Classes de complexité standards

Classes complexité: préliminaires

- Une fonction $f : \mathbb{N} \rightarrow \mathbb{N}$ est une fonction de complexité propre si
  - f est croissante.
  - Il existe une MT qui retourne $O^{f(n)}$ sur une entrée quelconque de longueur n en temps $\Theta(n)$.
- Dorénavant considère uniquement des fonctions de complexité propre.
- **Notations:**
  - $O(f) = \{g | \exists c \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \leq c * f(n)\}$
  - $o(f) = \{g | \forall c \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \leq c * f(n)\}$
  - $\Omega(f) = \{g | \exists c \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq c * f(n)\}$
  - $\Theta(f) = O(f) \cap \Omega(f)$

Classes complexité

- Soit $f : \mathbb{N} \to \mathbb{N}$ une fonction. Σ un alphabet.
  - $TIME(f) = \{L \subset \Sigma^*, \exists MT\ M\ tq\ L$ accepté par M en temps $f\}$
  - $NTIME(f) = \{L \subset \Sigma^*, \exists MT\ ND\ M\ tq\ L$ accepté par M en temps $f\}$
  - $SPACE(f) = \{L \subset \Sigma^*, \exists MT\ M\ tq\ L$ accepté par M en espace $f\}$
  - $NSPACE(f) = \{L \subset \Sigma^*, \exists MT\ ND\ M\ tq\ L$ accepté par M en espace $f\}$

Exemple: L est dans $TIME(f)$ ssi il existe une machine de Turing M telle que sur toute entrée $w \in \Sigma^*$, si on pose $n = |w|$,

1. **si $w \in L$, M atteint un état d'acception en temps $t \leq f(n)$**
2. **si $w \notin L$, M n'accepte pas w.**

Théorème de l'accélération

- **Théorème:** $L \in SPACE(f)$. Alors $\forall \epsilon > 0$, $L \in SPACE(f')$ avec $f'(n) = \epsilon f(n)$.

  Démonstration. Pour $\epsilon \geq 1$, il n'y a rien à prouvé puisque $f(n) \leq f'(n)$. Supposons $0 < \epsilon < 1$. On simule M à k-rubans par une machine M' à k rubans. Idée: coder plusieurs symboles de M à la fois. Soit $m \in \mathbb{N}$ dépendant de M et ϵ. Alphabet de M' défini par $\Sigma' = \Sigma \cup \Sigma^m$. M' commence par lire entrée et coder en m-uplet: besoin aucun espace supplémentaire que celui de l'entrée. Les rubans de M' contiennent dorénavant plus que des m-uples. M' simule chaque étape de M sur ces m-uples. Espace nécessaire: $\lceil f(n)/m \rceil$. Choisir m tel que $m\epsilon \geq 2$ pour obtenir $\lceil f(n)/m \rceil < f(n)/m + 1 \leq \epsilon f(n)/2 + 1$, et donc $\leq \epsilon f(n)$ pour n assez grand.

Théorème de l'accélération

- **Théorème:** $L \in TIME(f)$. Supposons $n \in o(f(n))$. Alors $\forall \epsilon > 0$, $L \in TIME(f')$ avec $f'(n) = \epsilon f(n)$.

Démonstration. On simule M à k-rubans par une machine M' à k rubans pour $k \geq 2$, à 2-rubans pour $k = 1$. Idée: coder plusieurs symboles de M à la fois comme dans la preuve précédente.
Le codage en m-uplet de l'entrée peut se faire en temps n en écrivant au fur et à mesure le codage de l'entrée sur le ruban numéro 2 (pour garantir un temps linéaire).
Il faut $\lceil n/m \rceil$ étapes pour remettre la tête de lecture sur le m-uplet correspondant à la première lettre.
On simule ensuite les étapes de M de telle sorte que 6-étapes de M' correspondent à m-étapes de M: dans chaque bloc de ces 6-étapes, chaque tête de M' fait $\leftarrow, \rightarrow, \rightarrow, \leftarrow$, en se souvenant des symboles lus. M' peut alors complètement prévoir les m-prochains mouvements de M, et mettre à jour ses rubans en au plus 2 pas.

Temps total: $T = n + \lceil n/m \rceil + \lceil f(n)/m \rceil$. Pour $m \geq 2$, cela est $\leq 2n + 6 + 6/m f(n)$. Pour n assez grand, $2n + 6 \leq 3n$. Puisque $n \in o(f(n))$ pour n assez grand, $3n \leq \epsilon f(n)$. On obtient $T \leq \epsilon/2 f(n) + 6/m f(n)$. Le choix $m \geq 12/\epsilon$ garantie $T \leq \epsilon f(n)$.

Sur le nombre de rubans

- **Théorème:** L reconnu par MT M avec k-rubans en temps $t(n)$, alors L reconnu par une MT avec un seul ruban de travail en temps $O(t(n)^2)$.

  Démonstration: chacun des rubans de travail utilise au plus $t(n)$ cases. On simule l'évolution des configurations $C_i = (q, w, w_2, \ldots, w_k)$ de M sur l'entrée w. C_i est codé par le mot $q\$B^{i_2}w_2B^{j_2}\$\ldots\$B^{i_k}w_1B^{j_k}$ où $\$$ est une nouvelle lettre, où les i_l et j_l sont choisis pour que les mots $B^{i_l}w_lB^{j_l}\$$ aient une longueur constante $t(n)$. Il faut un temps $O(t(n))$ pour calculer le codage de C_{i+1} à partir de celui de C_i (il suffit de le parcourir et de mettre à jour). La simulation des $t(n)$ instructions se fait donc en temps total $O(t(n)^2)$.

- **Théorème:** L reconnu par MT M avec k-rubans de travail en temps $t(n)$, alors L reconnu par une MT avec deux rubans de travail en temps $O(t(n)logt(n))$.

  Démonstration. Voir votre bibliothèque préférée.

Classes de complexité standard

- $L = \cup_c SPACE(c \log n) = SPACE(\log n)$
- $NL = \cup_c NSPACE(c \log n) = NSPACE(\log n)$
- $P = \cup_c TIME(n^c)$
- $NP = \cup_c NTIME(n^c)$
- $PSPACE = \cup_c SPACE(n^c)$

Déterminisme vs Non-déterminisme

- **Théorème:** $TIME(f) \subset NTIME(f)$.
- **Théorème:** $SPACE(f) \subset NSPACE(f)$.

  Démonstration. Une machine déterministe est une machine non-déterministe particulière.
- **Théorème:** $TIME(f) \subset SPACE(f)$.

  Démonstration. On peut au plus utiliser une nouvelle case du ruban à chaque étape, et donc en temps $f(n)$ on peut utiliser au plus un espace $f(n)$.

Déterminisme vs Non-déterminisme

- **Théorème:** $NTIME(f) \subset \cup_c TIME(c^{f(n)})$.

Démonstration. Soit M une machine non-deterministe. Pour chaque $(q, a_1, \ldots, a_k) \in Q \times \Gamma^k$ on peut considérer le cardinal de $\delta(q, a_1, \ldots, a_k)$. Note d le maximum de ce cardinal quand $q, a_1, \ldots, a_k$ varient. (d est appelé le degré de non-déterminisation).
Une séquence de t choix non-deterministes se code par une suite de t entiers dans intervalle $0, \ldots, d-1$.
On construit une machine M' qui sur un mot w considère toutes les suites de $f(n)$ tels entiers, et qui pour chaque suite simule M sur w pour les choix non-déterministes correspondants. M' accepte si il détecte qu'un des calculs de M sur w est acceptant.

Temps de la simulation: $O(d^{f(n)} f(n)) \leq O(d^{f(n)} 2^{f(n)}) = O((2d)^{f(n)})$.

Temps Non-déterministe vs Espace

- **Théorème:** $NTIME(f) \subset SPACE(f)$.

Démonstration. La machine M' de la preuve précédente utilise un espace $O(f(n))$ pour générer un à un les suites de $f(n)$ entiers dans intervalle $0, \ldots, d-1$ en comptant en base d. Elle a besoin ensuite uniquement d'un espace $f(n)$ pour simuler M sur chacune des suites (le même espace est (ré)utilisé pour chacune des suites).

Cela donne un espace total $O(f(n))$.

Problème REACH

- Problème ***REACH*:**
  - **Instance:** un graphe orienté $G = (V, E)$,deux sommets $u, v \in V$
  - **Question:** Existe-t'-il un chemin entre u et v dans G?
- Il existe un algorithme "raisonnable" (polynômial): parcours de graphe en profondeur, largeur,. . . en temps $O(n^2)$ où n est le nombre de sommets.

Algorithme pour REACH

- **Parcours de graphe:**
  1. $S = \{u\}$.
  2. Pour $i = 1$ à $|V|$ faire Marque[i]:=0;
  3. Tant que $S \neq \emptyset$
     (a) Choisir $i \in S$; $S := S - \{i\}$; $Marque[i] := 1$.
     (b) Pour tous les j avec $(i, j) \in E$
        i. Si $Marque[j] = 0$ alors Ajouter j à S.
  4. Accepter si $Marque[v] = 1$, rejeter sinon.

- **Cet Algorithme en temps et espace mémoire polynômial $O(n^2)$ avec $n = |V|$.**

Espace non-déterministe vs Temps

- **Théorème:** $NSPACE(f) \subset \cup_c TIME(c^{f(n)+log(n)})$.

Démonstration. Définit graphe des configurations $G(M, w)$:

1. **sommets = toutes les configurations possibles de M sur mot w.**
2. **Arc de C vers C' ssi $C \vdash C'$.**

Sans perte de généralité, on peut supposer qu'une machine de Turing M accepte toujours avec ses rubans vides. Notons C_{final} la configuration $(q_a, \#, \ldots, \#)$ correspondante. Rapellons que $C[w] = (q_0, \#w, \#, \ldots, \#)$.
On a $w \in L$ ssi $(G(M, w), C[w], C_{final}) \in Reach$.
Taille de $G(M, w)$ est $S \leq |Q| * (n + 1) * (|\Gamma|^{f(n)})^{2k-2}$.
(En effet, une configuration se note $C = (q, u_1 \# v_1, \ldots, u_k \# v_k)$. on a $|Q|$ choix possibles pour q, u_1 et v_1 sont toujours tels que $u_1 v_1 = w$ sur le 1ier ruban en lecture seulement, seul la position de la tête de lecture sur le 1ier ruban change, soit $n + 1$ possibilités. Pour les $k - 1$ autres rubans, u_i, v_i sont des mots sur Γ de longueur $\leq f(n)$).
On a donc $S \leq |Q| 2^{\log(n+1)} (|\Gamma|^{2k-2})^{f(n)}$ qui est $\leq O(d^{f(n)+\log n})$ pour une certaine constante d et pour n assez grand.

Il suffit alors d'utiliser un algorithme qui résoud $REACH$ par exemple en temps quadratique pour obtenir un algorithme en $O(d^{2(f(n)+\log n)}) = O((d^2)^{f(n)+\log n})$.
Choisir $c = d^2$.

Remarque: la machine ne construit pas explicitement $G(M, w)$: elle décide au cas par cas, si arc entre une configuration et une autre.

Espace non-déterministe vs Espace dé

- **Théorème:** $REACH \in SPACE(log^2 n)$.

  Démonstration. Soit $G = (V, E)$ **le graphe orienté en entrée. On définit relation** $CHEMIN(x, y, i)$ **vrai ssi il y a un chemin entre** x **et** y **de longueur** $\leq 2^i$**.** $(G, u, v) \in REACH$ **ssi** $CHEMIN(u, v, logn)$**. On a** $CHEMIN(x, y, i)$ **ssi** $\exists z, CHEMIN(x, z, i-1) \wedge CHEMIN(z, y, i-1)$**. Pour représenter** x, y, i, z **il faut** $O(logn)$ **bits. Récurrence de profondeur** $logn$**. Chaque étape examine triplet** (x, y, i)**. Puisque longueur triplet** $\leq 3logn$**, simule tout en espace** $O(log^2 n)$**.**

- Théorème Savitch: Si $f(n) \geq \log n$, alors $NSPACE(f) \subset SPACE(f^2)$.

  Démonstration. Utiliser algorithme précédent pour décider si $(G(M,w), C_{initial}, C_{final}) \in Reach$**.** *Attention: la machine ne doit pas construire explicitement* $G(M,w)$*, car celui-ci peut être de taille trop importante. Mais ok si la machine décide au cas par cas, si arc.*

- Corollaire: $PSPACE = \cup_c NSPACE(n^c)$.

  $Dmonstration : (n^c)^2 = n^{2c}$.

Problème REACH

- **Théorème:** $REACH \in NL$.

  Démonstration. Deviner le chemin entre u et v arc par arc. Comme on garde uniquement ce qu'on est en train de vérifier, on reste en espace $3 * logn$.

Problème REACH

- Théorème: $REACH \in coNL$.

Démonstration: exercice* (technique du comptage inductif (cf slides)).

- Théorème: Si $f(n) \geq \log n$, alors $NSPACE(f) = co - NSPACE(f)$

Démonstration: Soit $L \in NSPACE(f)$. Pour déterminer si $w \in L$, utiliser l'algorithme précédent pour déterminer si $(G(M,w), C_{initial}, C_{final}) \in Reach$.

Classes de complexité

- **Théorème:** $L \subset NL \subset P \subset NP \subset PSPACE$.

  Démonstration. Chaque inclusion est un corollaire immédiat des résultats précédents.

- **Théorème:** Les classes déterministes sont closes par complément.

  Démonstration. Inverser l'état d'acceptation et de rejet.

Théorèmes de Hiérarchie

- **Théorème:** Soit $f : \mathbb{N} \to \mathbb{N}$ fonction totale récursive non-décroissante. $\exists L \notin TIME(f)$. $\exists L \notin SPACE(f)$.

  Démonstration. Considèrons une énumération $(M_i)_i$ des machines de Turing, et une une énumération $(w_i)_i$ des mots sur Σ^*.
  Considérons $L = \{w_i | M_i$ n'accepte pas w_i en temps $f(|w_i|)\}$.
  L est reconnu par une certaine machine de Turing: il suffit de construite une machine de Turing qui sur l'entrée w, énumère les mots w_i pour $i = 1, 2, \ldots$ jusqu'à trouver i tel que $w_i = w$, puis simule au plus $f(|w_i|)$ pas de M_i pour décider si $w_i \in L$ ou non.
  Supposons par l'absurde que L soit reconnu par une certaine machine M_{i_0} en temps $f(n)$.

  - **Supposons $w_{i_0} \in L$. Cela implique w_{i_0} accepté par M_{i_0} en temps $f(|w_{i_0}|)$, donc que w_{i_0} n'appartient pas à L: absurde.**
  - **Supposons $w_{i_0} \notin L$. Cela implique w_{i_0} rejetté par M_{i_0} en temps $f(|w_{i_0}|)$, donc w_{i_0} appartient à L: absurde.**

  X

Théorèmes de Hiérarchie: espace

- **Théorème:** Soient $f, f' : \mathbb{N} \to \mathbb{N}$ avec $f = o(f')$. $SPACE(f')$ contient un langage qui n'est pas dans $SPACE(f)$.

Démonstration. Soit $(M_i)_i$ une énumération des machines de Turing.
On considère le langage L inclus dans $\{M_i 1 O^k | i, k \geq 0\}$ reconnu par la machine M qui sur l'entrée w de longueur n,

1. **calcule $f'(n)$,**
2. **rejette w s'il n'est pas de la forme $M_i 1 O^k$,**
3. **sinon simule M_i sur l'entrée w sur l'espace $f'(n)$,**
   (a) **rejette ssi M_i l'accepte dans cet espace.**

Ce langage est reconnu en espace $O(f'(n))$ car il est reconnu par la machine M précédente qui fonctionne en espace $O(f'(n))$.
Supposons qu'il soit dans $SPACE(f)$. Il reconnu par une certaine machine M_{i_0} en espace $c.f(n)$ pour une constante c. Soit n_0 tel que $cf(n) \leq f'(n)$ pour tout $n \geq n_0$. On considère le mot $w = M_{i_0} 1 O^k$ pour un $k \geq n_0$.

- **Si $w \in L$, puisque M_{i_0} décide L en espace $f(|w|)$, M_{i_0} l'accepte en espace $f(|w|)$, donc aussi en espace inférieur à $f'(|w|)$ donc $w \notin L$ par définition de L. Absurde.**
- **Si $w \notin L$, puisque M_{i_0} décide L en espace $f(|w|)$, M_{i_0} le rejette en espace $f(|w|)$, donc aussi en espace inférieur à $f'(|w|)$ donc $w \in L$ par définition de L. Absurde.**

Corollaires

- **Théorème:** $NL \subsetneq PSPACE$

Démonstration.

$NSPACE(c \log n) \subset SPACE(c^2 \log^2 n) \subset SPACE(\log^2 n) \subsetneq SPACE(n)$.

- **Théorème:** $PSPACE \subsetneq EXPSPACE$

Définition. $EXPSPACE = \cup_c SPACE(2^{cn})$.

Démonstration. $SPACE(n^c) \subset SPACE(n^{logn}) \subsetneq SPACE(2^n)$.

Théorèmes de Hiérarchie: temps

- **Théorème:** Soient $f, f' : \mathbb{N} \to \mathbb{N}$ avec $f = o(f'/\log n)$. $TIME(f')$ contient un langage qui n'est pas dans $TIME(f)$.

Démonstration. Même principe que pour l'espace. Le log vient du besoin éventuel de simuler k-rubans par 2-rubans.

Corollaires

- **Théorème:** $P \subsetneq EXPTIME$.

  Définition. $EXPTIME = \cup_c TIME(2^{cn})$.

  Démonstration. $TIME(n^c) \subset TIME(n^{logn}) \subsetneq TIME(2^n)$.