IFT2010 Hiver 2006 — Solutions au Devoir 1

Miklós Csűrös

16 février 2006

1 Oh, la notation

1.a

Une fonction quelconque $f(n) \in O(n^3) \setminus O(n^2)$ ssi $f(n) \in O(n^3)$ mais $f(n) \notin O(n^2)$. Si $f(n) \in o(n^3) \subset O(n^3)$ alors $f(n) \notin \Theta(n^3)$. On doit juste trouver une fonction $f(n) \in o(n^3)$ telle que $f(n) \notin O(n^2)$. Il y a beacoup de fonctions de cette sorte : par exemple

$$n^2 \lg n, n^{\frac{5}{2}}, \frac{n^3}{\lg \lg n}, \dots$$

Donc la réponse est que $O(n^3) \setminus O(n^2) \neq \Theta(n^3)$. [Par contre, $O(n^3) \setminus o(n^3) = \Theta(n^3)$.]

1.b

Si $f(n) \notin O\big(g(n)\big)$, alors pour tout $c > 0$ et N, il existe $n \geq N$ tel que $f(n) > c \cdot g(n)$. D'une façon équivalente [démontrez l'équivalence comme exercice], pour tout $c > 0$, il existe un nombre infini de n t.q. $f(n) > c \cdot g(n)$. Si $g(n) \notin O\big(f(n)\big)$, alors pour tout $c' > 0$, il existe un nombre infini de n t.q. $g(n) > c' \cdot f(n)$. Est-ce qu'il existe une paire de fonctions f, g t.q. f «dépasse» g fréquemment et g «dépasse» f fréquemment ? Bien sûr ! Par exemple,

$$f(n) = \begin{cases} 1 & \text{si } n \text{ est paire}; \\ n & \text{si } n \text{ est impaire}, \end{cases} \qquad g(n) = \begin{cases} n & \text{si } n \text{ est paire}; \\ 1 & \text{si } n \text{ est impaire}, \end{cases}$$

1.c

Si $f(n) = O(n \log n)$, alors il existe une constante $c > 0$ et un seuil N t.q. $f(n) \leq c \cdot n \lg n$ pour tout $n \geq N$. On a donc

$$g(n) = f\big(\lceil \sqrt{n} \rceil\big) \leq c \cdot \lceil \sqrt{n} \rceil \lg \lceil \sqrt{n} \rceil$$

si $\lceil \sqrt{n} \rceil \geq N$. Sans l'arrondi au plafond, on serait fini ici :

$$c \cdot \sqrt{n} \lg \sqrt{n} = \frac{c}{2} \sqrt{n} \lg n.$$

Avec l'arrondi, on se sert d'une borne comme $\lceil \sqrt{n} \rceil \leq 2\sqrt{n}$. (Si on écrit $n = m^2 - a$ avec $a < 2m - 1$, alors $\lceil \sqrt{n} \rceil = m$ et $\sqrt{n} > m - 1$, donc $\lceil \sqrt{n} \rceil / \sqrt{n} < 1 + 1/(m-1) \leq 2$ pour $m \geq 2$.)

$$g(n) \leq c \cdot \lceil \sqrt{n} \rceil \lg \lceil \sqrt{n} \rceil \leq 2c\sqrt{n}\Big(1 + \frac{\lg n}{2}\Big) \leq 2c\sqrt{n} \lg n$$

quand $1 \leq \frac{\lg n}{2}$, ou $4 \leq n$. On a donc que si $n \geq \max\{N, 4\}$, $g(n) \leq c'\sqrt{n} \lg n$ (avec $c' = 2c$) et ainsi $g(n) \in O(\sqrt{n} \log n)$. [Notez que cette borne est beaucoup meilleure que $O(n \log n)$, comme $\sqrt{n} \lg n \in o(n \log n)$.]

1.d

Pour avoir une bonne idée du comportement asymptotique de $T(n)$, on considère $n = 2^k$. On va ignorer l'arrondi dans cette phase «exploratoire». En utilisant les substitions de la récurrence :

$$T\big(2^k\big) = T\big(2^{k/2}\big) + 1 = T\big(2^{k/4}\big) + 1 + 1 = \cdots = T\big(2^1\big) + \underbrace{1 + 1 + \cdots + 1}_{\lg k \text{ fois}}$$

Comme $k = \lg n$, on va chercher une borne de $T(n) \in O(\log \log n)$.

La preuve est comme suit : (A) démontrer la borne pour $n = 2^{2^k}$, (B) démontrer que $T(n)$ est une fonction croissante de n, (C) gérer le cas de n général en utilisant les résultats de B et A.

A. Soit $n = 2^{2^k}$ avec $k \geq 0$. On va démontrer par induction que $T(n) = k + 1$. [On a cette prémonition après la dérivation exploratoire…] Cas de base : $k = 0$, OK par inspection ($T(2) = 1$). Hypothèse d'induction :

$T\big(2^{2^m}\big) = m+1$ pour tout $m = 0, \ldots, k-1$. On a

$$\begin{aligned} T\big(2^{2^k}\big) &= T\big(2^{2^{k-1}}\big) + 1 && \text{(récurrence)} \\ &= \big((k-1)+1\big) + 1 && \text{(hypothèse d'induction)} \\ &= k+1. \end{aligned}$$

Donc $T\big(2^{2^k}\big) = k+1$ pour tout $k \in \mathbb{N}$.

B. On va démontrer par induction que $T(n) \leq T(m)$ pour tout $1 \leq n < m$. [Exercice — preuve par induction.]

C. Maintenant, soit $n > 2$ quelconque, $k = \lceil \lg \lg n \rceil$ et $m = 2^{2^k}$.

$$\begin{aligned} T(n) &\leq T(m) && \text{(par B : } n \leq m\text{)} \\ &\leq k+1. && \text{(par A).} \end{aligned}$$

Donc $T(n) \leq \lceil \lg \lg n \rceil + 1$ pour tout $n = 1, 2, \ldots$. C'est assez bon déjà mais si on veut être tout minutieux, il reste à démontrer qu'il existe une constante $c > 0$ et un seuil N t.q. $\lceil \lg \lg n \rceil + 1 \leq c \lg \lg n$ pour tout $n \geq N$. Par exemple, $\lceil \lg \lg n \rceil + 1 \leq \lg \lg n + 2 \leq 2 \lg \lg n$ quand $\lg \lg n \geq 2$, ou $n \geq 16$. CQFD.

Si on n'utilise pas la monotonicité de $T(n)$ de (B), la preuve peut devenir plus compliquée :

Preuve altérnative pour $\forall n \geq N : T(n) \leq c \lg \lg n$ (avec c, N spécifiés à la fin de la preuve). Supposons [hypothèse d'induction] que

$$T(k) \leq c \cdot \lg \lg k$$

pour tout $N \leq k < n$, y incluant $k = \lceil \sqrt{n} \rceil$. Alors,

$$T(n) = T(\lceil \sqrt{n} \rceil) + 1 \leq 1 + c \cdot \lg \lg \lceil \sqrt{n} \rceil \qquad (*)$$

par l'hypothèse d'induction. Si $n \geq 2$, alors $\lceil \sqrt{n} \rceil \leq 2\sqrt{n}$. En continuant (*)

$$T(n) \leq 1 + c \cdot \lg \lg \big(2\sqrt{n}\big) = 1 + c \cdot \lg \Big(1 + \frac{1}{2} \lg n\Big). \qquad (**)$$

Maintenant on utilise que $\ln(1+x) < \frac{1}{x} + \ln x$ (par expansion Taylor), et donc

$$\lg(1+x) < \lg x + \frac{1}{x \ln 2}.$$

Avec $x = \frac{1}{2} \lg n$, (**) devient

$$T(n) < 1 + c \cdot \Big(\lg \lg n - 1 + \frac{2}{\ln n}\Big). \qquad (***)$$

Or, $\frac{2}{\ln n} - 1 < -0.038$ quand $n \geq 8$, et donc (***) est borné par

$$T(n) < 1 - c \cdot 0.038 + c \cdot \lg \lg n.$$

Avec $c = 27$, $c \cdot 0.038 > 1$ et on a $T(n) < c \cdot \lg \lg n$. Pour $n = 3, \ldots, 8$ on peut vérifier directement que $T(n) \leq 27 \lg \lg n$ est vrai (cas de base pour l'induction). Donc on a que $T(n) \leq 27 \lg \lg n$ pour tout $n \geq 3$ et donc $T(n) \in O(\log \log n)$.

Remarque. Il est important d'arriver à une conclusion de la même forme que l'hypothèse d'induction pour que la preuve soit correcte : en (***) je ne pouvais pas continuer en disant que «alors il existe une constante c' t.q. $T(n) \leq c' \lg \lg n$», parce qu'on doit arriver à une borne $c \lg \lg n$ *avec la même constante* c que dans l'hypothèse d'induction.

1.e

On veut démontrer que $g(n) = n^{O(1)}$ ssi $g(n) = O(n^k)$ pour un k quelconque.

$\Leftarrow$ Si $g(n) = O(n^k)$, alors il existe $c > 0$ et N t.q. $g(n) \leq cn^k$ pour tout $n \geq N$. Donc

$$g(n) \leq cn^k \leq n^{k+1}$$

quand $c \leq n$. Avec $N' = \max\{N, c\}$ et

$$h(n) = \begin{cases} \log_n g(n) & \text{quand } n < N'; \\ k+1 & \text{quand } n \geq N' \end{cases}$$

on a $g(n) = n^{h(n)}$ et $h(n) \in O(1)$. [Le $\log_n g(n)$ n'est pas trop joli, mais on a besoin d'une fonction h t.q. $n^{h(n)} = g(n)$ même quand $n \leq N'$.]

$\Rightarrow$ Si $g(n) = n^{O(1)}$ alors il existe $h(n) \in O(1)$ t.q. $g(n) = n^{h(n)}$. Comme $h(n) \in O(1)$, il existe une constante $c > 0$ et un seuil N t.q. $h(n) \leq c$ pour tout $n \geq N$. Alors,

$$g(n) = n^{h(n)} \leq n^c$$

pour tout $n \geq N$ et donc $g(n) \in O(n^c)$.

2 Minipile

La difficulté ici est la gestion du cas quand le minimum courant x est dépilé. On ne veut pas examiner tous les éléments sur la pile pour trouver le min, mais on doit «se rappeler» plutôt ce qui était cette valeur avant d'empiler x. La solution est d'utiliser deux piles : une pour les vraies valeurs et l'autre pour les minimums. Chaque fois qu'on empile un élément x, on le met sur la «vraie» pile, mais on empile aussi la valeur de $\min\{\mathsf{getMin}(), x\}$ sur une pile auxiliaire. La fonction $\mathsf{getMin}()$ retourne la valeur de l'élément au sommet de la pile auxiliaire (sans le dépiler, bien sûr). Quand on dépile, on doit dépiler de la pile auxiliaire aussi.

Remarque. J'ai vu des implantatations basées sur la même idée sauf qu'on empile le min sur la pile auxiliaire seulement quand le min change. La valeur sur la pile auxiliaire est dépilée lors d'un $\mathsf{pop}()$ si les sommets des deux piles sont égaux. Ceci peut causer des problèmes quand des éléments identiques sont empilés :

$\mathsf{push}(2)$	empiler 2 sur pile auxiliaire
$\mathsf{push}(3)$	rien empilé sur pile auxiliaire
$\mathsf{push}(1)$	empiler 1 sur pile auxiliaire
$\mathsf{push}(1)$	rien empilé sur pile auxiliaire
pop	dépiler 1 de la pile auxiliaire
	$\mathsf{getMin}() = 2$?

Avec une telle solution il est important d'empiler sur la pile auxiliare si l'argument de push est inférieur *ou égal* au minimum courant. Par contre, on utilise moins de mémoire qu'avec la solution esquissée en haut.

3 Deux c'est mieux

L'idée est d'utiliser une pile pour stocker les éléments dans l'ordre LIFO (sur laquelle enqueue = push) et utiliser une autre pile pour enlever dans l'ordre FIFO (sur laquelle dequeue = pop).

Implantation avec deux piles P_{enq} et P_{deq} — toutes les deux sont initialisées comme vide.

```
E1 ENQUEUE (x)
E2 if isEmpty(P_enq) then // (empiler sur P_enq)
E3     while not isEmpty(P_deq) do
E4         push(P_enq, pop(P_deq))
E5 push(P_enq, x)

D1 DEQUEUE
D2 if isEmpty(P_deq) then // (empiler sur P_deq)
D3     while not isEmpty(P_enq) do
D4         push(P_deq, pop(P_enq))
D5 retourner pop(P_deq)
```

[Il faut montrer l'implantation de l'opération isEmpty aussi.]

Remarque. Il existe d'autres solutions : on peut mettre le dépilement-empilement entre les deux piles dans l'une des opérations (par exemple, garder P_{deq} vide hors de l'exécution des opérations) mais la solution ici offre l'avantage que des appels répétés de dequeue ou enqueue prendent seulement $O(1)$.