IFT2015 hiver 2012 — Devoir 1

Miklós Csűrös

18 janvier 2012

À remettre avant 20 :15 le 25 janvier. Remettez un rapport écrit par courriel (à csuros@iro. . .) en format PDF. Travaillez seul.

1.1 Nombre harmonique (15+6 points)

Le **nombre harmonique** $H(n)$ est défini pour tout $n = 1, 2, \ldots$ par

$$H(n) = \sum_{k=1}^{n} \frac{1}{k} = \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n}.$$

a. (5 points) ▶ Donnez une définition récursive de $H(n)$.

b. (10 points) ▶ Démontrez par induction que pour tout $k = 0, 1, 2, \ldots,$

$$H\big(2012^k\big) \geq 1 + \frac{2011}{2012}k.$$

Indice. Utilisez la borne $x_1 + x_2 + \cdots + x_m \geq m \min_{i=1,\ldots,m} x_i$ pour les termes de $H(2012^{k+1}) - H(2012^k)$ dans le cas inductif.

c. (6 points boni) ▶ Démontrez par induction qu'il existe une constante c telle que pour tout nombre Fibonacci $F(n)$ avec $n = 2, 3, 4, \ldots,$

$$H(F(n)) \geq \frac{n-2}{\phi^2} + c,$$

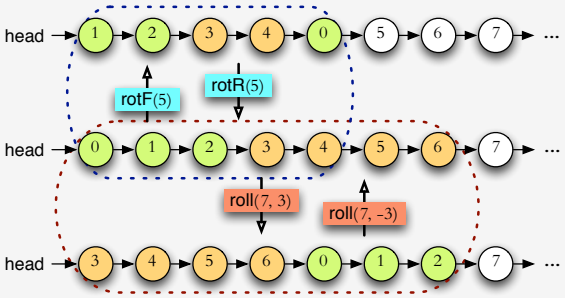
où $\phi = \frac{1+\sqrt{5}}{2}$.

REMARQUE. Cet exercice montre que la croissance de $H(n)$ est au moins logarithmique en n. En vérité, $H(n) = \gamma + \ln n + e(n)$, avec une erreur $\lim_{n\to\infty} e(n) = 0$. La limite $\gamma = \lim_{n\to\infty}\big(H(n) - \ln n\big) = 0.57721\cdots$ s'appelle la constante d'Euler-Mascheroni.

1.2 Décalage circulaire (15 points)

On veut une structure qui supporte des rotations d'éléments sur une liste. Soit $(x_0, x_1, \ldots, x_{\ell-1})$ l'ordre des nœuds sur la liste (x_0 est le premier nœud). Les opérations suivantes changent l'ordre des éléments $x_0, \ldots, x_{n-1}$ avec $n \leq \ell$. En une **rotation avant** (rotF), on avance les nœuds $x_1, \ldots, x_{n-1}$ vers la tête et on place x_0 après x_{n-1}. En une **rotation arrière** (rotR), les nœuds $x_0, \ldots, x_{n-2}$ reculent vers la queue vec le placement de x_{n-1} à la tête. En un **décalage circulaire** (roll) on performe plusieurs rotations avant ou arrière.

Opération	**Résultat**
rotF(n)	$(x_1, x_2, \ldots, x_{n-1}, x_0, \underbrace{x_n, x_{n+1}, \ldots, x_{\ell-1}}_{\text{ne change pas}})$
rotR(n)	$(x_{n-1}, x_0, x_1, \ldots, x_{n-2}, x_0, \underbrace{x_n, x_{n+1}, \ldots, x_{\ell-1}}_{\text{ne change pas}})$
roll(n, j)	$(x_{j \bmod n}, x_{(j \bmod n)+1}, \ldots, x_{n-1}, x_0, x_1, \ldots, x_{(j \bmod n)-1}, \underbrace{x_n, \ldots, x_{\ell-1}}_{\text{ne change pas}})$



L'opération roll(n, j) correspond à j rotations avant (si $j > 0$) rotF ou $|j|$ rotations arrière (si $j < 0$) rotR. On rétablit l'ordre original par l'opération inverse roll$(n, -j) =$ roll$(n, n - j)$.

▶ Montrez comment implanter les opérations rotF(n), rotR(n), et roll(n, j) sur une liste simplement chaînée.

Spécifications. La liste est simplement chaînée ; sa tête est stocké par la variable head ; et chaque nœud N possède la variable N.next. La fin de la liste est dénotée par null. L'algorithme doit déplacer les nœuds et non pas leur contenu. L'algorithme devrait être aussi efficace que possible. (P.e., j ne devrait pas influencer le temps de calcul.) Vous pouvez décrire vos algorithmes en pseudocode ou en Java.