

IF228 - Calculabilité et Complexité

Cours #5 : complexité : P, NP, EXPTIME

Frédéric Herbreteau

`frederic.herbreteau@bordeaux-inp.fr`

(d'après les supports de Corentin Travers)

18 février 2025

Complexité

Recherche dans un tableau trié

SORTED-SEARCH

ENTRÉE : un tableau T trié, un entier x

QUESTION : x est-il un élément de T ?

1	3	8	9	11	23	45	78	82	93	107	234
---	---	---	---	----	----	----	----	----	----	-----	-----

- Peut-on décider $x \in T$ en temps linéaire $\mathcal{O}(|T|)$?

Recherche dans un tableau trié

SORTED-SEARCH

ENTRÉE : un tableau T trié, un entier x

QUESTION : x est-il un élément de T ?

1	3	8	9	11	23	45	78	82	93	107	234
---	---	---	---	----	----	----	----	----	----	-----	-----

- Peut-on décider $x \in T$ en temps linéaire $\mathcal{O}(|T|)$?
→ Oui : recherche linéaire

Recherche dans un tableau trié

SORTED-SEARCH

ENTRÉE : un tableau T trié, un entier x

QUESTION : x est-il un élément de T ?

1	3	8	9	11	23	45	78	82	93	107	234
---	---	---	---	----	----	----	----	----	----	-----	-----

- Peut-on décider $x \in T$ en temps linéaire $\mathcal{O}(|T|)$?
→ **Oui** : recherche linéaire
- Existe-t-il un algorithme **plus efficace** ?

Recherche dans un tableau trié

SORTED-SEARCH

ENTRÉE : un tableau T trié, un entier x

QUESTION : x est-il un élément de T ?

1	3	8	9	11	23	45	78	82	93	107	234
---	---	---	---	----	----	----	----	----	----	-----	-----

- Peut-on décider $x \in T$ en temps linéaire $\mathcal{O}(|T|)$?
→ **Oui** : recherche linéaire
- Existe-t-il un algorithme **plus efficace** ?
→ **Oui** : recherche dichotomique

Recherche dans un tableau trié

SORTED-SEARCH

ENTRÉE : un tableau T trié, un entier x

QUESTION : x est-il un élément de T ?

1	3	8	9	11	23	45	78	82	93	107	234
---	---	---	---	----	----	----	----	----	----	-----	-----

- Peut-on décider $x \in T$ en temps linéaire $\mathcal{O}(|T|)$?
→ **Oui** : recherche linéaire
- Existe-t-il un algorithme **plus efficace** ?
→ **Oui** : recherche dichotomique

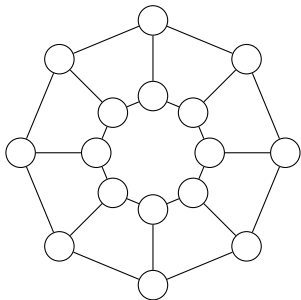
Gain exponentiel : $\mathcal{O}(\log_2(|T|))$ au lieu de $\mathcal{O}(|T|)$

Ensembles indépendants

IND-SET

ENTRÉE : un graphe G et un entier $k \in \mathbb{N}$

QUESTION : G admet-il un ensemble indépendant de taille k ?



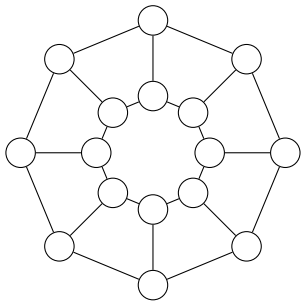
$k=8$?

Ensembles indépendants

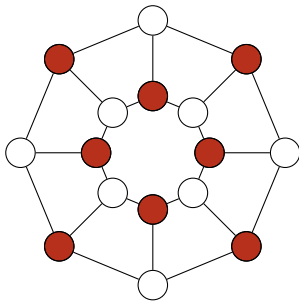
IND-SET

ENTRÉE : un graphe G et un entier $k \in \mathbb{N}$

QUESTION : G admet-il un ensemble indépendant de taille k ?



$k=8$?

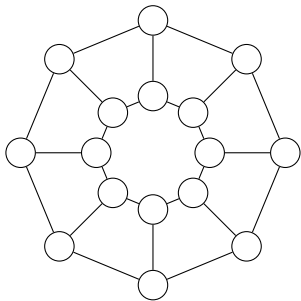


Ensembles indépendants

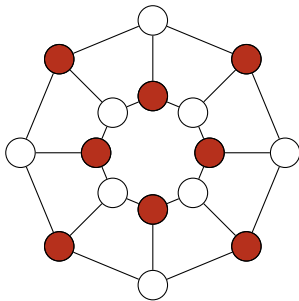
IND-SET

ENTRÉE : un graphe G et un entier $k \in \mathbb{N}$

QUESTION : G admet-il un ensemble indépendant de taille k ?



$k=8$?



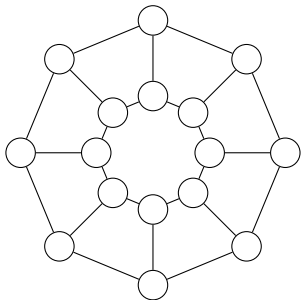
- Algorithme **brute-force** en $n!/k!(n-k)!$ pour n sommets

Ensembles indépendants

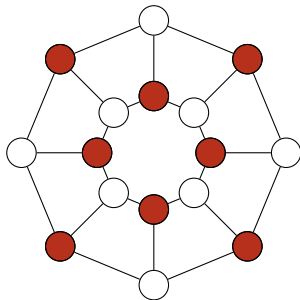
IND-SET

ENTRÉE : un graphe G et un entier $k \in \mathbb{N}$

QUESTION : G admet-il un ensemble indépendant de taille k ?



$k=8$?



- Algorithme **brute-force** en $n!/k!(n-k)!$ pour n sommets
- Solution **polynomiale** en la taille de G ?

Division entière

DIVIDE

ENTRÉE : deux entiers $x, y \in \mathbb{N}$ et $y \neq 0$

QUESTION : est-ce que y divise x ?

Division entière

DIVIDE

ENTRÉE : deux entiers $x, y \in \mathbb{N}$ et $y \neq 0$

QUESTION : est-ce que y divise x ?

Ex :

1458 | 6

Division entière

DIVIDE

ENTRÉE : deux entiers $x, y \in \mathbb{N}$ et $y \neq 0$

QUESTION : est-ce que y divise x ?

Ex :

$$1458 \mid 6$$

$$137 \mid 6$$

Division entière

DIVIDE

ENTRÉE : deux entiers $x, y \in \mathbb{N}$ et $y \neq 0$

QUESTION : est-ce que y divise x ?

Ex :

1458 $\mid$ 6

137 $\mid$ 6

Algorithme **polynomial** : $\mathcal{O}(\max(|x|, |y|)^2)$

Nombres premiers

PRIME

ENTRÉE : un entier $x \in \mathbb{N}$

QUESTION : x est-il premier ?

Ex : 879 190 747 est-il premier ?

Nombres premiers

PRIME

ENTRÉE : un entier $x \in \mathbb{N}$

QUESTION : x est-il premier ?

Ex : 879 190 747 est-il premier ?

- Algorithme **brute-force** en $\mathcal{O}(10^{|x|})$
pour chaque n entre 2 et $x-1$
tester si n divise x

Nombres premiers

PRIME

ENTRÉE : un entier $x \in \mathbb{N}$

QUESTION : x est-il premier ?

Ex : 879 190 747 est-il premier ?

- Algorithme **brute-force** en $\mathcal{O}(10^{|x|})$
pour chaque n entre 2 et $x-1$
tester si n divise x
- Solution **polynomiale** en la taille de x ?

Rappels

Fonctions, langages, problèmes

Calculabilité de la fonction **booléenne** : $f : \{0, 1\}^* \rightarrow \{0, 1\}$

Décidabilité du **langage** : $L_f = \{x \in \{0, 1\}^* \mid f(x) = 1\}$

Problème de **décision** :

ENTRÉE : $x \in \{0, 1\}^*$

QUESTION : est-ce que $x \in L_f$?

Temps de calcul

Soit $f : \{0, 1\}^* \rightarrow \{0, 1\}$ et $T : \mathbb{N} \rightarrow \mathbb{N}$ deux fonctions, et M une MT qui **s'arrête pour tout mot d'entrée**.

Temps de calcul

Soit $f : \{0, 1\}^* \rightarrow \{0, 1\}$ et $T : \mathbb{N} \rightarrow \mathbb{N}$ deux fonctions, et M une MT qui **s'arrête pour tout mot d'entrée**.

Définition

M **calcule** f (ou **décide** le langage L_f) **en temps** T ssi :

- M calcule f
- et pour tout $x \in \{0, 1\}^*$, M s'arrête sur l'entrée x après avoir effectué au plus $T(|x|)$ transitions.

Temps de calcul

Soit $f : \{0, 1\}^* \rightarrow \{0, 1\}$ et $T : \mathbb{N} \rightarrow \mathbb{N}$ deux fonctions, et M une MT qui **s'arrête pour tout mot d'entrée**.

Définition

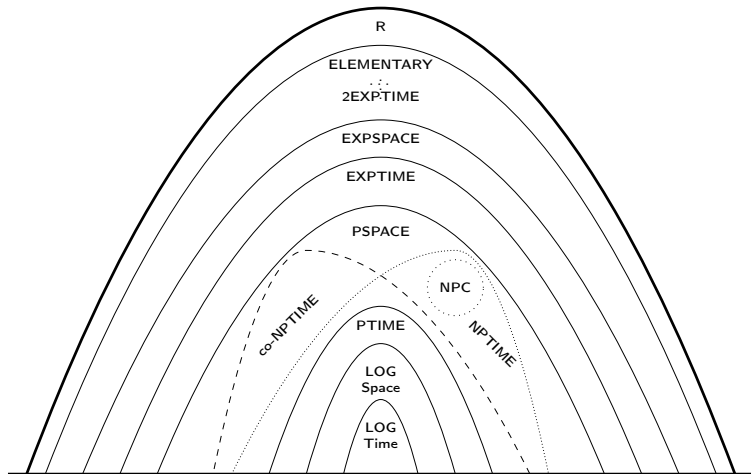
M **calcule** f (ou **décide** le langage L_f) **en temps** T ssi :

- M calcule f
- et pour tout $x \in \{0, 1\}^*$, M s'arrête sur l'entrée x après avoir effectué au plus $T(|x|)$ transitions.

NB : complexité **au pire**

Classes de complexité

Ensemble des fonctions **calculables** avec des ressources limitées en **temps** et/ou en **espace**



DTIME

Définition

Soit $T : \mathbb{N} \rightarrow \mathbb{N}$ une fonction. Un langage L est dans la classe **DTIME(T)** si et seulement si L est décidé par une MT **déterministe** en temps $c \cdot T$, où $c > 0$ est une constante.

DTIME

Définition

Soit $T : \mathbb{N} \rightarrow \mathbb{N}$ une fonction. Un langage L est dans la classe $DTIME(T)$ si et seulement si L est décidé par une MT **déterministe** en temps $c \cdot T$, où $c > 0$ est une constante.

Exemple :

- $DTIME(n^2)$: langages décidables en temps quadratique

DTIME

Définition

Soit $T : \mathbb{N} \rightarrow \mathbb{N}$ une fonction. Un langage L est dans la classe $DTIME(T)$ si et seulement si L est décidé par une MT **déterministe** en temps $c \cdot T$, où $c > 0$ est une constante.

Exemple :

- $DTIME(n^2)$: langages décidables en temps quadratique
- $DTIME(2^n)$: langages décidables en temps exponentiel

DTIME

Définition

Soit $T : \mathbb{N} \rightarrow \mathbb{N}$ une fonction. Un langage L est dans la classe **DTIME(T)** si et seulement si L est décidé par une MT **déterministe** en temps $c \cdot T$, où $c > 0$ est une constante.

Exemple :

- $DTIME(n^2)$: langages décidables en temps quadratique
- $DTIME(2^n)$: langages décidables en temps exponentiel
- $DTIME(\log_2(n))$: langages décidables en temps log.

DTIME

Définition

Soit $T : \mathbb{N} \rightarrow \mathbb{N}$ une fonction. Un langage L est dans la **classe** $DTIME(T)$ si et seulement si L est décidé par une MT **déterministe** en temps $c \cdot T$, où $c > 0$ est une constante.

Exemple :

- $DTIME(n^2)$: langages décidables en temps quadratique
- $DTIME(2^n)$: langages décidables en temps exponentiel
- $DTIME(\log_2(n))$: langages décidables en temps log.

$$DTIME(\log_2(n)) \subseteq DTIME(n^2) \subseteq DTIME(2^n)$$

DTIME

Définition

Soit $T : \mathbb{N} \rightarrow \mathbb{N}$ une fonction. Un langage L est dans la **classe** $DTIME(T)$ si et seulement si L est décidé par une MT **déterministe** en temps $c \cdot T$, où $c > 0$ est une constante.

Exemple :

- $DTIME(n^2)$: langages décidables en temps quadratique
- $DTIME(2^n)$: langages décidables en temps exponentiel
- $DTIME(\log_2(n))$: langages décidables en temps log.

$$DTIME(\log_2(n)) \subseteq DTIME(n^2) \subseteq DTIME(2^n)$$

NB : machine de Turing est **déterministe**

La classe **P** ou **PTIME**

$$\mathbf{P} = \bigcup_{c \geq 1} \text{DTIME}(n^c)$$

La classe **P** ou **PTIME**

$$\mathbf{P} = \bigcup_{c \geq 1} \text{DTIME}(n^c)$$

Classe des langages décidables en **temps polynomial** par une MT **déterministe**

La classe **P** ou **PTIME**

$$\mathbf{P} = \bigcup_{c \geq 1} \text{DTIME}(n^c)$$

Classe des langages décidables en **temps polynomial** par une MT **déterministe**

Exemple :

	∈ P?
SORTED-SEARCH	
PRIME	
DIVIDE	
IND-SET	

La classe **P** ou **PTIME**

$$\mathbf{P} = \bigcup_{c \geq 1} \text{DTIME}(n^c)$$

Classe des langages décidables en **temps polynomial** par une MT **déterministe**

Exemple :

	∈ P?	
SORTED-SEARCH	oui	parcours linéaire/dichotomique
PRIME		
DIVIDE		
IND-SET		

La classe **P** ou **PTIME**

$$\mathbf{P} = \bigcup_{c \geq 1} \text{DTIME}(n^c)$$

Classe des langages décidables en **temps polynomial** par une MT **déterministe**

Exemple :

	∈ P?	
SORTED-SEARCH	oui	parcours linéaire/dichotomique
PRIME	oui	Agrawal-Kayal-Saxena 2002
DIVIDE		
IND-SET		

La classe **P** ou **PTIME**

$$\mathbf{P} = \bigcup_{c \geq 1} \text{DTIME}(n^c)$$

Classe des langages décidables en **temps polynomial** par une MT **déterministe**

Exemple :

	∈ P?	
SORTED-SEARCH	oui	parcours linéaire/dichotomique
PRIME	oui	Agrawal-Kayal-Saxena 2002
DIVIDE	oui	cf. CM1
IND-SET		

La classe **P** ou **PTIME**

$$\mathbf{P} = \bigcup_{c \geq 1} \text{DTIME}(n^c)$$

Classe des langages décidables en **temps polynomial** par une MT **déterministe**

Exemple :

	∈ P?	
SORTED-SEARCH	oui	parcours linéaire/dichotomique
PRIME	oui	Agrawal-Kayal-Saxena 2002
DIVIDE	oui	cf. CM1
IND-SET	???	

Classe EXPTIME

$$\text{EXPTIME} = \bigcup_{c \geq 1} \text{DTIME}(2^{n^c})$$

NB : $P \subseteq \text{EXPTIME}$

Classe EXPTIME

$$\text{EXPTIME} = \bigcup_{c \geq 1} \text{DTIME}(2^{n^c})$$

NB : $P \subseteq \text{EXPTIME}$

Exemple : les problèmes SORTED-SEARCH, PRIME, DIVIDE et IND-SET sont dans la classe **EXPTIME**

Classe **EXPTIME**

$$\text{EXPTIME} = \bigcup_{c \geq 1} \text{DTIME}(2^{n^c})$$

NB : $P \subseteq \text{EXPTIME}$

Exemple : les problèmes SORTED-SEARCH, PRIME, DIVIDE et IND-SET sont dans la classe **EXPTIME**

- **Intuition** : les algorithmes pour les problèmes de **P** passent à l'échelle mieux que ceux de **EXPTIME**

Classe **EXPTIME**

$$\text{EXPTIME} = \bigcup_{c \geq 1} \text{DTIME}(2^{n^c})$$

NB : $P \subseteq \text{EXPTIME}$

Exemple : les problèmes SORTED-SEARCH, PRIME, DIVIDE et IND-SET sont dans la classe **EXPTIME**

- **Intuition** : les algorithmes pour les problèmes de **P** passent à l'échelle mieux que ceux de **EXPTIME**
→ à relativiser (exemple : n^{100})

Classe EXPTIME

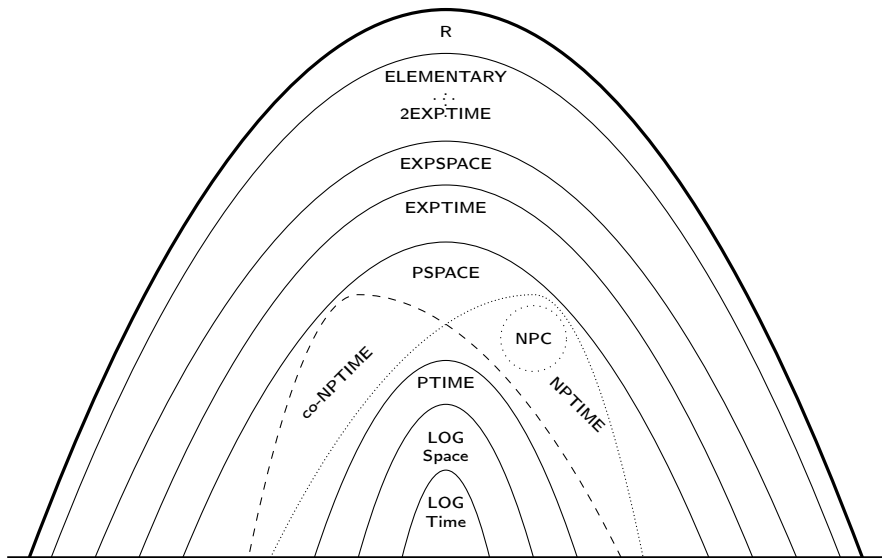
$$\text{EXPTIME} = \bigcup_{c \geq 1} \text{DTIME}(2^{n^c})$$

NB : $P \subseteq \text{EXPTIME}$

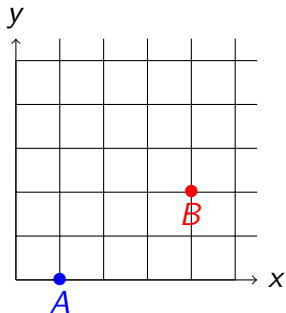
Exemple : les problèmes SORTED-SEARCH, PRIME, DIVIDE et IND-SET sont dans la classe **EXPTIME**

- **Intuition** : les algorithmes pour les problèmes de **P** passent à l'échelle mieux que ceux de **EXPTIME**
→ à **relativiser** (exemple : n^{100})
- Une approche **brute-force** donne très souvent un algorithme en temps exponentiel

Au-delà de EXPTIME ?

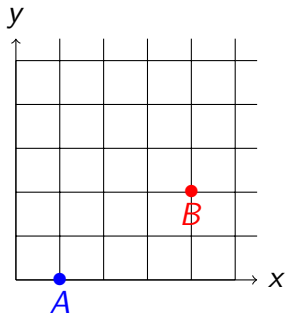


Un problème non-élémentaire



Chemin de A à B ?

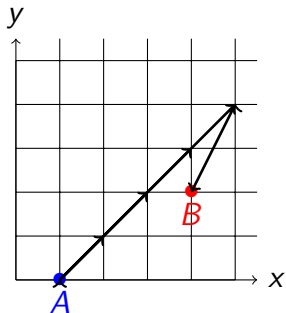
Un problème non-élémentaire



Chemin de A à B ?

$$V = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -2 \end{pmatrix} \right\}$$

Un problème non-élémentaire

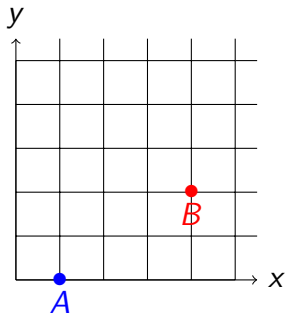


Chemin de A à B ?

$$V = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -2 \end{pmatrix} \right\}$$

→ **oui** : $B = A + 4v_1 + v_2$

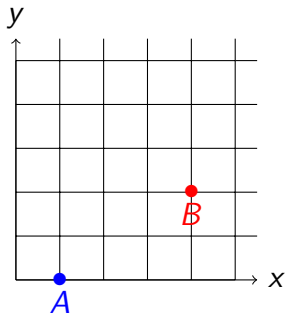
Un problème non-élémentaire



Chemin de A à B ?

$$V = \left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix} \right\}$$

Un problème non-élémentaire

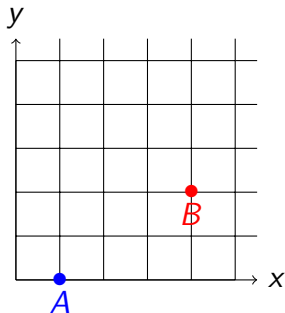


Chemin de A à B ?

$$V = \left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix} \right\}$$

→ **non** (tous les points accessibles ont $x \leq 1$)

Un problème non-élémentaire



Chemin de A à B ?

$$V = \left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix} \right\}$$

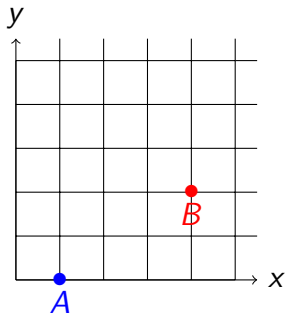
→ **non** (tous les points accessibles ont $x \leq 1$)

VAS-REACH : accessibilité VAS/Réseaux de Petri

ENTRÉE : $V \subseteq \mathbb{Z}^n$ ens. fini de vecteurs et deux points $A, B \in \mathbb{N}^n$

QUESTION : existe-t-il une séquence finie $w \in V^*$ tq $B = A + w$?

Un problème non-élémentaire



Chemin de A à B ?

$$V = \left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \end{pmatrix} \right\}$$

→ **non** (tous les points accessibles ont $x \leq 1$)

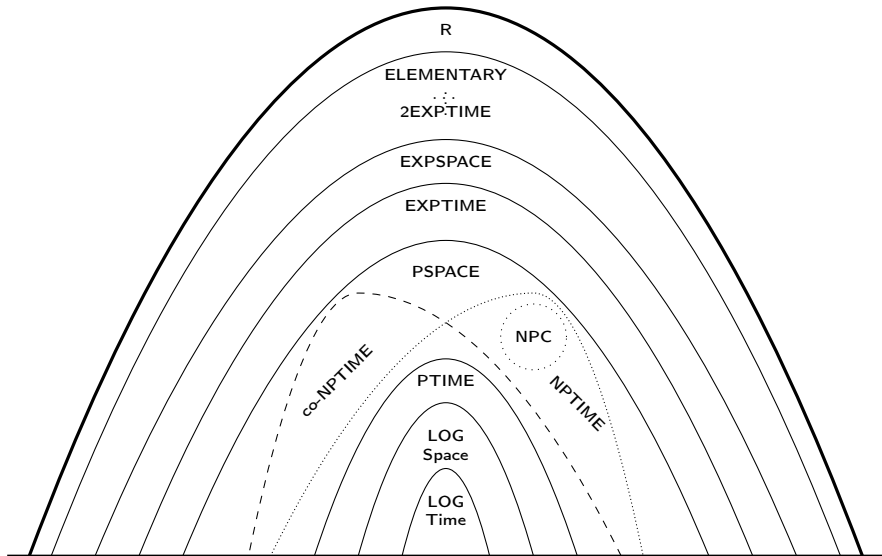
VAS-REACH : accessibilité VAS/Réseaux de Petri

ENTRÉE : $V \subseteq \mathbb{Z}^n$ ens. fini de vecteurs et deux points $A, B \in \mathbb{N}^n$

QUESTION : existe-t-il une séquence finie $w \in V^*$ tq $B = A + w$?

VAS-REACH est **non-élémentaire** et même **non primitif récursif** pour $n \geq 10$ (Leroux/Czerwinski&Orlikowski, 2021)

Entre P et EXPTIME?



La classe NP

Sudoku : problème de décision

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

SUDOKU

ENTRÉE : une grille G de taille n , partiellement remplie

QUESTION : G peut-elle complétée de façon valide ?

Sudoku : problème de décision

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

SUDOKU

ENTRÉE : une grille G de taille n , partiellement remplie

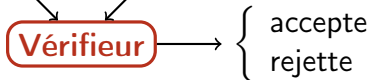
QUESTION : G peut-elle complétée de façon valide ?

Sudoku : vérification

Comment **vérifier** qu'une grille partielle est complétée de façon valide ?

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9



Sudoku : vérification

Comment **vérifier** qu'une grille partielle est complétée de façon valide ?

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Certificat

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Vérifieur

{ accepte
rejette

Vérifieur

Soit L un langage sur l'alphabet $\{0, 1\}$

Définition

Un **vérifieur** pour L est une MT **déterministe** V qui s'arrête pour toute entrée $\langle w, c \rangle$ et telle que :

- si $w \in L$ alors $\exists c \in \{0, 1\}^*$ tq V accepte $\langle w, c \rangle$
- et si $w \notin L$ alors $\forall c \in \{0, 1\}^*$, V rejette $\langle w, c \rangle$

NB : $L = \{w \mid V \text{ accepte } \langle w, c \rangle \text{ pour un certain } c \in \{0, 1\}^*\}$

Vérifieur

Soit L un langage sur l'alphabet $\{0, 1\}$

Définition

Un **vérifieur** pour L est une MT **déterministe** V qui s'arrête pour toute entrée $\langle w, c \rangle$ et telle que :

- si $w \in L$ alors $\exists c \in \{0, 1\}^*$ tq V accepte $\langle w, c \rangle$
- et si $w \notin L$ alors $\forall c \in \{0, 1\}^*$, V rejette $\langle w, c \rangle$

NB : $L = \{w \mid V \text{ accepte } \langle w, c \rangle \text{ pour un certain } c \in \{0, 1\}^*\}$

Un **vérifieur polynomial** est un vérifieur qui s'exécute en temps polynomial en la taille de w

→ cela entraîne que c est de taille polynomiale en $|w|$

Un vérifieur pour SUDOKU

SUDOKU

ENTRÉE : une grille G de taille n , partiellement remplie

QUESTION : G peut-elle complétée de façon valide ?

Un vérifieur pour SUDOKU

SUDOKU

ENTRÉE : une grille G de taille n , partiellement remplie

QUESTION : G peut-elle complétée de façon valide ?

$$SUDOKU = \{\langle G \rangle \mid G \text{ admet un remplissage valide}\}$$

Un vérifieur pour SUDOKU

SUDOKU

ENTRÉE : une grille G de taille n , partiellement remplie

QUESTION : G peut-elle complétée de façon valide ?

$$SUDOKU = \{\langle G \rangle \mid G \text{ admet un remplissage valide}\}$$

Vérifieur : sur entrée $\langle G, c \rangle$

Un vérifieur pour SUDOKU

SUDOKU

ENTRÉE : une grille G de taille n , partiellement remplie

QUESTION : G peut-elle complétée de façon valide ?

$$SUDOKU = \{\langle G \rangle \mid G \text{ admet un remplissage valide}\}$$

Vérifieur : sur entrée $\langle G, c \rangle$

- vérifie que c encode une grille $n \times n$

Un vérifieur pour SUDOKU

SUDOKU

ENTRÉE : une grille G de taille n , partiellement remplie

QUESTION : G peut-elle complétée de façon valide ?

$$SUDOKU = \{\langle G \rangle \mid G \text{ admet un remplissage valide}\}$$

Vérifieur : sur entrée $\langle G, c \rangle$

- vérifie que c encode une grille $n \times n$
- qui respecte le remplissage partiel de G

Un vérifieur pour SUDOKU

SUDOKU

ENTRÉE : une grille G de taille n , partiellement remplie

QUESTION : G peut-elle complétée de façon valide ?

$$SUDOKU = \{\langle G \rangle \mid G \text{ admet un remplissage valide}\}$$

Vérifieur : sur entrée $\langle G, c \rangle$

- vérifie que c encode une grille $n \times n$
- qui respecte le remplissage partiel de G
- qui respecte les contraintes d'unicité

Un vérifieur pour SUDOKU

SUDOKU

ENTRÉE : une grille G de taille n , partiellement remplie

QUESTION : G peut-elle complétée de façon valide ?

$$SUDOKU = \{\langle G \rangle \mid G \text{ admet un remplissage valide}\}$$

Vérifieur : sur entrée $\langle G, c \rangle$

- vérifie que c encode une grille $n \times n$
- qui respecte le remplissage partiel de G
- qui respecte les contraintes d'unicité

→ en **temps polynomial**

La classe NP ou NPTIME

NP = classe des langages qui admettent un **vérifieur polynomial**

Exemple : SUDOKU est dans NP

La classe NP ou NPTIME

NP = classe des langages qui admettent un **vérifieur polynomial**

Exemple : SUDOKU est dans NP

vs.

P = classe des langages **décidés** par une MT **déterministe**
en **temps polynomial**

IND-SET est dans NP

IND-SET

ENTRÉE : un graphe G et un entier $k \in \mathbb{N}$

QUESTION : G admet-il un ensemble indépendant de taille k ?

Lemme

IND-SET est dans NP

Preuve :

P vs. NP vs. EXPTIME

Théorème

$P \subseteq NP \subseteq EXPTIME$

Preuve :

P vs. NP vs. EXPTIME

Théorème

$$P \subseteq NP \subseteq EXPTIME$$

Preuve :

- $P \subseteq NP$. Soit $L \in P$:
 - il existe une MT M qui décide L en **temps polynomial**.
 - prendre M avec certificat ϵ comme vérifieur pour L .

P vs. NP vs. EXPTIME

Théorème

$P \subseteq NP \subseteq EXPTIME$

Preuve :

- $P \subseteq NP$. Soit $L \in P$:
 - il existe une MT M qui décide L en **temps polynomial**.
 - prendre M avec certificat ϵ comme vérifieur pour L .
- $NP \subseteq EXPTIME$. Soit $L \in NP$:
 - il existe un vérifieur polynomial V tq pour tout mot $w \in L$, V accepte $\langle w, c \rangle$ pour un certain certificat c
 - $|c| \leq p(|w|)$ pour un certain polynôme p
 - on construit une MT M qui, sur entrée w , énumère les certificats c de taille $\leq p(|w|)$, et exécute V sur $\langle w, c \rangle$
 - M décide L en temps $2^{\mathcal{O}(p(|w|))}$

P vs. NP vs. EXPTIME (2)

On ne sait pas si :

- $P = NP$ ou $P \neq NP$
- $NP = EXPTIME$ ou $NP \neq EXPTIME$

On sait que :

- $P \neq EXPTIME$

(cf. théorème de la hiérarchie temporelle)

NP = Non-déterministe Polynomial

MT non-déterministe

Une configuration peut avoir **plusieurs successeurs** :

$$\delta \subseteq Q \times \Sigma^k \times Q \times \Sigma^k \times \{L, R, S\}^k$$

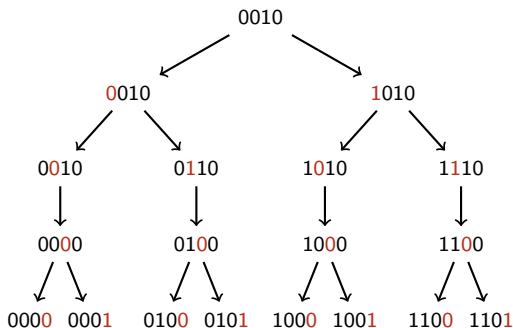
est une **relation**

Ex :

0 $\rightarrow$ 0, R

0 $\rightarrow$ 1, R

1 $\rightarrow$ 0, R



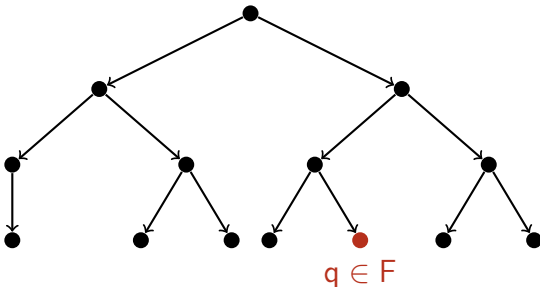
Calcul non-déterministe

L'exécution d'une M **non-déterministe** sur un mot w produit un **arbre**.

Calcul non-déterministe

L'exécution d'une M **non-déterministe** sur un mot w produit un **arbre**.

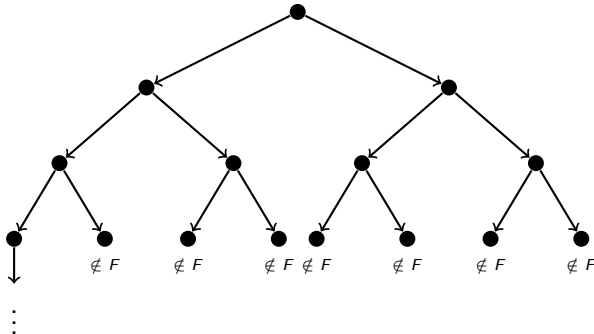
- M **accepte** w si elle s'arrête et **une** branche de l'arbre aboutit dans un état acceptant $q \in F$



Calcul non-déterministe

L'exécution d'une M **non-déterministe** sur un mot w produit un **arbre**.

- M **accepte** w si elle s'arrête et **une** branche de l'arbre aboutit dans un état acceptant $q \in F$
- M **n'accepte pas** w si **aucune** branche de l'arbre aboutit dans un état acceptant $q \in F$

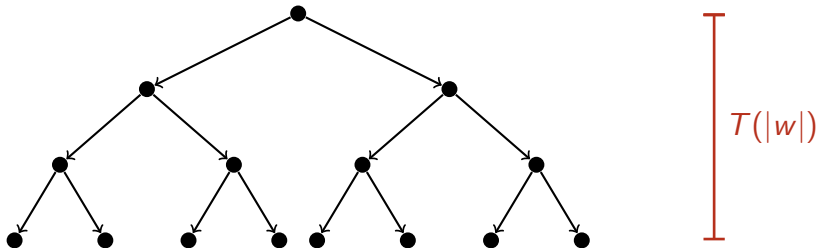


Temps de calcul

Soit M une MT non-déterministe **qui s'arrête pour tout mot d'entrée**, et $T : \mathbb{N} \rightarrow \mathbb{N}$ une fonction.

Définition

M s'exécute en temps T si pour tout mot $w \in \{0, 1\}^*$, l'arbre d'exécution de M sur w est de hauteur au plus $T(|w|)$.



NTIME

Définition

Soit $T : \mathbb{N} \rightarrow \mathbb{N}$ une fonction. Un langage L est dans la **classe** $\text{NTIME}(T)$ si et seulement si L est décidé par une MT **non-déterministe** en temps $c \cdot T$, pour une constante $c > 0$.

NTIME

Définition

Soit $T : \mathbb{N} \rightarrow \mathbb{N}$ une fonction. Un langage L est dans la **classe** $\text{NTIME}(T)$ si et seulement si L est décidé par une MT **non-déterministe** en temps $c \cdot T$, pour une constante $c > 0$.

Exemple :

$\text{NTIME}(n^2)$: langages décidables en temps quadratique **non-déterministe**

NTIME

Définition

Soit $T : \mathbb{N} \rightarrow \mathbb{N}$ une fonction. Un langage L est dans la **classe** $\text{NTIME}(T)$ si et seulement si L est décidé par une MT **non-déterministe** en temps $c \cdot T$, pour une constante $c > 0$.

Exemple :

$\text{NTIME}(n^2)$: langages décidables en temps quadratique **non-déterministe**

NB : machine de Turing est **non-déterministe**

NP = non-déterministe polynomial

Théorème

$$NP = \bigcup_{c \geq 0} NTIME(n^c)$$

Preuve :

NP = non-déterministe polynomial

Théorème

$$NP = \bigcup_{c \geq 0} NTIME(n^c)$$

Preuve :

- $\bigcup_{c \geq 0} NTIME(n^c) \subseteq NP$ Soit $L \in \bigcup_{c \geq 0} NTIME(n^c)$ décidé par M_L . Sur entrée $\langle w, c \rangle$:
 - certificat c = séquence de choix non-déterministes
 - vérifieur polynomial simule M_L sur w , guidé par c

NP = non-déterministe polynomial

Théorème

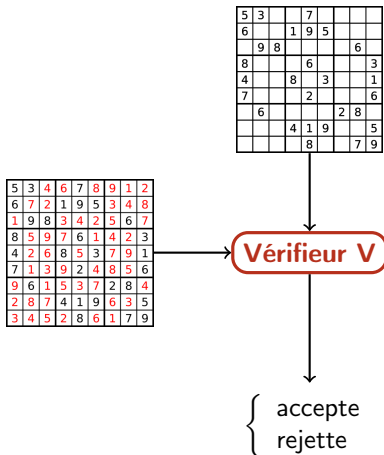
$$NP = \bigcup_{c \geq 0} NTIME(n^c)$$

Preuve :

- $\bigcup_{c \geq 0} NTIME(n^c) \subseteq NP$ Soit $L \in \bigcup_{c \geq 0} NTIME(n^c)$ décidé par M_L . Sur entrée $\langle w, c \rangle$:
 - certificat c = séquence de choix non-déterministes
 - vérifieur polynomial simule M_L sur w , guidé par c
- $NP \subseteq \bigcup_{c > 0} NTIME(n^c)$ Soit $L \in NP$ et V un vérifieur polynomial pour L . Sur entrée w :
 - M devine un certificat c avec le **non-déterminisme**
 - puis elle simule V sur $\langle w, c \rangle$

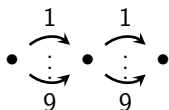
SUDOKU en temps poly. non-dét.

SUDOKU $\in NP$: vérificateur polynomial déterministe V



SUDOKU en temps poly. non-dét.

SUDOKU $\in NP$: vérificateur polynomial déterministe V



...

Diagram illustrating a node (represented by a dot) connected to a 9x9 grid representing a Sudoku puzzle. The grid contains numbers 1 through 9, with some cells highlighted in red.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Diagram illustrating a 9x9 grid representing a Sudoku puzzle. The grid contains numbers 1 through 9, with some cells highlighted in red.

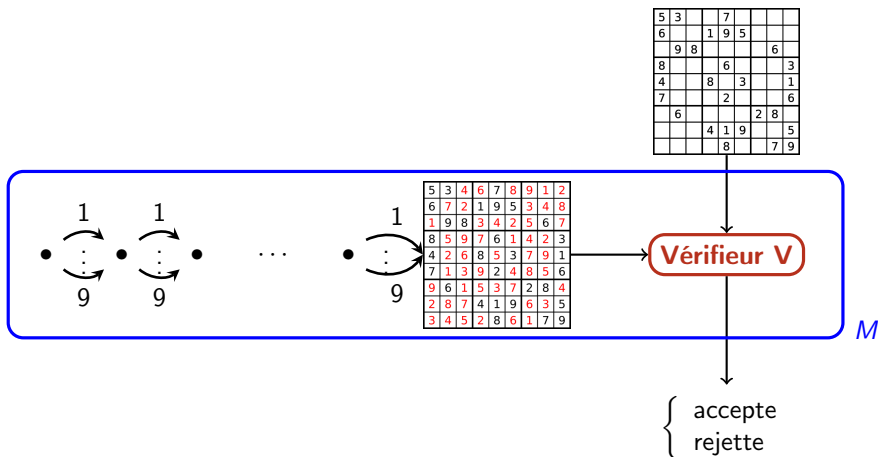
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Vérifieur V

{ accepte
rejette

SUDOKU en temps poly. non-dét.

SUDOKU $\in NP$: vérificateur polynomial déterministe V



M décide SUDOKU en temps polynomial **non-déterministe**