Aix-Marseille Université
Théorie Algorithmique des Nombres
`http://iml.univ-mrs.fr/~kohel/tch/M2-TAN/`

Complexité algorithmique

---

1. **a.** Montrer que g est dans $O(f)$ si et seulement si $\limsup_{x\to\infty} \dfrac{|g(x)|}{f(x)} = c < \infty$.

b. Montrer que $O(f)$ est un espace vectoriel réel.

2. **a.** Montrer que g est dans $o(f)$ si et seulement si $\limsup_{x\to\infty} \dfrac{|g(x)|}{f(x)} = 0$.

b. Montrer que $o(f)$ est un sous-espace vectoriel réel de $O(f)$.

3. Soit f et g des fonctions positives. Montrer que g est dans $O(f)$ si et seulement si $O(g) \leq O(f)$.

4. Si $g \in O(f)$, alors $O(g) + O(f) = O(f)$.

5. Pour des fonctions $f_1, \ldots, f_r$ positives, et scalaires réels $\alpha_1, \ldots, \alpha_r$ positifs, montrer les identités

a. $\sum_{i=1}^r O(f_i) = O(\sum_{i=1}^r \alpha_i f_i)$ et en particulier, $\sum_{i=1}^r O(f_i) = O(\sum_{i=1}^r f_i)$.

b. $\prod_{i=1}^r O(f_i) = O(\prod_{i=1}^r f_i)$.

Conclure que $(O(f_1)+O(g_1))(O(f_2)+O(g_2)) = O((f_1+g_1)(f_2+g_2))$ (*cf.* exercice 7).

6. Définir la somme $(f_1 + O(g_1)) + (f_2 + O(g_2))$ et démontrer que

$$(f_1 + O(g_1)) + (f_2 + O(g_2)) = (f_1 + f_2) + O(g_1 + g_2).$$

7. Définir le produit $(f_1 + O(g_1))(f_2 + O(g_2))$ et démontrer que

$$\begin{aligned}(f_1 + O(g_1))(f_2 + O(g_2)) &\subseteq f_1 f_2 + O(f_1 g_2 + f_2 g_1 + g_1 g_2) \\ &= (f_1 + O(g_1))(f_2 + O(g_2)) + O(g_1 g_2).\end{aligned}$$

Lesquelles des fonctions f_1, f_2, g_1, g_2 doivent être positives ?
Si $g_1 \in O(f_1)$ ou $g_2 \in O(f_2)$, alors

$$(f_1 + O(g_1))(f_2 + O(g_2)) = f_1 f_2 + O(f_1 g_2 + f_2 g_1 + g_1 g_2),$$

et également si $f_1 \in O(g_1)$ où $f_2 \in O(g_2)$ (dans ce dernier cas pour des raisons triviales car $f_1 + O(g_1) = O(g_1)$ ou $f_2 + O(g_2) = O(g_2)$).

8. Trouver une expression pour la classe $\big(x + O(x^{1/2})\big)\big(x + O(\log(x))\big)^2$.

9. Montrer qu'il existe des fonctions f et $g : \mathbb{R}_{>0} \to \mathbb{R}$ telles que $g \notin O(f)$ et $f \notin O(g)$.
Indication. On peut construire des fonctions comme combinaison linéaire avec un terme dominant multiplié par des fonctions périodiques $\sin^2(x)$ et $\cos^2(x)$.

10. Répéter exercice 5 en remplaçant 'O' avec 'o'.

11. Supposer que $f, g : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$.

a. Montrer que $f \sim g$ si et seulement si $f/g \in 1 + o(1)$.

b. Montrer que $f \sim g$ si et seulement si $f - g \in o(g)$.

Arithmétique et algorithmique

1. Soit $m = 100110111_2$ et $n = 10111_2$ (en représentation binaire). Trouver $m + n$ en précisant les étapes pour leur addition.

2. Répéter l'exercice précédent avec $m = 2^{16} - 1$ et $n = 1$ (cf. **3.b.** ci-dessous).

3. Exprimer la complexité de l'addition de m et n en fonction de la taille x des données, en particulier, démontrer que cette complexité est $O(x)$.

a. Plus précisement, décrire un algorithme et compter le nombre d'opérations binaires en termes de $\lfloor \log_2(m) \rfloor + 1$ et $\lfloor \log_2(n) \rfloor + 1$, pour le problème suivant :
Entrées : m et n entiers ($x = \lfloor \log_2(m) \rfloor + \lfloor \log_2(n) \rfloor + 2$) **Sortie :** $m + n$

b. Supposer que m, n, et $m+n$ s'écrivent en représentation binaire. Est-il possible de décrire un algorithme pour $m \mathrel{+}= n$ avec complexité $O(\log(n))$?

4. Pour des polynômes f et g dans $\mathbb{F}_2[x]$, montrer qu'il existe un algorithme pour le calcul $f \mathrel{+}= g$ dans la classe $O(\deg(g))$.

5. Compter le nombre d'opérations pour la multiplication en utilisant la formule

$$\left(m = \sum_{i=0}^{k} m_i 2^i,\ n = \sum_{j=0}^{\ell} n_j 2^j \right) \longmapsto mn = \sum_{i=0}^{k+\ell} \left(\sum_{j=0}^{i} m_{i-j} n_j \right) 2^i$$

en termes de $k = \lfloor \log_2(m) \rfloor + 1$ et $\ell = \lfloor \log_2(n) \rfloor + 1$.

6. Démontrer que la complexité de la multiplication de deux nombres n, m de tailles dans $O(x)$, est dans $O(x^2)$, en utilisant la multiplication naïve avec retenues.

7. Décrire un algorithme pour la reduction $(n, m) \mapsto n \bmod m$ avec complexité

$$O(\log(m) \log(n/m)).$$

Indication. Étant donnés $n = n_r \dots n_1 n_0$ et $m = m_s \dots m_1 m_0$, la première étape est la soustraction $n \mathrel{-}= 2^{r-s} m$:

n_r	n_{r-1}	$\dots$	n_{r-s}	n_{r-s-1}	$\dots$	n_0
m_s	m_{s-1}	$\dots$	m_0	0	$\dots$	0
0	$*$	$\dots$	$*$	n_{r-s-1}	$\dots$	n_0

ou $n \mathrel{-}= 2^{r-s-1} m$:

n_r	n_{r-1}	$\dots$	n_{r-s-1}	n_{r-s-2}	$\dots$	n_0
0	m_s	$\dots$	m_0	0	$\dots$	0
0	$*$	$\dots$	$*$	n_{r-s-2}		n_0

Le nombre de coordonnées changées est au maximum $s+2$, et le résultat a au moins un bit de moins. En itérant, montrer qu'on obtient, après $O(\log(n/m))$ étapes, la complexité $O(\log(m) \log(n/m))$.

8. Décrire un algorithme pour la fonction $n \mapsto n \bmod 2$ et comparer sa complexité avec l'algorithme de l'exercice précédent pour n grand.