# TD 7 - complexité en temps
# P, NP et co-NP

## La classe P

**La classe** P**.** Cette classe contient tous les problèmes de décision que l'on peut résoudre en temps polynomial sur une machine de Turing déterministe. On a P = $\cup_{c>1}$DTIME($n^c$).

**Exercice 1** *Quelques propriétés de P*
Montrez que la classe P est close par union et par intersection. Qu'en est-il pour la complémentation ?

**Exercice 2** *vous reprendrez bien un peu de padding*
Soit une machine de Turing déterministe $\mathcal{M}$ reconnaissant un langage $\mathcal{A}$ en temps $2^{\mathcal{O}(n)}$. Notons $\mathcal{A}'$ le langage $\{\langle x, 0^{2^{|x|}}\rangle : x \in \mathcal{A}\}$ (où 0 est un symbole n'appartenant pas à l'alphabet des mots de $\mathcal{A}$). Montrez que $\mathcal{A}'$ appartient à P.

## La classe NP

**La classe** NP**.** Cette classe contient tous les problèmes de décision que l'on peut résoudre en temps polynomial sur une machine de Turing non déterministe. On a NP = $\cup_{c>1}$NTIME($n^c$).

**Exercice 3** *Quelques propriétés de NP*

- ▲ Montrez que la classe NP est close par union et par intersection.
- ◆ Montrez que P $\subseteq$ NP.

**Complémentation.** On ne sait pas si NP est close par complémentation. On note co-NP l'ensemble des langages dont le complémentaire est dans NP.

**Exercice 4** *Temps exponentiel*
Montrer que tout langage de NP est décidable par un algorithme s'exécutant en temps $2^{O(n^k)}$ pour une certaine constante $k$.

**Proposition contraposée (ou *modus tollens*).** C'est un raisonnement qui étant donnée une implication du type $A \Rightarrow B$, consiste à poser la négation du conséquent pour en déduire la négation de l'antécédant (c'est-à-dire $\neg B \Rightarrow \neg A$).

Puisque la cause d'une implication engendre la conséquence, alors l'absence de la conséquence implique automatiquement l'absence de la cause (*tollens* est le participe présent du verbe latin tollere, ôter, enlever). [merci wikipédia]

Ainsi, la proposition contraposée de « s'il pleut, alors le sol est mouillé » est « si le sol n'est pas mouillé, alors il ne pleut pas ».

**Exercice 5** *La classe co*-NP
Montrez que si NP $\neq$ co-NP alors P $\neq$ NP.

**Les classes** EXP **et** NEXP**.** On rappelle que EXP = $\cup_{c\geqslant 1}$DTIME($2^{n^c}$) et NEXP = $\cup_{c\geqslant 1}$NTIME($2^{n^c}$).

**Exercice 6** *Padding*
Nous allons montrer que si EXP $\neq$ NEXP alors P $\neq$ NP.

- ▲ Quelle est la contraposée de cette proposition ?

Supposons qu'un langage $L \in$ NEXP et que $\mathcal{M}$ soit une machine de Turing non déterministe qui décide $L$ en temps $2^{n^c}$, pour une certaine constante $c$.

Notons $L_{pad}$ le langage défini par $L_{pad} = \{\langle x, 1^{2^{|x|^c}}\rangle : x \in L\}$, où 1 est un symbole n'apparaissant pas dans $L$.

- ◆ Montrez que $L_{pad} \in$ NP.
- ⬟ Montrez que si P = NP alors $L \in$ *EXP*.
- ⬢ En déduire que si EXP $\neq$ NEXP alors P $\neq$ NP.

**Padding.** L'*argument du padding* est un outil pour montrer que si certaines classes de complexité sont égales, alors d'autres classes plus grandes le sont aussi.

**Exercice 7** *Les problèmes de la classe* NP
Pour chacune des affirmations suivantes, indiquer si elle est « vraie », « fausse », « vraie sous l'hypothèse P = NP », ou « vraie sous l'hypothèse P $\neq$ NP ».

1. Tout problème dans NP est aussi dans P.
2. Tout problème dans P est aussi dans NP.
3. Un problème dans NP n'admet pas d'algorithme déterministe polynomial pour le résoudre.
4. Il existe des problèmes dans NP qui admettent un algorithme déterministe polynomial pour les résoudre.
5. Il existe des problèmes dans NP qui n'admettent pas d'algorithme déterministe polynomial pour les résoudre.