de longueur n-1 à partir de s et si aucun
      sommet rencontré sur ce chemin est u rejeter ]
    if u = t then rejeter
    d ← d + 1
  endif
endifor
if d ≠ c then rejeter
else accepter
```

43

Calculer c ?

- Soit c_i le nombre de sommets qui sont atteignables à partir de s par un chemin de longueur inférieure ou égale à i .
- $c = c_{n-1}$ se calcule récursivement.

```
c0 ← 1
for i ← 0 to n-2
  ci+1 ← 0
  d ← 0
  for v sommet de G do
    for u sommet de G do
      ran ← guess x ∈ {0,1}
      if ran then
        [ suivre de façon non déterministe un chemin
          de longueur i à partir de s et si aucun
          sommet rencontré est u rejeter ]
        d ← d + 1
        if (u,v) est un arc de G then
          ci+1 ← ci+1 + 1
          [ sortir de la boucle for u sommet de G
            pour passer au prochain v ]
        endif
      endif
    endfor
  endfor
  if d ≠ ci then rejeter
endifor
c ← cn-1
```

44

Conséquence

Théorème

$\text{NLOGSPACE} = \text{coNLOGSPACE}$.

45

Preuve

- Il suffit de montrer que $\text{coNLOGSPACE} \subset \text{NLOGSPACE}$.
- Pour décider si un mot w doit être accepté par un langage de coNLOGSPACE , on peut utiliser l'algorithme non-déterministe précédent qui utilise un espace logarithmique pour déterminer s'il existe un chemin entre $X[w]$ et X^* dans le graphe G_w .

46

Au menu

Classes de complexité en espace

Relations entre espace et temps

Quelques résultats & résumé

47

Sous menu

Quelques résultats & résumé

Théorèmes de hiérarchie

Classes de complexité usuelles

48

Théorèmes de hiérarchie

Théorème (Théorème de hiérarchie)

Pour toute fonction $f : \mathbb{N} \rightarrow \mathbb{N}$ constructible en espace, il existe un langage L qui est décidable en espace $O(f(n))$ mais pas en espace $o(f(n))$.

Autrement dit :

Théorème (Théorème de hiérarchie)

Soient $f, f' : \mathbb{N} \rightarrow \mathbb{N}$ des fonctions constructibles en espace telles que $f(n) = o(f'(n))$. Alors l'inclusion $\text{SPACE}(f) \subset \text{SPACE}(f')$ est stricte.

Se généralise aux classes non-déterministes.

49

Preuve

- On considère le langage (très artificiel) L qui est décidé par l'algorithme B suivant :
 - ▶ sur une entrée w de taille n , B calcule $f(n)$ en utilisant la constructibilité en espace de f , et réserve un espace $f(n)$ pour la simulation qui va venir.
 - ▶ Si w n'est pas de la forme $\langle A \rangle 10^n$, pour un certain algorithme A , alors l'algorithme B rejette.
 - ▶ Sinon, B simule l'algorithme A sur le mot w pendant au plus $2^{f(n)}$ étapes pour déterminer si A accepte en ce temps avec un espace inférieur à $f(n)$.
 - si A accepte en ce temps et cet espace, alors B rejette;
 - sinon B accepte.
- L est dans $\text{SPACE}(f(n))$:
 - ▶ plus concrètement, si A utilise un espace $g(n) \leq f(n)$, alors B utilise un espace au plus $dg(n)$ pour une constante d .
- Supposons que L soit décidé par un algorithme A en espace $g(n)$ avec $g(n) = o(f(n))$. Il doit exister un entier n_0 tel que pour $n \geq n_0$, on ait $dg(n) < f(n)$.

50

- Considérons ce qui se passe lorsque B est lancé sur l'entrée $\langle A \rangle 10^{n_0}$.
 - ▶ Puisque cette entrée est de taille plus grande que n_0 , B répond l'inverse de l'algorithme A sur la même entrée.
 - ▶ Donc l'algorithme A ne décide pas L , ce qui mène à une contradiction.

51

Conséquences

Théorème

$\text{NLOGSPACE} \subsetneq \text{PSPACE}$.

52

Preuve

- La classe NLOGSPACE est complètement incluse dans $\text{SPACE}(\log^2 n)$ par le théorème de Savitch.
- Hors cette dernière est un sous-ensemble strict de $\text{SPACE}(n)$, qui est inclus dans PSPACE.

53

Sur le même principe

Soit

$$\text{EXPSPACE} = \bigcup_{c \in \mathbb{N}} \text{SPACE}(2^{n^c}).$$

Théorème

$$\text{PSPACE} \subsetneq \text{EXPSPACE}.$$

54

Sous menu

Quelques résultats & résumé

Théorèmes de hiérarchie

Classes de complexité usuelles

55

Classes les plus usuelles

$$\text{LOGSPACE} = \text{SPACE}(\log n)$$

$$\text{NLOGSPACE} = \text{NSPACE}(\log n)$$

$$\text{P} = \bigcup_{c \in \mathbb{N}} \text{TIME}(n^c)$$

$$\text{NP} = \bigcup_{c \in \mathbb{N}} \text{NTIME}(n^c)$$

$$\text{PSPACE} = \bigcup_{c \in \mathbb{N}} \text{SPACE}(n^c)$$

56

Résumé

- On obtient :

$\text{LOGSPACE} \subset \text{NLOGSPACE} \subset \text{P} \subset \text{NP} \subset \text{PSPACE}$.

- $\text{NLOGSPACE} \subsetneq \text{PSPACE}$ mais on ne sait pas lesquelles des inclusions strictes intermédiaires sont vraies.
- Les classes déterministes sont closes par complément.
- $\text{NLOGSPACE} = \text{coNLOGSPACE}$.