Aix-Marseille Université
Préparation à l'agrégation: Option C
`https://www.i2m.univ-amu.fr/perso/david.kohel/tch/Agreg/`

Complexité

Exercices

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.

Solution.

a. *Comme $f(x) > 0$, la fonction $h(x) = |g(x)|/f(x)$ est bien-définie. Supposons que $c = \limsup h(x)$ existe. Alors pour tout $a = c+\epsilon$ il existe x_0 tel que $h(x) < a$ pour $x > x_0$. Par conséquent $g(x) < af(x)$ et donc g est dans $O(f)$.*

Supposons que g est dans $O(f)$. Par la définition de $O(f)$, il existe $x_0 > 0$ et $a > 0$ tel que $0 \leq h(x) \leq a$ pour tout $x > x_0$. Par conséquent, le supremum $c_n \geq 0$ de $h([n, \infty[)$ existe pour tout $n > x_0$ et la $\limsup$ de $h(x)$ vaut la limite de la suite (c_n).

Attention : On utilse le résultat suivant : Si S est un sous-ensemble majoré de $\mathbb{R}$, alors le supremum de S existe.

b. *Il suffit de montrer que $O(f)$ est close pour les opérations d'addition et multiplication scalaire.*

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

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

Solution.

a. *L'équivalence est une conséquence évidente de la définition of $o(f)$ et de la limite. En effet $g \in o(f)$ est équivalent à les énoncés suivants :*
- *$\forall\, \varepsilon > 0,\ \exists\, x_0 > 0$ tel que $\forall\, x > x_0,\ |g(x)| < \varepsilon f(x)$,*
- *$\forall\, \varepsilon > 0,\ \exists\, x_0 > 0$ tel que $\forall\, x > x_0,\ |g(x)|/f(x) < \varepsilon$,*
- *$\forall\, \varepsilon > 0,\ \limsup_{x\to\infty} |g(x)|/f(x) < \varepsilon$,*
- *$\limsup_{x\to\infty} |g(x)|/f(x) = 0$,*
- *$\lim_{x\to\infty} g(x)/f(x) = 0$.*

b. *Par les exercices **1. a.** et **2. a.**, $o(f)$ est un sous-ensemble de $O(f)$, et par les propriétés de limites, $o(f)$ est stable pour les opérations d'addition et de multiplication par scalaires, donc un sous-espace vectoriel.*

3. Soient f et g des fonctions positives.

a. Montrer que $O(g) \leq O(f)$ si et seulement si g est dans $O(f)$.

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

Solution.

a. *L'implication si $O(g) \leq O(f)$, alors $g \in O(f)$ est évidente. Reciproquement, si $g \in O(f)$, pour tout $h \in O(g)$, on a*

$$\left(\limsup_{x\to\infty} \frac{|h(x)|}{f(x)}\right) \leq \left(\limsup_{x\to\infty} \frac{|h(x)|}{g(x)}\right)\left(\limsup_{x\to\infty} \frac{g(x)}{f(x)}\right) < \infty,$$

et donc $h \in O(f)$. Il suit que $O(g) \leq O(f)$.

b. *Le résultat est un corollaire la partie précédente, car $O(g) + O(f) = O(f)$.*

On note, par conséquent, que si $a_n x^n + \cdots + a_1 x + a_0$ est un polynôme avec $a_n > 0$, que $O(a_n x^n + \cdots + a_1 x + a_0) = O(x^n)$.

4. Pour des fonctions $f_1, \ldots, f_t$ positives, et scalaires réels $\alpha_1, \ldots, \alpha_t$ positifs, montrer les identités

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

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

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 **6**).

Solution. *On donne une solution en utilisant des choix astucieux pour la décomposition (additive out multiplicative) de $h(x)$, en utilisant des fonctions caracteristiques. On observe que les inclusions*

$$\sum_{i=1}^t O(f_i) \subseteq O(\sum_{i=1}^t \alpha_i f_i) \text{ et } \prod_{i=1}^t O(f_i) \subseteq O(\prod_{i=1}^t f_i)$$

sont évidentes, et il suffit de démontrer les inclusions reciproques.

a. *On pose $f = \sum_{i=1}^t \alpha_i f_i$ et prend $h \in O(f)$. Par rapport à l'expression additive pour f, on définit une fonction caracteristique $e_i = \alpha_i f_i / f$, et on pose $h_i = e_i h$. On a évidemment $\sum_{i=1}^t e_i = 1$ et par conséquent,*

$$\sum_{i=1}^t h_i = h.$$

Il ne reste qu'à démontrer que $h_i \in O(f_i)$. Par définition de h, et l'exercice **1**, *on a*

$$\limsup_{x\to\infty} \frac{|h(x)|}{f(x)} = c, \text{ d'où } \limsup_{x\to\infty} \frac{|h_i(x)|}{f_i(x)} = \limsup_{x\to\infty} \alpha_i \frac{|h(x)|}{f(x)} = \alpha_i c < \infty.$$

Par conséquent, $h = \sum_{i=1}^t h_i \in \sum_{i=1}^t O(f_i)$ et l'inclusion $O(f) \subseteq \sum_{i=1}^t O(f_i)$ est vérifiée.

b. *On pose $f = \prod_{i=1}^t f_i$ et prend $h \in O(f)$. Par le lemme suivant, on peut supposer que $h(x) = |h(x)|$ pour tout $x \in \mathbb{R}_{>0}$.*

Lemma. *Une function h est dans $O(f)$ si est seulement si $|h| \in O(f)$.*

Par rapport l'expression multiplicative pour f, on définit $h_i = f_i \sqrt[n]{h/f}$ en notant que $\prod_{i=1}^{t} h_i = h$. Il ne reste qu'à démontrer que $h_i \in O(f_i)$. Par définition de h et de h_i,

$$\limsup_{x\to\infty} \frac{|h(x)|}{f(x)} = c, \text{ d'où } \limsup_{x\to\infty} \frac{|h_i(x)|}{f_i(x)} = \limsup_{x\to\infty} \sqrt[n]{\frac{|h(x)|}{f(x)}} = \sqrt[n]{c} < \infty.$$

Par conséquent, $h = \prod_{i=1}^{t} h_i \in \prod_{i=1}^{t} O(f_i)$ et l'inclusion $O(f) \subseteq \prod_{i=1}^{t} O(f_i)$ est vérifiée.

Remarque. *Dans ce cadre multiplicative on définit les fonctions caracteristiques, par rapport au produit pour f, dl'être $u_i = f_i \sqrt[n]{1/f}$ avec la propriété $\prod_{i=1}^{t} u_i = 1$. On a donc $h_i = u_i \sqrt[n]{h}$.*

Solution. *On donne une deuxième solution, en décomposant l'intervalle $]0, \infty[$ en morceaux I_j où la fonction f_j est dominante. Il suffit de démontrer les résultats pour $t = 2$. Le résultat pour r général suit par itération. On remarque que $O(\alpha_i f_i) = O(f_i)$, et donc on peut supposer également que $\alpha_i = 1$.*

a. *La direction $O(f_1) + O(f_2) \subseteq O(f_1 + f_2)$ est évidente. Soit $h \in O(f_1 + f_2)$. On pose*

$$I_1 = \{x \in \mathbb{R}_{>0} \mid f_1(x) \geq f_2(x)\},$$
$$I_2 = \{x \in \mathbb{R}_{>0} \mid f_1(x) < f_2(x)\}.$$

et pour $i \in \{1, 2\}$, on pose

$$h_i(x) = \begin{cases} h(x) & \text{si } x \in I_i, \\ 0 & \text{sinon.} \end{cases}$$

Alors $h_1(x) + h_2(x) = h(x)$ et on observe que $h_i(x) \in O(f_i(x))$. En effet, si $h(x) \leq c(f_1(x) + f_2(x))$ pour $x > x_0$, alors

$$h_i(x) = h(x) \leq c(f_1(x) + f_2(x)) \leq 2cf_i(x) \text{ si } x \in I_i,$$

et $h_i(x) = 0 < 2cf_i(x)$ sinon. Il suit que $h_i(x) \in O(f_i(x))$.

b. *L'inclusion $O(f_1)O(f_2) \subseteq O(f_1 f_2)$ est évidente. Pour l'inclusion inverse, pour $h \in O(f_1 f_2)$, on pose*

$$h_1(x) = \begin{cases} f_1(x) & \text{si } x \in I_1, \\ h(x)/f_2(x) & \text{si } x \in I_2. \end{cases} \qquad h_2(x) = \begin{cases} h(x)/f_1(x) & \text{si } x \in I_1, \\ f_2(x) & \text{si } x \in I_2. \end{cases}$$

Par construction $h_1 h_2 = h$, et on vérifie facilement que $h_i \in O(f_i)$.

L'identité $(O(f_1) + O(g_1))(O(f_2) + O(g_2)) = O((f_1 + g_1)(f_2 + g_2))$ suit des deux parties de cet exercice :
— *$O(f_i) + O(g_i) = O(f_i + g_i)$, et*
— *$O(f_1 + g_1)O(f_2 + g_2) = O((f_1 + g_1)(f_2 + g_2))$.*

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

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

L'ensemble $f_i + O(g_i)$ porte quelle structure ?

Solution. *En utilisant la propriété de l'exercice* **4** *que* $O(g_1) + O(g_2) = O(g_1 + g_2)$*, l'expression est évident. Les ensembles* $f_i + O(g_i)$ *portent une structure de sous-espace affine de l'espace vectoriel des fonctions* $\mathbb{R}_{>0} \to \mathbb{R}$*, dirigé par* $O(g_i)$*.*

6. 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 ?

Solution. *Les fonctions* g_1 *et* g_2 *doivent être positives, mais aussi l'expression* $f_1 g_2 + f_2 g_1 + g_1 g_2$*, pour que les classes* O *soient bien-définies. Dans l'exercise, il suffit de supposer que* f_1 *et* f_2 *sont aussi des fonctions positives. On fait cette hypothèse dans la suite.*

Pour deux sous-ensembles S_1 *et* S_2 *d'un ensemble pour lequel la multiplication et bien-définie, on pose*

$$S_1 S_2 = \{a_1 a_2 \mid a_1 \in S_1, a_2 \in S_2\}.$$

Par expansion du produit $(f_1 + O(g_1))(f_2 + O(g_2))$*, on a*

$$(f_1 + O(g_1))(f_2 + O(g_2)) \subseteq f_1 f_2 + f_1 O(g_2) + f_2 O(g_1) + O(g_1) O(g_2).$$

Sous l'hypothèse que $f_1(x), f_2(x) > 0$*, on* $f_i O(g_j) = O(f_i g_j)$*, et cette dernière expression est égal à*

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

Une élément de cet ensemble est de la forme

$$f_1 f_2 + f_1 h_2 + f_2 h_1 + h_1' h_2' = (f_1 + h_1)(f_2 + h_2) + (h_1' h_2' - h_1 h_2),$$

qui est dans $(f_1 + O(g_1))((f_2 + O(g_2)) + O(g_1 g_2)$*, car*

$$h_1' h_2' - h_1 h_2 \in O(g_1 g_2) + O(g_1 g_2) = O(g_1 g_2).$$

On a donc

$$f_1 f_2 + O(f_1 g_2 + f_2 g_1 + g_1 g_2) \subseteq (f_1 + O(g_1))(f_2 + O(g_2)) + O(g_1 g_2).$$

Inversement un élément de $(f_1 + O(g_1))(f_2 + O(g_2)) + O(g_1 g_2)$ *est de la forme*

$$\begin{aligned}(f_1 + h_1)(f_2 + h_2) + h_1' h_2' &= f_1 f_2 + f_1 h_2 + f_2 h_1 + (h_1 h_2 + h_1' h_2') \\ &\in f_1 f_2 + O(f_1 g_2) + O(f_2 g_1) + O(g_1 g_2) \\ &= f_1 f_2 + O(f_1 g_2 + f_2 g_1 + g_1 g_2).\end{aligned}$$

On a donc l'égalité

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

La prochaine partie concerne des conditions sous lesquelles on a égalité sans le terme d'erreur $O(g_1 g_2)$ à gauche.

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

Solution. *Évident. Si $g_1 \in O(f_1)$ ou $g_2 \in O(f_2)$, on note que*

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

si $f_1 \in O(g_1)$, on a $O(f_1 g_2 + f_2 g_1 + g_1 g_2) = O(g_1(f_2 + g_2))$. et si $f_2 \in O(g_2)$, alors $O(f_1 g_2 + f_2 g_1 + g_1 g_2) = O((f_1 + g_1) g_2)$.

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

Solution. *On calcule d'abord $\big(x + O(\log(x))\big)^2 = x^2 + O(x \log(x))$ car $O(\log(x)^2)$ est negligible $O(\log(x)^2) < O(x \log(x))$. Ensuite, le produit final est*

$$\big(x + O(x^{1/2})\big)\big(x^2 + O(x \log(x))\big) = x^3 + O(x^{5/2}),$$

car $O(x^2 \log(x)) < O(x^{5/2})$.

8. 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 $\cos^2(x)$.

Solution. *On peut choisir des fonctions f et g de la forme $a(x) + b(x)\sin^2(x)$ et $a(x) + b(x)\cos^2(x)$ où la fonction dominante est $b(x)$, i.e. $a(x) \in o(b(x))$.*

9. Répéter exercice **4** en remplaçant 'O' par 'o'.

Solution. *Les mêmes arguemnts marchent pour petit o, soit en utilisant les fonctions caracteristiques soit les décomposition en intervalles.*

10. Supposer que $f, g : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$. Montrer l'équivalence des expressions suivantes

a. $f \sim g$, **b.** $f - g \in o(g)$, **c.** $f/g \in 1 + o(1)$, **d.** $f/g \sim 1$.

Solution. *Ces équivalences suivent des propriétés additives des limites et multiplicatives, à partir de la définition de $f \sim g$:*

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = 1.$$

Complexité des programmes

On donne un exercice qui sert comme un mini-texte de modélisation. On peut envisager de préparer un exposé court autour de cet exercice, en intégrant des expériences et experimentation en Python/Sage.

Exercice de modélisation. Soit donné les algorithmes ci-dessous, avec donnée d'entrée n.

a.

```
def runtime1(n):
    m = 0
    for i in range(n):
        m += 1
    return m
```

b.

```
def runtime2(n):
    m = 0
    for i in range(n):
        m += i
    return m
```

c.

```
def runtime3(n):
    m = 0
    for i in range(n):
        m *= 2
        m += 1
    return m
```

d.

```
def runtime4(n):
    m = 0
    for i in range(n):
        m *= 2
        m += i
    return m
```

Déterminer, pour chaque programme, une expression en forme de somme pour la valeur de sortie, la taille (en bits) de cette valeur, et la complexité de l'algorithme, en fonction de $\log_2(n)$. Vérifier que cette complexité correspond bien au temps de calcul à l'ordinateur.

Indication. Pour le programme `runtime4`, on peut formuler et démontre l'identité suivante pour l'expression de la somme représentant la valeur de sortie.

Lemme.

$$\sum_{i=0}^{n-1} i\, 2^{n-i-1} = 2^n - n - 1.$$

Solution. *Les valuers de sortie sont* ***a.*** n, ***b.*** $n(n-1)/2$, ***c.*** 2^n-1, ***d.*** 2^n-n-1, *et donc leurs tailles sont* ***a.*** $\sim \log_2(n)$, ***b.*** $\sim 2\log_2(n)$, ***c.*** $\sim n$, ***d.*** $\sim n$.

Remarque (notation). *On rappelle que les complexités sont des fonctions la taille d'entrée $x \sim \log_2(n)$. On utilise la notation $T(x) \sim f(x)$ au lieu de $T(x) \in O(f(x))$ pour donner le coefficient du terme dominant ; précisement $T(x) \in f(x) + o(f(x))$.*

Remarque (modélisation). *On rappelle dans le cadre de modélisation des mathématiques, qu'on peut utiliser Sage pour des expériences afin de formuler des conjectures. On note que la somme calculée par le programme* `runtime4` *est de la forme :*

$$\sum_{i=0}^{n-1} i\, 2^{n-i-1} = \frac{1}{2^{n-1}} \sum_{i=0}^{n-1} \frac{i}{2^i}.$$

L'expression $\sum_{i=0}^{\infty} i/2^i$ a une convergence géométrique, donc la somme

```
sage: for n in range(1,17):
....:     print "%2s: %s" % (n, sum([ 1.0*i/2^i for i in range(n) ]) )
```

converge rapidement à sa limite.

```
 1: 0.000000000000000          9: 1.96093750000000
 2: 0.500000000000000         10: 1.97851562500000
 3: 1.00000000000000          11: 1.98828125000000
 4: 1.37500000000000          12: 1.99365234375000
 5: 1.62500000000000          13: 1.99658203125000
 6: 1.78125000000000          14: 1.99816894531250
 7: 1.87500000000000          15: 1.99902343750000
 8: 1.92968750000000          16: 1.99948120117188
```

On peut formuler la conjecture (en alliant plus loin) que $\sum_{i=0}^{\infty} i/2^i = 2$*, ce qui donne le terme dominant* $\sum_{i=0}^{n-1} i\,2^{n-i-1} \sim 2^n$ *Ensuite le calcul exact :*

```
sage: for n in range(9,17):
....:     s = sum([ i*2^(n-i-1) for i in range(n) ])
....:     print "%2s: %s" % (n, 2^n - s)
```

```
 9: 10
10: 11
11: 12
12: 13
13: 14
14: 15
15: 16
16: 17
```

permet de formuler comme conjecture le lemme ci-dessus : $\sum_{i=0}^{n-1} i\,2^{n-i-1} = 2^n - n - 1$.

On détermine ensuite les complexités de ces algorithmes.

- ***a.*** *À l'itération* i *il faut calculer l'addition* $i+1$*, dont complexité est* $\log_2(i)$*, donc, on a* $\sum_{i=1}^{n} \log_2(i) \sim n \log_2(n)$.
- ***b.*** *À l'itération* i *il faut calculer l'addition* $i(i-1)/2 + i$*, dont complexité est* $\sim 2\log_2(i)$*, donc on a* $\sum_{i=1}^{n} 2\log_2(i) \sim 2n\log_2(n)$.
- ***c.*** *Ici, une implémentation optimal permet de calculer la multiplication par 2 sans recopier* m*, et l'addition d'un n'est qu'une opération d'un bit. Donc le temps de calcul est exactement* n *opérations, le nombre de bits de la sortie* $(= 2^n - 1)$.
- ***d.*** *On suppose, comme précédemment que la multiplication par 2 est gratuit. Pour l'addition* $m + i$*, l'entier* $m\,(= 2^i - 2i - 2)$ *est plus grand, donc l'addition prend* $\log_2(m) \sim i$ *opérations binaires, et la complexité totale est* $\sim \sum_{i=0}^{n-1} i = n(n-1)/2$.

En conclusion, les complexités sont ***a.*** $\sim n\log_2(n)$, ***b.*** $\sim 2n\log_2(n)$, ***c.*** $\sim n$, ***d.*** $\sim n^2/2$.

On peut vérifier une croissance linéaire, en $r = \log_2(n)$*, ou exponentiel (linéaire ou quadratique en* n*), avec des boucles comme les suivants :*

```
sage: for r in range(9,17):
....:     print "r =", r
....:     %time runtime1(2^r)
....:     %time runtime2(2^r)
```

```
sage: for n in range(128,160):
....:     print "n =", n
....:     %time runtime3(n)
....:     %time runtime4(n)
```