

Complexité algorithmique

(survol des classes de complexité usuelles)

Arnaud Casteigts

17 octobre 2023

Algorithmique (12X001)

Complexité algorithmique ?

Étude des ressources nécessaires pour résoudre un **problème**.

Problème $\Pi : \{0, 1\}^* \rightarrow \{0, 1\}^*$

Met en relation une **entrée** (ou **instance**) avec une **sortie**.

Résoudre un problème $\Pi =$ savoir calculer $\Pi(x)$ pour toute instance $x \in \{0, 1\}^*$ (ou d'un sous-domaine prédéfini).

Complexité algorithmique ?

Étude des ressources nécessaires pour résoudre un **problème**.

Problème $\Pi : \{0, 1\}^* \rightarrow \{0, 1\}^*$

Met en relation une **entrée** (ou **instance**) avec une **sortie**.

Résoudre un problème $\Pi =$ savoir calculer $\Pi(x)$ pour toute instance $x \in \{0, 1\}^*$ (ou d'un sous-domaine prédéfini).

Types de problèmes (exemples portant sur la coloration d'un graphe donné G)

- ▶ **Décision:** est-ce que G admet une 3-coloration ?
- ▶ **Recherche:** trouver une 3-coloration de G .
- ▶ **Dénombrement:** combien G admet-il de 3-colorations ?
- ▶ **Optimisation:** trouver le plus petit k tel que G admet une k -coloration.

Complexité algorithmique ?

Étude des ressources nécessaires pour résoudre un **problème**.

Problème $\Pi : \{0, 1\}^* \rightarrow \{0, 1\}^*$

Met en relation une **entrée** (ou **instance**) avec une **sortie**.

Résoudre un problème $\Pi =$ savoir calculer $\Pi(x)$ pour toute instance $x \in \{0, 1\}^*$ (ou d'un sous-domaine prédéfini).

Types de problèmes (exemples portant sur la coloration d'un graphe donné G)

- ▶ **Décision:** est-ce que G admet une 3-coloration ?
- ▶ **Recherche:** trouver une 3-coloration de G .
- ▶ **Dénombrement:** combien G admet-il de 3-colorations ?
- ▶ **Optimisation:** trouver le plus petit k tel que G admet une k -coloration.

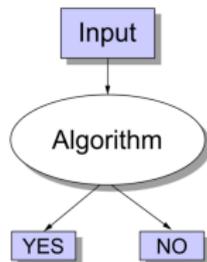
Problème de décision (focus)

Cas particulier où $\Pi : \{0, 1\}^* \rightarrow \{0, 1\}$ (réponse OUI ou NON)

$x \in \{0, 1\}^*$ est une **instance positive** si $\Pi(x) = 1$ (**négative** si $\Pi(x) = 0$)

Les instances positives définissent un **langage formel** $L \subseteq \{0, 1\}^*$

Résoudre un problème de décision \equiv reconnaître le langage associé



Complexité algorithmique ?

Étude des **ressources** nécessaires pour résoudre un problème.

Quel type de ressources ?

- ▶ Temps
- ▶ Espace (= mémoire)
- ▶ Non-déterminisme
- ▶ Aléa (p.ex. nombre de bits aléatoires)
- ▶ ...

Complexité algorithmique ?

Étude des **ressources** nécessaires pour résoudre un problème.

Quel type de ressources ?

- ▶ Temps
- ▶ Espace (= mémoire)
- ▶ Non-déterminisme
- ▶ Aléa (p.ex. nombre de bits aléatoires)
- ▶ ...

Point de vue asymptotique

- ▶ **Évolution** de ces quantités en fonction de la **taille** de l'entrée (en général, notée n)
- ▶ Notations $O(\cdot)$, $\Omega(\cdot)$, $\Theta(\cdot)$, $o(\cdot)$, $\omega(\cdot)$
Intuition \leq \geq $=$ $<$ $>$ pour $n \rightarrow \infty$, en ignorant les facteurs constants
- ▶ Quelques adjectifs:

Constant	$O(1)$	Quadratique	$O(n^2)$
Logarithmique	$O(\log n)$	Polynomial	$O(n^c) = n^{O(1)} = \text{poly}(n)$
Linéaire	$O(n)$	Exponentiel	$O(2^n)$ ou $O(2^{\text{poly}(n)})$
Quasi-linéaire	$O(n \log n)$	Factoriel	$O(n^n)$

L'espace et le temps

Sauf mention contraire, on ne parle désormais que de problèmes **de décision**.

Classes génériques

- ▶ $\text{TIME}(f(n))$: Ensemble de problèmes que l'on peut résoudre en temps $O(f(n))$.
- ▶ $\text{SPACE}(f(n))$: Ensemble de problèmes que l'on peut résoudre en espace $O(f(n))$.

L'espace et le temps

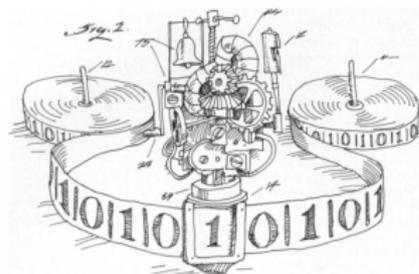
Sauf mention contraire, on ne parle désormais que de problèmes **de décision**.

Classes génériques

- ▶ $\text{TIME}(f(n))$: Ensemble de problèmes que l'on peut résoudre en temps $O(f(n))$.
- ▶ $\text{SPACE}(f(n))$: Ensemble de problèmes que l'on peut résoudre en espace $O(f(n))$.

Modèle de machine ?

- ▶ Machine de Turing déterministe
- ▶ Intuition "algorithme standard" OK
- ▶ Universalité ?



L'espace et le temps

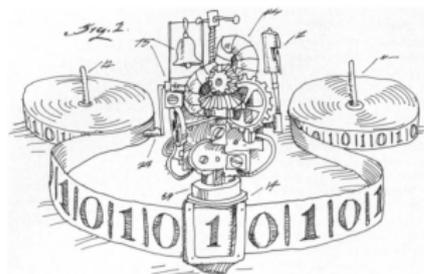
Sauf mention contraire, on ne parle désormais que de problèmes de **décision**.

Classes génériques

- ▶ $\text{TIME}(f(n))$: Ensemble de problèmes que l'on peut résoudre en temps $O(f(n))$.
- ▶ $\text{SPACE}(f(n))$: Ensemble de problèmes que l'on peut résoudre en espace $O(f(n))$.

Modèle de machine ?

- ▶ Machine de Turing déterministe
- ▶ Intuition "algorithme standard" OK
- ▶ Universalité ?



Classes incontournables

L	:=	$\text{SPACE}(\log n)$	Problèmes résolubles en espace logarithmique
P	:=	$\text{TIME}(\text{poly}(n))$	Problèmes résolubles en temps polynomial
PSPACE	:=	$\text{SPACE}(\text{poly}(n))$	Problèmes résolubles en espace polynomial
EXP	:=	$\text{TIME}(2^{\text{poly}(n)})$	Problèmes résolubles en temps exponentiel

$$L \subseteq P \subseteq \text{PSPACE} \subseteq \text{EXP}$$

L'espace et le temps (suite)

$$L \subseteq P \subseteq \text{PSPACE} \subseteq \text{EXP}$$

Aucune inclusion n'est démontrée stricte, mais elles sont toutes soupçonnées l'être.

L'espace et le temps (suite)

$$L \subseteq P \subseteq \text{PSPACE} \subseteq \text{EXP}$$

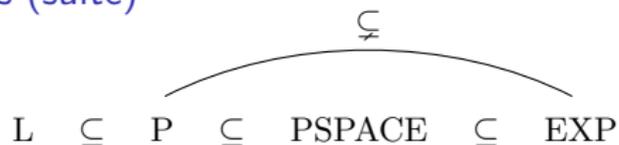
Aucune inclusion n'est démontrée stricte, mais elles sont toutes soupçonnées l'être.

Théorèmes de hiérarchie (légèrement simplifiés)

Pour deux fonctions $f_1(n)$ et $f_2(n)$ abrégées en f_1 et f_2 :

- ▶ Hiérarchie en temps :
 $f_1 = o(f_2 \log f_2) \implies \text{TIME}(f_1) \subsetneq \text{TIME}(f_2)$ (Stearns et Hartmanis'65, puis d'autres)

L'espace et le temps (suite)



Aucune inclusion n'est démontrée stricte, mais elles sont toutes soupçonnées l'être.

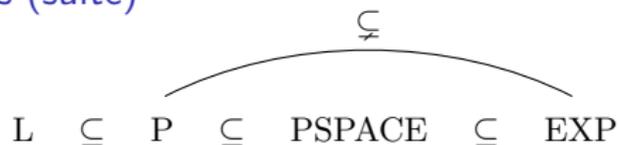
Théorèmes de hiérarchie (légèrement simplifiés)

Pour deux fonctions $f_1(n)$ et $f_2(n)$ abrégées en f_1 et f_2 :

- ▶ Hiérarchie en temps :

$$f_1 = o(f_2 \log f_2) \implies \text{TIME}(f_1) \subsetneq \text{TIME}(f_2) \quad (\text{Stearns et Hartmanis '65, puis d'autres})$$

L'espace et le temps (suite)



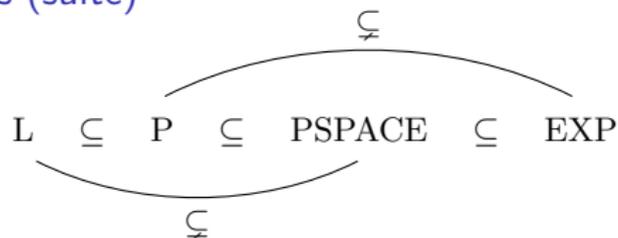
Aucune inclusion n'est démontrée stricte, mais elles sont toutes soupçonnées l'être.

Théorèmes de hiérarchie (légèrement simplifiés)

Pour deux fonctions $f_1(n)$ et $f_2(n)$ abrégées en f_1 et f_2 :

- ▶ Hiérarchie en temps :
 $f_1 = o(f_2 \log f_2) \implies \text{TIME}(f_1) \subsetneq \text{TIME}(f_2)$ (Stearns et Hartmanis'65, puis d'autres)
- ▶ Hiérarchie en espace :
 $f_1 = o(f_2) \implies \text{SPACE}(f_1) \subsetneq \text{SPACE}(f_2)$

L'espace et le temps (suite)



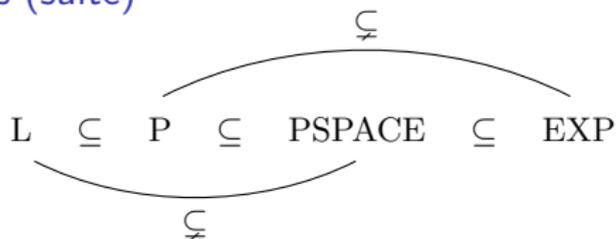
Aucune inclusion n'est démontrée stricte, mais elles sont toutes soupçonnées l'être.

Théorèmes de hiérarchie (légèrement simplifiés)

Pour deux fonctions $f_1(n)$ et $f_2(n)$ abrégées en f_1 et f_2 :

- ▶ Hiérarchie en temps :
 $f_1 = o(f_2 \log f_2) \implies \text{TIME}(f_1) \subsetneq \text{TIME}(f_2)$ (Stearns et Hartmanis'65, puis d'autres)
- ▶ Hiérarchie en espace :
 $f_1 = o(f_2) \implies \text{SPACE}(f_1) \subsetneq \text{SPACE}(f_2)$

L'espace et le temps (suite)



Aucune inclusion n'est démontrée stricte, mais elles sont toutes soupçonnées l'être.

Théorèmes de hiérarchie (légèrement simplifiés)

Pour deux fonctions $f_1(n)$ et $f_2(n)$ abrégées en f_1 et f_2 :

- ▶ Hiérarchie en temps :
 $f_1 = o(f_2 \log f_2) \implies \text{TIME}(f_1) \subsetneq \text{TIME}(f_2)$ (Stearns et Hartmanis'65, puis d'autres)
- ▶ Hiérarchie en espace :
 $f_1 = o(f_2) \implies \text{SPACE}(f_1) \subsetneq \text{SPACE}(f_2)$

Espace versus temps ?

- ▶ $\text{TIME}(f(n)) \subseteq \text{SPACE}(f(n)/\log(f(n)))$ (Hopcroft, Paul, Valiant'77)
→ L'espace est **strictement** plus fort que le temps !

Non-déterminisme (NP et coNP)

Plusieurs définitions équivalentes, la plus simple :

NP: \exists **preuve courte** que la réponse est OUI (certificat positif)

coNP: \exists **preuve courte** que la réponse est NON (certificat négatif)

Preuve courte = vérifiable en temps polynomial

Exemple : 3-COLORATION \in NP

(certificat = la coloration elle-même)

Non-déterminisme (NP et coNP)

Plusieurs définitions équivalentes, la plus simple :

NP: \exists **preuve courte** que la réponse est OUI (certificat positif)

coNP: \exists **preuve courte** que la réponse est NON (certificat négatif)

Preuve courte = vérifiable en temps polynomial

Exemple : 3-COLORATION \in NP

(certificat = la coloration elle-même)

Intuition non-déterministe : capacité à explorer en parallèle (**deviner**) un certificat.

Non-déterminisme (NP et coNP)

Plusieurs définitions équivalentes, la plus simple :

NP: \exists **preuve courte** que la réponse est OUI (certificat positif)

coNP: \exists **preuve courte** que la réponse est NON (certificat négatif)

Preuve courte = vérifiable en temps polynomial

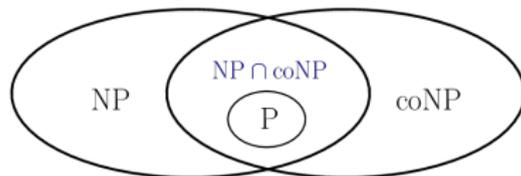
Exemple : 3-COLORATION \in NP

(certificat = la coloration elle-même)

Intuition non-déterministe : capacité à explorer en parallèle (**deviner**) un certificat.

Exercice : classer les problèmes suivant

- ▶ Ce graphe est-il connexe ?
- ▶ Ce nombre est-il premier ? est-il composite ?
- ▶ Ce nombre admet-il un facteur $\leq k$?
- ▶ Ces deux graphes sont-ils isomorphes ?
- ▶ Ce graphe admet-il une clique de taille k ?
- ▶ Ce plateau d'échec ($n \times n$) admet-il une stratégie gagnante pour le joueur 1 ?
- ▶ L'hypothèse de Riemann est-elle vraie ?
Cet énoncé est-il vrai ?



Non-déterminisme (suite)

Plus généralement

- ▶ $\text{NTIME}(f(n))$: \exists certificats positifs vérifiables en temps $O(f(n))$.
- ▶ $\text{NSPACE}(f(n))$: \exists certificats positifs vérifiables en espace $O(f(n))$.
- ▶ coNTIME et coNSPACE : idem pour certificats négatifs.

Non-déterminisme (suite)

Plus généralement

- ▶ $\text{NTIME}(f(n))$: \exists certificats positifs vérifiables en temps $O(f(n))$.
- ▶ $\text{NSPACE}(f(n))$: \exists certificats positifs vérifiables en espace $O(f(n))$.
- ▶ coNTIME et coNSPACE : idem pour certificats négatifs.

Classes non-déterministes incontournables (les mêmes qu'avant)

NL	:=	$\text{NSPACE}(\log n)$
NP	:=	$\text{NTIME}(\text{poly}(n))$
NPSPACE	:=	$\text{NSPACE}(\text{poly}(n))$
NEXP	:=	$\text{NTIME}(2^{\text{poly}(n)})$

Non-déterminisme (suite)

Plus généralement

- ▶ $\text{NTIME}(f(n))$: \exists certificats positifs vérifiables en temps $O(f(n))$.
- ▶ $\text{NSPACE}(f(n))$: \exists certificats positifs vérifiables en espace $O(f(n))$.
- ▶ coNTIME et coNSPACE : idem pour certificats négatifs.

Classes non-déterministes incontournables (les mêmes qu'avant)

NL	:=	$\text{NSPACE}(\log n)$
NP	:=	$\text{NTIME}(\text{poly}(n))$
NPSPACE	:=	$\text{NSPACE}(\text{poly}(n))$
NEXP	:=	$\text{NTIME}(2^{\text{poly}(n)})$

Relations

- ▶ $P \subseteq \text{NP} \cap \text{coNP}$ (algo = vérifieur avec certificat vide)
- ▶ $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$ (théorème de Savitch'70)
- ▶ $\text{NSPACE}(f(n)) = \text{coNSPACE}(f(n))$ (théorème d'Immerman–Szelepcsényi'87)
- ▶ $\text{NTIME}(f(n)) \subseteq \text{SPACE}(f(n))$ (énumération des certificats)

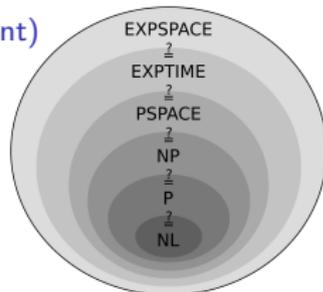
Non-déterminisme (suite)

Plus généralement

- ▶ $\text{NTIME}(f(n))$: \exists certificats positifs vérifiables en temps $O(f(n))$.
- ▶ $\text{NSPACE}(f(n))$: \exists certificats positifs vérifiables en espace $O(f(n))$.
- ▶ coNTIME et coNSPACE : idem pour certificats négatifs.

Classes non-déterministes incontournables (les mêmes qu'avant)

NL	:=	$\text{NSPACE}(\log n)$
NP	:=	$\text{NTIME}(\text{poly}(n))$
NPSPACE	:=	$\text{NSPACE}(\text{poly}(n))$
NEXP	:=	$\text{NTIME}(2^{\text{poly}(n)})$



Relations

- ▶ $P \subseteq \text{NP} \cap \text{coNP}$ (algo = vérifieur avec certificat vide)
- ▶ $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$ (théorème de Savitch'70)
- ▶ $\text{NSPACE}(f(n)) = \text{coNSPACE}(f(n))$ (théorème d'Immerman–Szelepcsényi'87)
- ▶ $\text{NTIME}(f(n)) \subseteq \text{SPACE}(f(n))$ (énumération des certificats)
- ▶ $L \stackrel{?}{=} \text{NL} \stackrel{?}{=} P \stackrel{?}{=} \text{NP} \stackrel{?}{=} \text{PSPACE} \stackrel{?}{=} \text{EXP} \stackrel{?}{=} \text{NEXP} \stackrel{?}{=} \text{EXPSPACE}$ (inclusions OK)

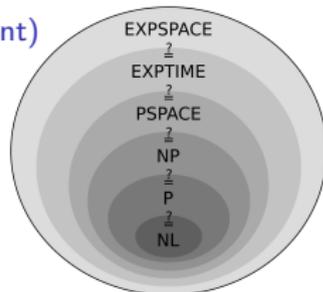
Non-déterminisme (suite)

Plus généralement

- ▶ $\text{NTIME}(f(n))$: \exists certificats positifs vérifiables en temps $O(f(n))$.
- ▶ $\text{NSPACE}(f(n))$: \exists certificats positifs vérifiables en espace $O(f(n))$.
- ▶ coNTIME et coNSPACE : idem pour certificats négatifs.

Classes non-déterministes incontournables (les mêmes qu'avant)

NL	:=	$\text{NSPACE}(\log n)$
NP	:=	$\text{NTIME}(\text{poly}(n))$
NPSPACE	:=	$\text{NSPACE}(\text{poly}(n))$
NEXP	:=	$\text{NTIME}(2^{\text{poly}(n)})$



Relations

- ▶ $P \subseteq \text{NP} \cap \text{coNP}$ (algo = vérifieur avec certificat vide)
- ▶ $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$ (théorème de Savitch'70)
- ▶ $\text{NSPACE}(f(n)) = \text{coNSPACE}(f(n))$ (théorème d'Immerman–Szelepcsényi'87)
- ▶ $\text{NTIME}(f(n)) \subseteq \text{SPACE}(f(n))$ (énumération des certificats)
- ▶ $L \stackrel{?}{=} \text{NL} \stackrel{?}{=} P \stackrel{?}{=} \text{NP} \stackrel{?}{=} \text{PSPACE} \stackrel{?}{=} \text{EXP} \stackrel{?}{=} \text{NEXP} \stackrel{?}{=} \text{EXPSPACE}$ (inclusions OK)
- ▶ Arguments de remplissage (padding) : $P = \text{NP} \implies \text{EXP} = \text{NEXP}$ (et d'autres)

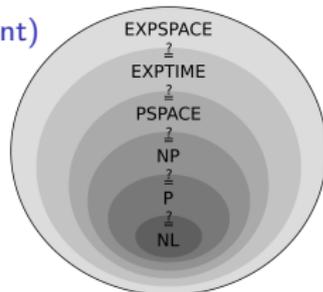
Non-déterminisme (suite)

Plus généralement

- ▶ $\text{NTIME}(f(n))$: \exists certificats positifs vérifiables en temps $O(f(n))$.
- ▶ $\text{NSPACE}(f(n))$: \exists certificats positifs vérifiables en espace $O(f(n))$.
- ▶ coNTIME et coNSPACE : idem pour certificats négatifs.

Classes non-déterministes incontournables (les mêmes qu'avant)

NL	:=	$\text{NSPACE}(\log n)$
NP	:=	$\text{NTIME}(\text{poly}(n))$
NPSPACE	:=	$\text{NSPACE}(\text{poly}(n))$
NEXP	:=	$\text{NTIME}(2^{\text{poly}(n)})$



Relations

- ▶ $P \subseteq \text{NP} \cap \text{coNP}$ (algo = vérifieur avec certificat vide)
- ▶ $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$ (théorème de Savitch'70)
- ▶ $\text{NSPACE}(f(n)) = \text{coNSPACE}(f(n))$ (théorème d'Immerman–Szelepcsényi'87)
- ▶ $\text{NTIME}(f(n)) \subseteq \text{SPACE}(f(n))$ (énumération des certificats)
- ▶ $L \stackrel{?}{=} \text{NL} \stackrel{?}{=} P \stackrel{?}{=} \text{NP} \stackrel{?}{=} \text{PSPACE} \stackrel{?}{=} \text{EXP} \stackrel{?}{=} \text{NEXP} \stackrel{?}{=} \text{EXPSPACE}$ (inclusions OK)
- ▶ Arguments de remplissage (padding) : $P = \text{NP} \implies \text{EXP} = \text{NEXP}$ (et d'autres)
- ▶ $P \stackrel{?}{=} \text{NP} \sim$ Ce qui est facile à vérifier est-il facile à trouver ?

Réductions et complétude

Réduction polynomiale

Un problème Π_1 **se réduit** en *temps polynomial* à un problème Π_2 , noté $\Pi_1 \leq \Pi_2$, si...

Réductions et complétude

Réduction polynomiale

Un problème Π_1 **se réduit** en *temps polynomial* à un problème Π_2 , noté $\Pi_1 \leq \Pi_2$, si...

- ▶ (par transformation) ... pour toute instance x_1 de Π_1 , on peut construire en temps polynomial une instance x_2 de Π_2 telle que $\Pi_1(x_1) = \Pi_2(x_2)$ (ou l'inverse).

Réductions et complétude

Réduction polynomiale

Un problème Π_1 **se réduit** en *temps polynomial* à un problème Π_2 , noté $\Pi_1 \leq \Pi_2$, si...

- ▶ (par transformation) ... pour toute instance x_1 de Π_1 , on peut construire en temps polynomial une instance x_2 de Π_2 telle que $\Pi_1(x_1) = \Pi_2(x_2)$ (ou l'inverse).
- ▶ (par oracle) ... on peut écrire un algorithme qui résout Π_1 en temps polynomial en ayant accès à **un oracle** pour Π_2 (machine qui répond à des questions de Π_2).

On dit que " Π_2 est au moins aussi difficile que Π_1 ".

Réductions et complétude

Réduction polynomiale

Un problème Π_1 **se réduit** en *temps polynomial* à un problème Π_2 , noté $\Pi_1 \leq \Pi_2$, si...

- ▶ (par transformation) ... pour toute instance x_1 de Π_1 , on peut construire en temps polynomial une instance x_2 de Π_2 telle que $\Pi_1(x_1) = \Pi_2(x_2)$ (ou l'inverse).
- ▶ (par oracle) ... on peut écrire un algorithme qui résout Π_1 en temps polynomial en ayant accès à **un oracle** pour Π_2 (machine qui répond à des questions de Π_2).

On dit que “ Π_2 est au moins aussi difficile que Π_1 ”.

Équivalence entre transformation et oracle ? Autres réductions ?

Réductions et complétude

Réduction polynomiale

Un problème Π_1 **se réduit** en *temps polynomial* à un problème Π_2 , noté $\Pi_1 \leq \Pi_2$, si...

- ▶ (par transformation) ... pour toute instance x_1 de Π_1 , on peut construire en temps polynomial une instance x_2 de Π_2 telle que $\Pi_1(x_1) = \Pi_2(x_2)$ (ou l'inverse).
- ▶ (par oracle) ... on peut écrire un algorithme qui résout Π_1 en temps polynomial en ayant accès à **un oracle** pour Π_2 (machine qui répond à des questions de Π_2).

On dit que " Π_2 est au moins aussi difficile que Π_1 ".

Équivalence entre transformation et oracle ? Autres réductions ?

Difficulté et complétude

Pour une classe \mathcal{C} de problèmes et un type de réduction:

- ▶ \mathcal{C} -difficile: au moins aussi difficile que tout problème dans \mathcal{C}
- ▶ \mathcal{C} -complet: à la fois dans \mathcal{C} et \mathcal{C} -difficile

Réductions et complétude

Réduction polynomiale

Un problème Π_1 **se réduit** en *temps polynomial* à un problème Π_2 , noté $\Pi_1 \leq \Pi_2$, si...

- ▶ (par transformation) ... pour toute instance x_1 de Π_1 , on peut construire en temps polynomial une instance x_2 de Π_2 telle que $\Pi_1(x_1) = \Pi_2(x_2)$ (ou l'inverse).
- ▶ (par oracle) ... on peut écrire un algorithme qui résout Π_1 en temps polynomial en ayant accès à **un oracle** pour Π_2 (machine qui répond à des questions de Π_2).

On dit que " Π_2 est au moins aussi difficile que Π_1 ".

Équivalence entre transformation et oracle ? Autres réductions ?

Difficulté et complétude

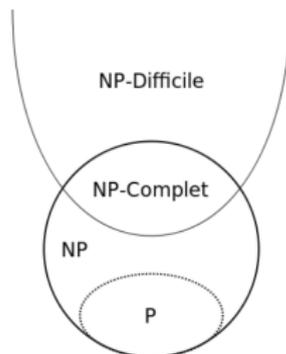
Pour une classe \mathcal{C} de problèmes et un type de réduction:

- ▶ \mathcal{C} -difficile: au moins aussi difficile que tout problème dans \mathcal{C}
- ▶ \mathcal{C} -complet: à la fois dans \mathcal{C} et \mathcal{C} -difficile

Ex: NP-complet = dans NP et NP-difficile

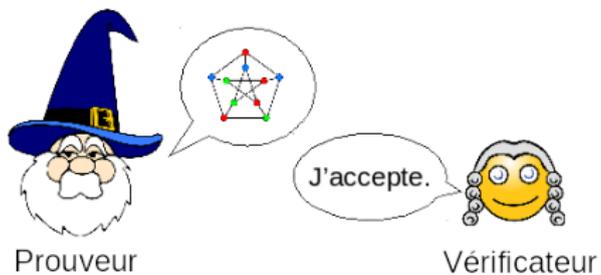
(ici, *par transformation en temps polynomial*)

SAT \in NP-complet (Cook, Levin'71)



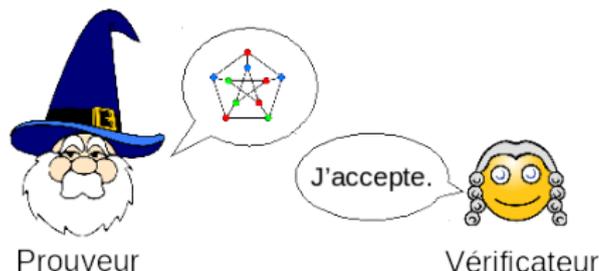
Preuves interactives

Une autre façon de voir NP



Preuves interactives

Une autre façon de voir NP



Variantes

- ▶ Le vérificateur peut utiliser de l'aléa: Arthur-Merlin (AM) et Merlin-Arthur (MA)
- ▶ Aléa + nombre d'interaction polynomial: classe IP
- ▶ $IP = PSPACE$ (Shamir'92)
- ▶ Plusieurs prouveurs (MIP), $MIP = NEXPTIME$ (Babai, Fortnow, Lund'91)
- ▶ $MIP^* = RE$ (Ji, Natarajan, Vidick, Wright, Yuen'20)
(MIP^* : les prouveurs partagent des bits intriqués)
(RE : langages récursivement énumérables)

Machines à oracles et hiérarchie polynomiale

Classes définies par oracles

- ▶ Soient deux classes X et Y , on note X^Y la classe des problèmes résolubles par un algorithme de X ayant accès à un oracle pour un problème Y -complet.
- ▶ Par exemple, P^{NP} = problèmes résolubles en temps polynomial avec un oracle pour SAT.
- ▶ Les oracles ne sont pas forcément des pbs de décisions, par exemple, $\#P$ (dénombrement).

Machines à oracles et hiérarchie polynomiale

Classes définies par oracles

- ▶ Soient deux classes X et Y , on note X^Y la classe des problèmes résolubles par un algorithme de X ayant accès à un oracle pour un problème Y -complet.
- ▶ Par exemple, P^{NP} = problèmes résolubles en temps polynomial avec un oracle pour SAT.
- ▶ Les oracles ne sont pas forcément des pbs de décisions, par exemple, $\#P$ (dénombrement).

Hiérarchie polynomiale (PH)

- ▶ Principe : Itération d'oracles impliquant P , NP et $coNP$

- ▶ Problèmes complets à chaque étage

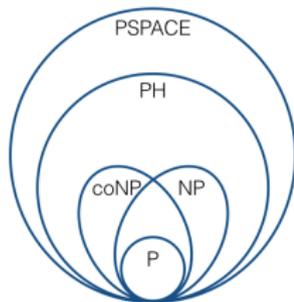
Quantified SAT (QSAT):

$$\exists x_1 \forall x_2 \exists x_3 \dots, \phi(x_1, x_2, x_3, \dots)$$

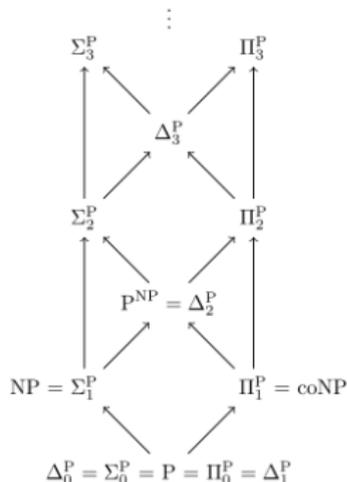
- ▶ Machine de Turing alternantes

- ▶ $PH \subseteq P^{\#P}$ (Toda'91)

- ▶ $PH \stackrel{?}{=} PSPACE$



(Meyer et Stockmeyer'72)



Aléa et ressources quantiques

BPP: *Bounded-error probabilistic polynomial time*

- ▶ Problèmes résolubles en temps polynomial par un **algorithme probabiliste** (peut tirer à pile ou face) avec une probabilité d'erreur strictement inférieure à $1/2$

Aléa et ressources quantiques

BPP: *Bounded-error probabilistic polynomial time*

- ▶ Problèmes résolubles en temps polynomial par un **algorithme probabiliste** (peut tirer à pile ou face) avec une probabilité d'erreur strictement inférieure à $1/2$

BQP: *Bounded error quantum polynomial time*

- ▶ Problèmes résolubles en temps polynomial par un **ordinateur quantique** avec une probabilité d'erreur strictement inférieure à $1/2$

Aléa et ressources quantiques

BPP: *Bounded-error probabilistic polynomial time*

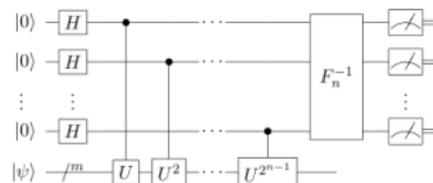
- ▶ Problèmes résolubles en temps polynomial par un **algorithme probabiliste** (peut tirer à pile ou face) avec une probabilité d'erreur strictement inférieure à $1/2$

BQP: *Bounded error quantum polynomial time*

- ▶ Problèmes résolubles en temps polynomial par un **ordinateur quantique** avec une probabilité d'erreur strictement inférieure à $1/2$
- ▶ Concrètement, circuit ou machine de Turing ?

Les deux, au choix :

- ▶ circuits quantiques uniformes
- ▶ machines de turing quantiques



Aléa et ressources quantiques

BPP: *Bounded-error probabilistic polynomial time*

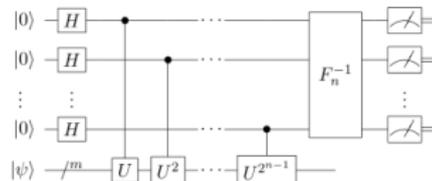
- ▶ Problèmes résolubles en temps polynomial par un **algorithme probabiliste** (peut tirer à pile ou face) avec une probabilité d'erreur strictement inférieure à $1/2$

BQP: *Bounded error quantum polynomial time*

- ▶ Problèmes résolubles en temps polynomial par un **ordinateur quantique** avec une probabilité d'erreur strictement inférieure à $1/2$
- ▶ Concrètement, circuit ou machine de Turing ?

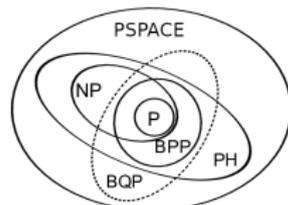
Les deux, au choix :

- ▶ circuits quantiques uniformes
- ▶ machines de turing quantiques



Que sait-on ?

- ▶ $P \subseteq BPP$ ($0 < 1/2$)
- ▶ $BPP \subseteq BQP$ (non-réversibilité simulable en temps poly)
- ▶ $BQP \subseteq PSPACE$ (Bernstein et Vazirani'97)
- ▶ $FACTORIZING \in BQP$ (Shor'94)
- ▶ Quid de BQP versus NP ? (pressentie incomparables)



(attention \uparrow conjectures)

Séparation par oracle

Principe : on veut montrer que deux classes sont différentes, mais on y arrive pas.

→ Essayer de montrer au moins qu'elles sont différentes **relativement** à un certain oracle.

Séparation par oracle

Principe : on veut montrer que deux classes sont différentes, mais on y arrive pas.

→ Essayer de montrer au moins qu'elles sont différentes **relativement** à un certain oracle.

Exemples de séparations

- ▶ Il existe un oracle X tel que $P^X \neq NP^X$ (Baker, Gill, Solovay'75)
(cependant : il existe aussi Y tel que $P^Y = NP^Y$) ("Relativization barrier")
- ▶ $\exists X$ tel que $NP^X \subsetneq BQP^X$ (Bennett, Bernstein, Brassard, Vazirani'97)
- ▶ $\exists X$ tel que $BQP^X \subsetneq NP^X$ (Berthiaume, Brassard'92)
- ▶ $\exists X$ tel que $BQP^X \subsetneq PH^X$ (Raz, Tal'19)

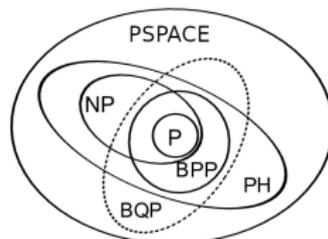
Séparation par oracle

Principe : on veut montrer que deux classes sont différentes, mais on y arrive pas.

→ Essayer de montrer au moins qu'elles sont différentes **relativement** à un certain oracle.

Exemples de séparations

- ▶ Il existe un oracle X tel que $P^X \neq NP^X$ (Baker, Gill, Solovay'75)
(cependant : il existe aussi Y tel que $P^Y = NP^Y$) ("Relativization barrier")
- ▶ $\exists X$ tel que $NP^X \subsetneq BQP^X$ (Bennett, Bernstein, Brassard, Vazirani'97)
- ▶ $\exists X$ tel que $BQP^X \subsetneq NP^X$ (Berthiaume, Brassard'92)
- ▶ $\exists X$ tel que $BQP^X \subsetneq PH^X$ (Raz, Tal'19)



(attention ↑ conjectures)

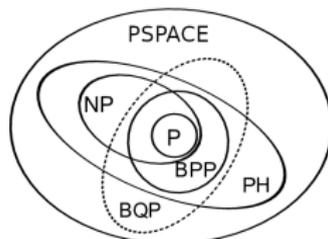
Séparation par oracle

Principe : on veut montrer que deux classes sont différentes, mais on y arrive pas.

→ Essayer de montrer au moins qu'elles sont différentes **relativement** à un certain oracle.

Exemples de séparations

- ▶ Il existe un oracle X tel que $P^X \neq NP^X$ (Baker, Gill, Solovay'75)
(cependant : il existe aussi Y tel que $P^Y = NP^Y$) ("Relativization barrier")
- ▶ $\exists X$ tel que $NP^X \subsetneq BQP^X$ (Bennett, Bernstein, Brassard, Vazirani'97)
- ▶ $\exists X$ tel que $BQP^X \subsetneq NP^X$ (Berthiaume, Brassard'92)
- ▶ $\exists X$ tel que $BQP^X \subsetneq PH^X$ (Raz, Tal'19)



(attention ↑ conjectures)

Remarque

N'importe quelle séparation non-relative impliquerait, à minima, que $P \neq PSPACE$, ce qui est un des principaux problèmes ouverts du domaine.

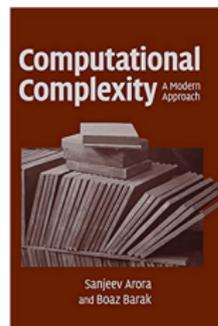
Les paris sont ouverts

Proposition	RW's Estimated Likelihood
TRUE	100%
$\text{EXP}^{\text{NP}} \neq \text{BPP}$	99%
$\text{NEXP} \not\subseteq \text{P/poly}$	97%
$\text{L} \neq \text{NP}$	95%
$\text{NP} \not\subseteq \text{SIZE}(n^k)$	93%
$\text{BPP} \subseteq \text{SUBEXP}$	90%
$\text{P} \neq \text{PSPACE}$	90%
$\text{P} \neq \text{NP}$	80%
ETH	70%
$\text{NC1} \neq \text{TC0}$	50%
$\text{NEXP} \neq \text{EXP}$	45%
SETH	25%
$\text{NEXP} \neq \text{coNEXP}$	20%
NSETH	15%
$\text{L} \neq \text{RL}$	5%
FALSE	0%

Ryan Williams (2019)

Pour aller plus loin

- ▶ Livre "Computational Complexity" (Arora et Barak'2007)
- ▶ Chaîne YouTube de Ryan O'Donnell (cours Carnegie Mellon)
- ▶ Wikipedia (si si !)

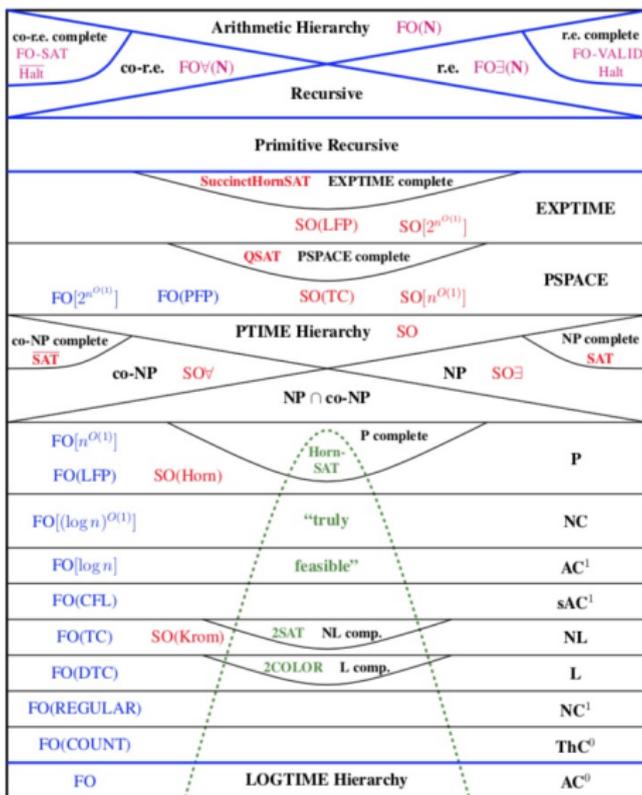


Choses dont je n'ai pas parlé

- ▶ Complexité de circuits (classe SIZE $(f(n))$, AC^0 , NC , $P/poly$)
- ▶ Bornes inférieures en lien avec les circuits
- ▶ Calcul du permanent ($\#P$ -complet) et Boson Sampling
- ▶ Complexité descriptive (liens entre logique et complexité algorithmique)
- ▶ Cryptographie : comment générer des instances difficiles
- ▶ Les cinq mondes d'Impagliazzo
- ▶ ...



Complexité descriptive



Neil Immerman (1999)