# TD 05 – Complexité: P et EXP

**Exercice 1.** *Ordres de grandeur (1)*

Soit $f, g : \mathbb{N} \to \mathbb{N}$.

1. Donner une définition de $f \in O(g)$.
   ☞ $\exists \alpha > 0, \exists n_0, \forall n \geq n_0, f(n) \leq g(n) * \alpha$

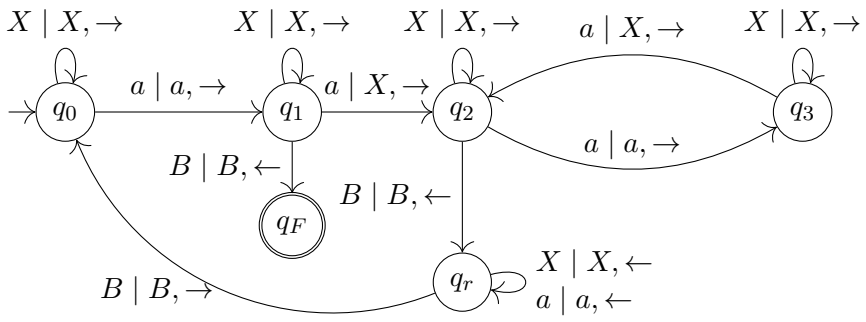
   La fonction $f$ est asymptotiquement bornée par la fonction $g$ , à un facteur multiplicatif près.

2. Classer les fonctions suivantes selon leur comportement asymptotique :
   **(a)** $n \mapsto n^2 + 15n + 10$
   **(b)** $n \mapsto 100 \log(n)$
   **(c)** $n \mapsto n^{10} + 2^n$
   **(d)** $n \mapsto 100n + n^2$
   **(e)** $n \mapsto \frac{n}{60} \log(n)$

   (de la plus petite à la plus grande)
   ☞
   — $b$ (équivalente asymptotiquement à $\log(n)$)
   — $a, d$ (équivalente asymptotiquement à $n^2$)
   — $e$ (équivalente asymptotiquement à $n \log(n)$)
   — $c$ (équivalente asymptotiquement à $2^n$)

**Exercice 2.** *Temps d'exécution*

On considère la machine de Turing $M_p$ suivante sur l'alphabet d'entrée $\Sigma = \{a\}$ :




1. Le **temps d'exécution** (noté $|M(x)|$) d'une machine $M$ sur une entrée $x$ est le nombre de transitions que prend la machine $M$ pour s'arrêter quand lancée sur le mot $x$. Calculer $|M_p(a^5)|$.
   ☞ $q_0 \to q_1 \to q_2 \to q_3 \to q_2 \to q_3$. 5 étapes.

2. Même question pour $|M_p(a^4)|$.
   ☞ $q_0 \to q_1 \to q_2 \to q_3 \to q_2 \to q_r \to q_r \to q_r \to q_r \to q_r \to q_0 \to q_1 \to q_1 \to q_2 \to q_2 \to q_r \to q_r \to q_r \to q_r \to q_r \to q_0 \to q_1 \to q_1 \to q_1 \to q_1 \to q_F$. 25 étapes.

3. Quel est le langage décidé par $M_p$ ?
   ☞ l'ensemble des mots $a^n$ où $n$ est une puissance de 2.

   Explication :
   — S'il y a un seul $a$ alors $M_p$ accepte (car on a $a^n$ avec $n = 2^0$).

— Sinon, si le nombre de $a$ est impair alors $M_p$ rejette (car un nombre impair qui n'est pas 1 n'est clairement pas une puissance de 2 ).
— Sinon $M_p$ divise le nombre de $a$ par 2 et recommence (si $n$ est une puissance de 2 alors $n/2$ aussi).

4. Le **temps de calcul au pire des cas** de $M$ d'alphabet $\Sigma$ est une fonction $t : \mathbb{N} \to \mathbb{N}$, tel que $t(n) = \max(\{\ |M(x)|\ |\ x \in \Sigma^n\ \})$.[1]
Notons que dans le cas de $M_p$ qui a un alphabet unaire, le temps de calcul au pire des cas est simplement $t_p(n) = |M_p(a^t)|$. Que vaut $t_p(n)$ avec $n = 2^k$ pour tout $k \geq 0$ ?
☞ À chaque aller-retour $M_p$ divise le nombre de $a$ par 2 jusqu'à ce qu'il n'en reste qu'un. Le nombre d'aller-retour va donc être $\log_2(n) = k$. Quand il n'en reste plus qu'un on fait un dernier aller simple. Un aller simple prend $n + 1$ étapes. Un retour prend $n + 1$ étapes. Un aller-retour prend $2n + 2$. Donc précisément $t_p(n) = (2n + 2)\log_2(n) + n + 1$...

5. Généralement, pour des raisons que l'on verra prochainement (théorème de l'accélération linéaire), on se contente de donner le temps de calcul asymptotiquement et à une constante multiplicative près. Pouvez-vous donner $t_p$ en grand O ?
☞ Quand $n$ n'est pas une puissance de 2, le programme va s'arrêter en moins de $\lceil \log(n) \rceil$ aller-retour. En supprimant les termes négligeables et les constantes multiplicatives, on obtient que $t_p \in O(n \log(n))$.

6. Pour toute fonction $f : \mathbb{N} \to \mathbb{N}$, $\mathsf{DTIME}(f)$ est la **classe de complexité en temps** $f$ qui regroupe l'ensemble des langages $L$ tels qu'il existe une machine de Turing $M$ avec $L(M) = L$ et dont le temps de calcul au pire des cas de $M$ est en $O(f)$.[2] À quelle classe de complexité appartient $L(M_p)$ ?
☞ On sait qu'elle appartient à $\mathsf{DTIME}(n \log(n))$.

**Exercice 3.** *$P$ versus $EXP$*

La classe $P$ est l'ensemble des langages reconnus en temps polynomial, c'est-à-dire

$$P = \mathsf{DTIME}(n^{O(1)}) = \bigcup_{k \in \mathbb{N}} \mathsf{DTIME}(n^k).$$

La classe EXP (ou $EXPTIME$) est l'ensemble des langages reconnus en temps exponentiel, c'est-à-dire

$$\mathsf{EXP} = \mathsf{DTIME}(2^{n^{O(1)}}) = \bigcup_{k \in \mathbb{N}} \mathsf{DTIME}(2^{n^k}).$$

Pour chacun des problèmes de décision suivants, donner un algorithme haut niveau pour le résoudre. Donner le temps de calcul au pire des cas de votre algorithme et indiquer si les problèmes appartiennent à P ou à EXP.
(On ne vous demande pas de prouver que l'algorithme est optimal).

1. **TRIANGLE**
*entrée :* Un graphe non orienté $G = (V, E)$.
*question :* Le graphe $G$ contient-il un triangle ?

(Un triangle est une clique de taille 3 : trois sommets voisins deux à deux.)
☞ Algo naïf : on fait 3 boucles imbriquées pour choisir 3 sommets $u, v, w$. Si $\{u, v\}$, $\{v, w\}$ et $\{u, w\} \in E$ alors on renvoie vrai. Sinon (en sortie de la boucle), on renvoie faux.
On pourrait optimiser l'algorithme en choisissant $v$ et $w$ dans les voisins de $u$ mais la version naïve est déjà en $O(n^3)$ avec $n$ le nombre de sommets de $n$. On peut considérer que la taille d'encodage d'un graphe est polynomial par rapport à son nombre de sommets (si $G$ est représenté par une matrice d'adjacence alors la taille de son encodage est $n^2$). Donc, le problème est dans P.

1. Il existe d'autres notions de complexité algorithmique comme le **temps de calcul en moyenne** mais on ne s'y intéressera pas dans ce cours.
2. Dans le domaine de la complexité (contrairement à la calculabilité), toutes les machines que nous considérerons s'arrêteront sur toute entrée. La question n'est plus de savoir si une machine s'arrête, mais en combien de temps elle le fait.

**2.**

**CLIQUE**
*entrée :* Un graphe non orienté $G = (V, E)$ et un entier $k$ encodé en binaire.
*question :* Le graphe $G$ contient-il une clique de taille $k$ ?

☞ Algo récursif : Si $k = 0$ alors on répond vrai. Sinon, on fait une boucle pour choisir $u \in V$ qui sera dans la clique que l'on cherche. On lance alors CLIQUE($G[N(u)]$,$k-1$) où $N(u)$ est l'ensemble des voisins de $u$ et $G[N(u)]$ est le sous-graphe induit par l'ensemble des voisins de $u$.Si la réponse est oui alors on retourne oui. Si après avoir considéré tous les $u$, aucun ne convient, on répond faux.

L'algorithme est donc en $O(n^k)$ avec $n = |V|$ ce qui est exponentiel. CLIQUE est donc dans EXP. Notons que nous n'avons pas prouvé que le problème n'est pas dans P. (et pour cause : on verra plus tard que la réponse à cette question est la réponse à la question à un million de dollars $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$).

**3.**

**FORTE CONNEXITÉ**
*entrée :* Un graphe non-orienté $G = (V, E)$.
*question :* Le graphe $G$ est-il connexe ?

*Le graphe $G = (V, E)$ est connexe si pour tous sommets $u, v \in V$, il existe un chemin dans $G$ entre $u$ et $v$.*
☞

**Entrée:** Un graphe non orienté G = (V, E)
**Sortie:** Vrai si G est connexe, Faux sinon
1: Initialiser un ensemble de sommets visités $visited$ vide
2: Choisir un sommet de départ $v_0$ parmi les sommets de G
3: Ajouter $v_0$ à $visited$
4: Créer une pile $stack$ et empiler $v_0$
5: **tant que** $stack$ n'est pas vide **faire**
6: Dépiler un sommet $v$ de $stack$
7: **pour** chaque voisin $u$ de $v$ dans G **faire**
8: **si** $u$ n'est pas dans $visited$ **alors**
9: Ajouter $u$ à $visited$
10: Empiler $u$ dans $stack$
11: **fin si**
12: **fin pour**
13: **fin tant que**
14: **si** $visited$ contient tous les sommets de $V$ **alors**
15: **retourne** Vrai
16: **sinon**
17: **retourne** Faux
18: **fin si**

Temps de calcul au pire des cas en $O(n^2)$ (chaque sommet $v$ ne sera contenu qu'une seule fois dans la *stack* et tous ses voisins sont au pire tous les sommets de $G$). Donc c'est dans P.

**4.**

**Arrêt en $k$ étapes.**
*entrée :* Une machine de Turing $M$, un mot $w$ et un entier $k$ encodé en binaire.
*question :* $|M(w)| \leq k$ ?

On suppose que si $M(w) \uparrow$ alors $|M(w)| = \inf$.

☞ On simule la machine $M$ sur l'entrée $w$. Au pire des cas (quand $M$ ne s'arrête pas ou s'arrête exactement en $k$ étapes), notre algorithme aura un temps d'exécution en environ $O(k)$. Comme $k$ est encodé en binaire, le temps est exponentiel par rapport à l'entrée et le problème est dans EXP. On verra plus tard que ce problème n'est pas dans P.

**Exercice 4.** *Vrai ou Faux* P *et* EXP

1. Un problème appartenant à P n'appartient pas à EXP.
   ☞ Faux, $\mathsf{P} \subseteq \mathsf{EXP}$
2. Un problème appartenant à EXP n'appartient pas à P.
   ☞ Faux, connexité par exemple est dans P et dans EXP.
3. Il existe des problèmes appartenant à P mais pas à EXP.
   ☞ Faux, $\mathsf{P} \subseteq \mathsf{EXP}$

**4.** Il existe des problèmes appartenant à EXP mais pas à P.

☞ Vrai, arrêt d'une machine de Turing en temps $k$ par exemple. Mais ça, on le prouvera au prochain TD.