IFT2010 Hiver 2006 — Devoir 1

Miklós Csűrös

1er février 2006

À remettre au cours le 9 février. Pour la partie théorique, remettez un rapport écrit. Un rapport soumis en format de PDF vaut 3 points de boni. Remettez aussi le fichier source `PSPile.java` en utilisant

```
remise ift2010 tp1 PSPile.java
```

1 Oh, la notation

a. (2 points) Est-ce que $O(n^3) \setminus O(n^2) = \Theta(n^3)$? (Où $\setminus$ dénote la soustraction sur des ensembles.) Justifiez votre réponse.

b. (3 points) Est-ce qu'il existe une paire de fonctions $f(n)$ et $g(n)$ telles que $f(n) \notin O(g(n))$ et $g(n) \notin O(f(n))$ en même temps ? Justifiez votre réponse.

c. (5 points) Supposons que $f(n) = O(n \log n)$. Soit $g(n) = f(\lceil \sqrt{n} \rceil)$. Qu'est ce qu'on peut dire de la croissance asympototique de $g(n)$? Donnez votre réponse en notation $O(\cdot)$ avec un preuve détaillée en utilisant la définition de $O(\cdot)$.

d. (10 points) Soit $T(n)$ défini par

$$\begin{aligned} &T(1) = 1; \\ &T(2) = 1; \\ &T(n) = T(\lceil \sqrt{n} \rceil) + 1 \qquad\qquad \text{si } n > 2 \end{aligned}$$

Qu'est ce qu'on peut dire de la croissance asympototique de $T(n)$? Donnez votre réponse en notation $O(\cdot)$ avec une preuve détaillée en utilisant la définition de $O(\cdot)$.

Indice : examinez $T(2^k)$ pour trouver la forme de $f(n)$ en $T(n) \in O(f(n))$, et utilisez de l'induction pour démontrer que l'inégalité de la définition est satisfaite pour tout n (à partir d'un seuil quelconque).

e. (10 points) Démontrez formellement que $g(n) \in n^{O(1)}$ si et seulement si il existe une constante $k > 0$ telle que $g(n) \in O(n^k)$. Ici,

$$n^{O(1)} = \bigcup_{h(n) \in O(1)} n^{h(n)}.$$

2 Minipile (5 points)

Considérez une pile avec les opérations usuelle pop et push. Montrez comment on peut aussi implanter l'opération getMin qui retourne l'élément de valeur minimale sur la pile (supposons que c'est une pile sur des nombres). Votre solution doit supporter toutes les trois opérations en temps $O(1)$.

3 Deux c'est mieux (15 points)

Expliquez comment implanter une queue en utilisant deux piles. Montrez la [pseudo]code pour l'implantation des opérations enqueue, dequeue, isEmpty. Analysez le temps de calcul de chacune des opérations.

4 PostScript

ADT (25 points)

Le langage PostScript est basé sur une pile. Pour exécuter une instruction, il faut d'abord empiler les arguments et puis appeler l'instruction qui les dépile. Les valeurs (s'il y en a) sont retournées sur la pile aussi. Par exemple, `3 5 add` empile les nombers 3 et 5, puis `add` les dépile et empile le résultat 8. Un autre exemple : `100 100 10.2 10.2 moveto lineto` empile les quatre nombres, puis

exécute `moveto` qui est une procédure à deux arguments pour bouger le *stylo* en position (10.2, 10.2), et finalement exécute `lineto` qui est une procédure à deux arguments pour dessiner une ligne jusqu'à (100,100). La pile est vide après l'exécution de ces opérations.

Dans cet exercice on analysera un type abstrait qui correspond à la fonctionnalité da la pile en PostScript. Les opérations sont les suivantes.

any	**pop**	–	Discard top element
$any_1\ any_2$	**exch**	$any_2\ any_1$	Exchange top two elements
any	**dup**	*any any*	Duplicate top element
$any_1 \ldots any_n\ n$	**copy**	$any_1 \ldots any_n\ any_1 \ldots any_n$	Duplicate top n elements
$any_n \ldots any_0\ n$	**index**	$any_n \ldots any_0\ any_n$	Duplicate arbitrary element
$any_{n-1} \ldots any_0\ n\ j$	**roll**	$any_{(j-1) \bmod n} \ldots any_0\ any_{n-1} \ldots any_{j \bmod n}$	Roll n elements up j times
$\vdash any_1 \ldots any_n$	**clear**	$\vdash$	Discard all elements
$\vdash any_1 \ldots any_n$	**count**	$\vdash any_1 \ldots any_n\ n$	Count elements on stack

Le syntaxe des opérations est le suivant : à la gauche se trouve la pile avant l'exécution de l'opération (bas de la pile vers la gauche), le côté droit montre comment la pile change (seulement pour la partie indiquée à la gauche). $\vdash$ dénote le bas de la pile. Exemples :

$$\vdash\ 1\ 2\ 3\ 4\ \ \mathsf{pop} \Rightarrow \vdash\ 1\ 2\ 3$$
$$\vdash\ 1\ 2\ 3\ 4\ \ \mathsf{exch} \Rightarrow \vdash\ 1\ 2\ 4\ 3$$
$$\cdots\ 1\ 2\ 3\ 4\ 3\ 1\ \ \mathsf{roll} \Rightarrow \cdots\ 1\ 4\ 2\ 3$$
$$\cdots\ 1\ 2\ 3\ 4\ 3\ {-1}\ \ \mathsf{roll} \Rightarrow \cdots\ 1\ 3\ 4\ 2$$
$$\vdash\ 1\ 2\ 3\ 4\ 1\ \ \mathsf{index} \Rightarrow \vdash\ 1\ 2\ 3\ 4\ 3$$

Notre ADT (PSPile) ne contient pas l'opération copy mais tous les autres : pop, exch, dup, index, roll, clear, count. Expliquez comment implanter cet ADT en utilisant une liste doublement chaînée. Analysez brièvement (sans preuve) le temps du calcul des opérations dans votre solution. Question de 3 points : quel est le résultat de

1 2 3 4 1 2 3 4 1 2 3 4 dup 3 index exch pop -2 roll pop pop index index roll

Implantation (25 points)

Implantez les méthodes suivantes dans une classe appellée `PSPile` :

```
public void pop();
public void exch();
public void roll();
```

```
public void count();
public void clear();
public void dup();
public void index();
```

Les opérations index et roll peuvent assumer que les arguments en haut de la pile sont de type `java.lang.Integer`. Vous devez aussi implanter les méthodes suivantes :

```
public void push(Object O);
public Object peek();
```

La méthode `push` implante l'opération usuelle de push sur les piles. La méthode `peek` retourne l'élément supérieur dans la pile (sans dépiler).

La liste chaînée est implantée dans la classe `ca.umontreal.iro.ift2010.h06.adt.ListeChaine`. Dans votre implantation, n'utilisez que des classes simples (package `java.lang`).