IFT2015 H10 — Examen Intra

Miklós Csűrös

15 février 2010

Aucune documentation n'est permise. L'examen vaut 100 points.
Répondez à toutes les questions dans les cahiers d'examen.

0 Votre nom (1 point)

Écrivez votre nom et code permanent sur tous les cahiers soumis.

1 Échange (9 points)

Montrez le code pour échanger deux éléments sur une liste chaînée. L'argument de l'opération est une référence au nœud après lequel on doit échange deux nœuds successifs.

2 Tours de Hanoï (20 points)

Dans le jeu de Tours de Hanoï, il faut déplacer des disques à diamètres différents $(1, 2, \ldots, n)$ d'une tour de départ à une tour d'arrivée en passant par une tour intermédiaire, tout en respectant les règles suivantes. (Les disques sont en ordre décroissant au début.)

Règle 1. On ne peut déplacer plus d'un disque à la fois. Un déplacement consiste de mettre le disque supérieur sur une tour au-dessus des autres disques (s'il y en a) sur une autre tour.

Règle 2. On ne peut placer un disque que sur un autre disque plus grand que lui ou sur un emplacement vide.

Pour la solution, on utilise la procédure récursive HANOI(i, j, k, n) qui déplace les n disques supérieurs sur tour i vers tour j en utilisant la tour intermédiare k.

```
Algo HANOI(i, j, k, n)
H1  si n ≠ 0
H2      HANOI(i, k, j, n − 1)
H3      déplacer disque n de i à j
H4      HANOI(k, j, i, n − 1)
```

Soit $T(n)$ le temps de calcul de cet algorithme récursive. Montrez les récurrences pour le temps de calcul. Démontrez que $T(n) = O(2^n)$ en utilisant la définition de $O(\cdot)$.

3 Taux de croissance (20 points)

Remplissez le tableau suivant : chaque réponse vaut 2 point. Pour chaque paire f, g, écrivez "=" si $\Theta(f) = \Theta(g)$, "$\ll$" si $f = o(g)$, "$\gg$" si $g = o(f)$, et "???" si aucun des trois cas n'applique. Il n'est pas nécessaire de justifier vos réponses.

	$f(n)$	$g(n)$
a	$f(n) = \sqrt{n}$	$g(n) = n^{1/3}$
b	$f(n) = 2n + \lg n$	$g(n) = n - 1$
c	$f(n) = 4^{\lg n}$	$g(n) = \sqrt{n}$
d	$f(n) = n!$	$g(n) = n^n$
e	$f(n) = 2^n$	$g(n) = 3^n$
f	$f(n) = \lg n$	$g(n) = \log_3 n$
g	$f(n) = n^2$	$g(n) = 2^{2\log_3 n}$
h	$f(n) = \begin{cases} 1 & \{n = 0\} \\ 2 \cdot f(\lfloor n/2 \rfloor) + 1 & \{n > 0\} \end{cases}$	$g(n) = n$
i	$f(n) = \begin{cases} 1 & \{n = 0, 1\} \\ f(n-1) + O(n) & \{n > 1\} \end{cases}$	$g(n) = \frac{n^2}{\lg(n+1)}$
j	$f(n) = \begin{cases} 1 & \{n \bmod 2 = 1\} \\ \lg^* n & \{n \bmod 2 = 0\} \end{cases}$	$g(n) = \begin{cases} \lg^* n & \{n \bmod 2 = 1\} \\ 1 & \{n \bmod 2 = 0\} \end{cases}$

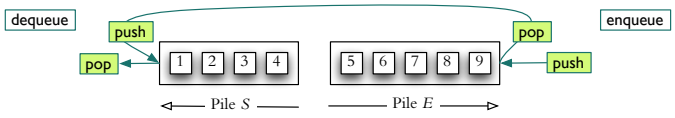
4 Exposition (20 points)

Soit T un arbre binaire. Dans cet exercice, on définit l'**exposition** d'un nœud u comme la distance *la plus courte* à une feuille dans le sous-arbre de u. L'exposition d'une feuille est 0.

Montrez un algorithme qui calcule l'exposition de chaque nœud dans l'arbre.

5 Deux c'est mieux (30+10 points)

On peut implanter une queue (file FIFO) en utilisant deux piles E et S («entrée» et «sortie»).




```
Opération enqueue(x)
1  E.push(x)

Opération dequeue()
1  si S est vide alors
2       tandis que E n'est pas vide faire S.push(E.pop())
3  retourner S.pop()
```

a. (15 points) Ajoutez l'opération delete(k) qui défile k fois ou arrête plus tôt quand la queue est vide. L'implantation ne devrait jamais mener à un débordement des piles sous-jacentes. L'opération ne retourne pas de valeur.

b. (15 points) Quel est le temps de calcul des opérations enqueue et dequeue(k) dans le pire cas ? Démontrez que le temps de calcul pour une séquence de n opérations quelconques est toujours $O(n)$ — autrement dit, que le coût amorti des opérations est $O(1)$. Les opérations pop et push, ainsi que le test «si vide» s'exécutent en $O(1)$.

b. (10 points de boni) Montrez une implantation détaillée avec les deux piles stockés *ensemble en un seul tableau*.

BONNE CHANCE !

English translation

No documentation is allowed. The examen is worth 100 points.
Answer each question in the exam booklet.

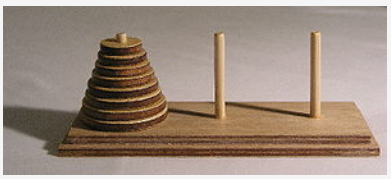
0 Your name (1 point)

Write your name and *code permanent* on each booklet that you submit.

1 Exchange (9 points)

Show the code for exchanging two elements on a linked list. The operation's argument is a reference to the node after which two successive elements are to be switched.

2 Towers of Hanoi (20 points)



The Towers of Hanoi puzzle consists of three pegs, and a set of disks of different diameters $(1, 2, \ldots, n)$. The objective is to move the stack of disks (initially in decreasing order) from one peg to another, obeying the following rules.

Rule 1. Only one disk may be moved at a time. A move consist of taking the upper disk from one peg and placing it on top of other disks (if any) on another peg.

Règle 2. A disk may be placed only on top of a smaller disk, or on an empty peg.

The solution uses the recursive procedure HANOI(i, j, k, n) that moves the top n disks from peg i to peg j using k as the auxiliary peg.

```
    Algo HANOI(i, j, k, n)
H1  if n ≠ 0
H2      HANOI(i, k, j, n − 1)
H3      move disk n from i to j
H4      HANOI(k, j, i, n − 1)
```

Let $T(n)$ be the running time of this algorithm. Show the recurrences for $T(n)$. Prove that $T(n) = O(2^n)$ using the definition of $O(\cdot)$.

3 Growth rates (20 points)

Fill out the following table : every answer is worth 2 points. For each pair f, g, write "=" if $\Theta(f) = \Theta(g)$, "$\ll$" if $f = o(g)$, "$\gg$" if $g = o(f)$, and "???" if neither of the three applies. You do not need to justify the answers.

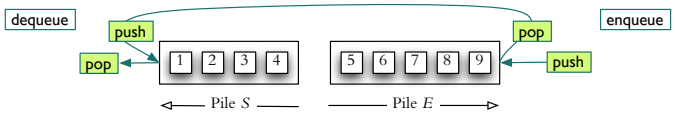
	$f(n)$	$g(n)$
a	$f(n) = \sqrt{n}$	$g(n) = n^{1/3}$
b	$f(n) = 2n + \lg n$	$g(n) = n - 1$
c	$f(n) = 4^{\lg n}$	$g(n) = \sqrt{n}$
d	$f(n) = n!$	$g(n) = n^n$
e	$f(n) = 2^n$	$g(n) = 3^n$
f	$f(n) = \lg n$	$g(n) = \log_3 n$
g	$f(n) = n^2$	$g(n) = 2^{2\log_3 n}$
h	$f(n) = \begin{cases} 1 & \{n = 0\} \\ 2 \cdot f\big(\lfloor n/2 \rfloor\big) + 1 & \{n > 0\} \end{cases}$	$g(n) = n$
i	$f(n) = \begin{cases} 1 & \{n = 0, 1\} \\ f(n-1) + O(n) & \{n > 1\} \end{cases}$	$g(n) = \frac{n^2}{\lg(n+1)}$
j	$f(n) = \begin{cases} 1 & \{n \bmod 2 = 1\} \\ \lg^* n & \{n \bmod 2 = 0\} \end{cases}$	$g(n) = \begin{cases} \lg^* n & \{n \bmod 2 = 1\} \\ 1 & \{n \bmod 2 = 0\} \end{cases}$

4 Exposure (20 points)

Let T be a binary tree. In this exercise, define the **exposure** of a node u as the *shortest* distance to a leaf within u's subtree. A leaf's exposure is 0. Design an algorithm that computes each node's exposure.

5 Two is better than one (30+10 points)

One can implement a queue by using two stacks E and S.




```
Operation enqueue(x)
1  E.push(x)

Operation dequeue()
1  if S is empty then
2       while E is not empty do S.push(E.pop())
3  return S.pop()
```

a. (15 points) Add the operation delete(k) that dequeues k times or stops earlier when the queue becomes empty. The implementation should never result in stack underflow. The operation does not return any value.

b. (15 points) What is the worst-case running time of the operations enqueue and dequeue(k) ? Show that any sequence of n opérations can be executed in $O(n)$ time ; in other words, that the amortized cost is $O(1)$. The stack operations pop and push, as well as the "is empty" test take $O(1)$ time.

c. (10 bonus points) Show a detailed implementation by storing the two stacks *together in a single array*.

GOOD LUCK !