# Chapitre 3 : Les classes de complexité

## ENSIIE - Théorie de la complexité

Dimitri Watel (dimitri.watel@ensiie.fr)

2018

# Complexité du calcul d'une machine de Turing

## Définition

- La complexité en temps d'un calcul d'une machine de Turing est le nombre d'itérations nécessaires à la machine avant qu'elle ne s'arrête.
- La complexité en espace d'un calcul d'une machine de Turing est le nombre de cases mémoires sur lesquelles la machine a écrit avant qu'elle ne s'arrête. ≠ nb écritures (on peut écrire 2 fois au même endroit.)

La complexité est souvent calculée en fonction de la taille de $x$. ← entrée

Définir la taille d'une entrée est rarement une mince affaire.
(par exemple les graphes, voir page 13 du tableau interactif. )
Cependant, la plupart des encodages sont équivalents dans le sens où un problème de faible complexité le sera quelle que soit la manière dont on a défini la taille de l'entrée. Un contre-exemple classique est celui des entiers. (voir page 3 du tableau).

# Complexité du calcul d'une machine de Turing

Dans toute la suite, on suppose que la machine dispose de 3 symboles, 1 tête et 1 bande. Cependant les théorèmes qu'on a montré au premier cours montrent qu'on peut utiliser les autres machines sans exploser la complexité. (elle augmente nécessairement mais n'explose pas).

## Définition

Soit une machine $\mathcal{M}$ **déterministe** et un mot $x$, on pose $t(\mathcal{M}, x)$ la complexité en temps du calcul de $\mathcal{M}$ quand l'entrée est $x$. De même, on pose $s(\mathcal{M}, x)$ sa complexité en espace.

## Définition

Soit une machine $\mathcal{M}$ **non-déterministe**, un mot $x$ et une suite de choix $C$, on pose $t(\mathcal{M}, x, C)$ la complexité en temps du calcul de $\mathcal{M}$ quand l'entrée est $x$ et que la machine fait les choix $C$. De même, on pose $s(\mathcal{M}, x, C)$ sa complexité en espace.

# Complexité dans le pire cas

## Définition

Soit une machine $\mathcal{M}$ **déterministe**, la complexité (en temps) dans le pire cas de $\mathcal{M}$ est une fonction $f$ telle que

$$f(n) = \max_{x \in \{0,1,B\}^n} (t(\mathcal{M}, x))$$

## Définition

Soit une machine $\mathcal{M}$ **non-déterministe**, la complexité (en temps) dans le pire cas de $\mathcal{M}$ est une fonction $f$ telle que

$$f(n) = \max_{x \in \{0,1,B\}^n} \min_{\text{Choix } C} (t(\mathcal{M}, x, C))$$

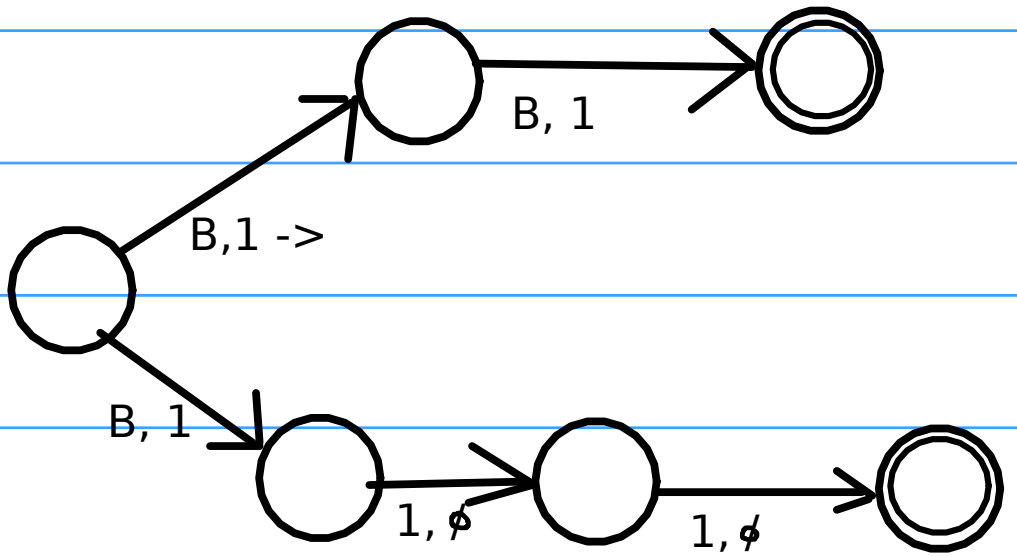
On définit de même la complexité en espace dans le pire cas.

Pourquoi cette définition de f(n) est mauvaise selon moi?

Elle dit : "la complexité d'une machine non déterministe est le temps mininum qu'il lui faut pour atteindre un état final quand l'entrée est x".

Cela pose pour moi 2 problèmes.

1) La complexité en espace. Si on condidère la machine suivante avec le mode vide :




Il y a deux choix possibles :
- C1 : soit il écrit 11 et s'arrête en 2 étapes : t(M,x,C1) = 2 ; s(M, x, C1) = 2
- C2 : soit il écrit 1 et s'arrête en 3 étapes : t(M,x,C1) = 3 ; s(M, x, C1) = 1

Avec les définition max min, on a donc une complexité en temps de 2 et en espace de 1. Ce qui est absurde car on parle de deux calculs différents. La machine non déterministe "fait toujours le bon choix" ; mais ici il n'y a pas de bon choix et elle n'est pas quantique, elle ne peut faire qu'un seul choix parmi les deux.

2) Elle est en contradiction avec la définition annexe de NP (cf page 7 du tableau) (vous aurez besoin d'avoir compris la suite du cours pour comprendre ceci)

La définition première est "NP est l'ensemble des problèmes qu'on peut résoudre avec une machine non déterministe en temps polynomial".

La seconde définition dit qu' "un problème est dans NP s'il existe un algorithme déterministe qui vérifie les solutions en temps polynomial."

Prenons le problème suivant : "soit n un entier, n est-il non premier ?"

Comme on l'a vu au premier cours: on peut utiliser l'algo non déterministe suivant:
- Choisir deux entiers k, k' > 1 de manière non déterministe.
- Vérifier si k*k' = n
- Si oui, renvoyer n non premier
- Si non, renvoyer n premier.

On a une machine non déterministe qui marche en temps polynomial. Elle est écrite sous la forme classique : construire un certificat (k et k') et le vérifier (k*k' = n).
Certificat = choix non déterministe
Vérifieur = la partie déterministe de la machine.

Imaginons maintenant la machine suivante :
- Choisir deux entiers k, k' > 1 de manière non déterministe.
- Vérifier si k*k' = n
- Si oui, renvoyer n non premier
- Si non, soit renvoyer n non premier, soit boucler pendant 2^n étapes et renvoyer n non premier.

On voit bien que cet algorithme risque de ne pas être polynomial. Mais ce n'est pas grave, la machine fait toujours le bon choix. Elle ne choisira pas de boucler. Le problème est du côté du certificat et du vérifieur.

A quoi correspondent le certificat et le vérifieur dans cette machine ?
Le certificat est en deux parties : k et k' ; et le choix de boucler ou non.
Ça peut paraître étrange de mettre le second choix dans le certificat. Cependant, comme expliqué plus haut, le vérifieur est la partie déterministe de la machine. Tout ce qui est non déterministe est dans le certificat.

On peut donc réécrire le vérifieur ainsi :

- Vérifier si k*k' = n
- Si oui, renvoyer n non premier
- Si non, si on a choisi de boucler, on boucle pendant 2^n étapes.
- Renvoyer n non premier

On voit bien que ce vérifieur n'est pas polynomial car il existe des entrées pour lesquelles la complexité est O(2^n).

On a donc une contradiction entre les deux définition, celle de la machine qui nous dit que le problème est dans NP et le second qui nous dit qu'il ne l'est peut être pas.

Pour toutes ces raisons, il me semble mieux de définir la complexité d'une machine non déterministe avec
f(n) = max max(t(M, x, C))
x C

Attention! Deux remarques. (si vous lisez encore jusqu'ici, je suis content)

- Si la démonstration précédente ne prouve pas qu'il n'existe pas une autre manière de transformer la machine en certificat/vérifieur qui rendrait les deux définitions cohérentes. Le soucis vient juste du fait que la machine ne peut pas être transformée de manière classique avec la part déterministe et la part non déterministe.

- En fait, les deux défintions de f(n) pour une machine non déterministe définissent le même ensemble NP.
En effet s'il existe une machine résolvant un problème de complexité 'max min' polynomiale alors il existe une machine de complexité 'max max' polynomiale.

Il suffit de stopper la machine quand elle tourne depuis trop longtemps et de répondre NON. En effet, elle répond nécessairement OUI en temps polynomial (sinon sa complexité 'max min' est exponentielle quand la réponse est OUI). Il suffit donc d'arrêter la machine quand ce temps polynomial est dépassé. Encore une fois si la réponse est OUI, elle choisira ce calcul polynomial pour répondre OUI.

Par exemple pour les nombres premiers, la seconde machine répond OUI en temps O(max(log(k*k'), log(n)). Si la réponse est NON et qu'elle rentre dans la boucle 2^n il suffit de l'arrêter quand O(max(log(k*k'), log(n)) est écoulé.

Si ces remarques détruisent le second argument, elle ne détruisent pas le premier.

# Complexité asymptotique dans le pire cas

## Définition

Soit une machine $\mathcal{M}$ dont la complexité dans le pire cas est $f$, on dit que $\mathcal{M}$

- a une complexité dans le pire cas bornée asymptotiquement par $g$ si $f(n) = O(g(n))$ quand $n \to +\infty$.
- a une complexité dans le pire cas minorée asymptotiquement par $g$ si $f(n) = \Omega(g(n))$ quand $n \to +\infty$.
- a une complexité dans le pire cas asymptotiquement de l'ordre de $g$ si $f(n) = \Theta(g(n))$ quand $n \to +\infty$.

Par abus de langage, on dira que $\mathcal{M}$ a une complexité $O(g(n))$ si la **complexité en temps dans le pire cas** de $\mathcal{M}$ est bornée asymptotiquement par $g$.

# Problème de décision

## Définition (formelle)

Un problème de décision $\Pi$ est constitué d'un ensemble $\mathcal{L}$, dont les éléments sont appelés *instances* ou *entrées*, et d'un ensemble $\mathcal{L}_Y \subset \mathcal{L}$ d'*instances positives*. On nomme $\mathcal{L} \backslash \mathcal{L}_Y = \mathcal{L}_N$ les *instances négatives*.

## Résoudre un problème avec une machine de Turing déterministe

Une machine de Turing **déterministe** $\mathcal{M}$ résoud $\Pi$ si

- quand $x \in \mathcal{L}_Y$, $\mathcal{M}$ accepte $x$ ;
- quand $x \in \mathcal{L}_N$ ou $x \notin \mathcal{L}$, $\mathcal{M}$ refuse $x$.

$\mathcal{M}$ s'arrête tout le temps

On sait reconnaître l'entrée.

Comme d'habitude, on supposera souvent qu'il n'est pas nécessaire de reconnaitre l'entrée.

# Complexité d'un problème de décision

**Définition : DTIME**

Soit un problème de décision $\Pi$, alors $\Pi$ a une complexité (au pire) $O(f(n))$ s'il existe une machine de Turing **déterministe** de complexité $O(f(n))$ résolvant $\Pi$. On écrit $\Pi \in \text{DTIME}(f(n))$.

# La classe de complexité P

## Définition de la classe P

Un problème de décision $\Pi$ est dit polynomial ou appartenant à la classe P si sa complexité est polynomiale. Autrement dit, il existe une machine de Turing **déterministe** qui résoud $\Pi$ en temps polynomial.

$$\mathrm{P} = \bigcup_{c \in \mathbb{N}} \mathrm{DTIME}(n^c)$$

# La classe de complexité EXPTIME

## Définition de la classe EXPTIME

Un problème de décision $\Pi$ est dit exponentiel ou appartenant à la classe EXPTIME si sa complexité est exponentielle. Autrement dit, il existe une machine de Turing **déterministe** qui résoud $\Pi$ en temps exponentiel :

$$\text{EXPTIME} = \bigcup_{c \in \mathbb{N}} \text{DTIME}(2^{(n^c)})$$

$$\text{P} \subsetneq \text{EXPTIME}$$

# Complexité en espace d'un problème de décision

**Définition : DSPACE**

Soit un problème de décision $\Pi$, alors $\Pi$ a une complexité en espace (au pire) $O(f(n))$ s'il existe une machine de Turing **déterministe** de complexité en espace $O(f(n))$ résolvant $\Pi$. On écrit $\Pi \in \text{DSPACE}(f(n))$.

# La classe de complexité PSPACE

## Définition de la classe PSPACE

Un problème de décision $\Pi$ appartient à la classe PSPACE si sa complexité en espace est polynomiale. Autrement dit, il existe une machine de Turing **déterministe** qui résoud $\Pi$ en espace polynomial.

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

Car écrire prend du temps $\Rightarrow$ complexité en espace $\leq$ complexité en temps

$\mathrm{P} \subset \mathrm{PSPACE} \subset \mathrm{EXPTIME}$ (plus tard)

$\subsetneq$ $\subsetneq$

# La classe de complexité EXPSPACE

## Définition de la classe EXPSPACE

Un problème de décision $\Pi$ appartient à la classe EXPSPACE si sa complexité en espace est exponentielle. Autrement dit, il existe une machine de Turing **déterministe** qui résoud $\Pi$ en espace exponentiel :

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

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

$$\text{P} \subset \text{PSPACE} \subset \text{EXPTIME} \subset \text{EXPSPACE}$$

# Problème de décision

## Définition (formelle)

Un problème de décision $\Pi$ est constitué d'un ensemble $\mathcal{L}$, dont les éléments sont appelés *instances* ou *entrées*, et d'un ensemble $\mathcal{L}_Y \subset \mathcal{L}$ d'*instances positives*. On nomme $\mathcal{L} \backslash \mathcal{L}_Y = \mathcal{L}_N$ les *instances négatives*.

## Résoudre un problème avec une machine de Turing non déterministe

Une machine de Turing **non déterministe** $\mathcal{M}$ résoud $\Pi$ si

- quand $x \in \mathcal{L}_Y$, $\mathcal{M}$ accepte $x$ ; $\Rightarrow \mathcal{L}_Y = \mathcal{L}_y(M)$
- quand $x \in \mathcal{L}_N$ ou $x \notin \mathcal{L}$, $\mathcal{M}$ refuse fortement $x$. $\Rightarrow \mathcal{L}_N \cup \overline{\mathcal{L}} = \mathcal{L}_N^s(M)$

# Complexité non déterministe d'un problème de décision

**Définition : NTIME**

Soit un problème de décision $\Pi$, alors $\Pi \in \text{NTIME}(f(n))$ s'il existe une machine de Turing **non déterministe** $\mathcal{M}$ de complexité $O(f(n))$ qui résoud $\Pi$.

# La classe de complexité NP

## Définition de la classe NP (non-deterministic Polynomial)

Un problème de décision $\Pi$ appartient à la classe NP s'il existe une machine de Turing **non déterministe** qui résoud $\Pi$ en temps polynomial.

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

$$\mathrm{P} \subset \mathrm{NP} \subset \mathrm{PSPACE}$$

# La classe de complexité NEXPTIME

## Définition de la classe NEXPTIME

Un problème de décision $\Pi$ appartient à la classe NEXPTIME s'il existe une machine de Turing **non déterministe** qui résoud $\Pi$ en temps exponentiel :

$$\text{NEXPTIME} = \bigcup_{c \in \mathbb{N}} \text{NTIME}(2^{n^c})$$

$$\text{EXPTIME} \subset \text{NEXPTIME} \subset \text{EXPSPACE}$$

$$\text{NP} \subsetneq \text{NEXPTIME}$$

# Complexité en espace non déterministe d'un problème de décision

**Définition : NSPACE**

Soit un problème de décision $\Pi$, alors $\Pi \in \text{NSPACE}(f(n))$ s'il existe une machine de Turing **non déterministe** $\mathcal{M}$ de complexité en espace $O(f(n))$ qui résoud $\Pi$.

# La classe de complexité NPSPACE

## Définition de la classe NPSPACE

Un problème de décision $\Pi$ appartient à la classe NPSPACE s'il existe une machine de Turing **non déterministe** qui résoud $\Pi$ en espace polynomial.

$$\text{NPSPACE} = \bigcup_{c \in \mathbb{N}} \text{NSPACE}(n^c)$$

!!!

$$\text{PSPACE} = \text{NPSPACE}$$

# La classe de complexité NEXPSPACE

## Définition de la classe NEXPSPACE

Un problème de décision $\Pi$ est appartient à la classe NEXPSPACE s'il existe une machine de Turing **non déterministe** qui résoud $\Pi$ en espace exponentiel :

$$\text{NEXPSPACE} = \bigcup_{c \in \mathbb{N}} \text{NSPACE}(2^{n^c})$$

!!!

$$\text{EXPSPACE} = \text{NEXPSPACE}$$

# La classe de complexité Co-NP

## Définition de la classe Co-NP

Un problème de décision $\Pi = (\mathcal{L}, \mathcal{L}_Y, \mathcal{L}_N)$ appartient à la classe Co-NP si $\Pi^c = (\mathcal{L}, \mathcal{L}_N, \mathcal{L}_Y)$ appartient à NP.

## Autre définition de la classe Co-NP

Un problème de décision $\Pi$ appartient à la classe Co-NP s'il existe une machine $\mathcal{M}$ **non déterministe** de complexité polynomiale qui accepte fortement les mots de $\mathcal{L}_Y$ et refuse les autres.

$$\text{P} \subset \text{NP} \cap \text{Co-NP}$$

$$\text{Co-NP} \subset \text{PSPACE}$$

## Et tant d'autres

- Hiérarchie polynomiale : $\Sigma_2, \Pi_2, \Sigma_k, \Pi_k, PH$
- Classes probabiliste : $BPP, ZPP, RP$
- Classes quantiques : $BQP, EQP$
- Classes non décidables : $RE, ALL$

`https://complexityzoo.uwaterloo.ca/Complexity_Zoo`