Universidad de Chile
Departamento de Ciencias de la Computación

CC61X **Design and Analysis of Adaptive Algorithms** **2008 March-June**

(Due: Wednesday August 13 by email)

Problem 1 (3 marks)
The `Hanoï Tower Problem` is a classic example on recursivity, originally proposed by Édouard Lucas. A recursive algorithm is known since 1892, moving the n disks of a Hanoï Tower in $2^n - 1$ unit moves, this value being proven optimal by a simple lower bound (see Figure 1).

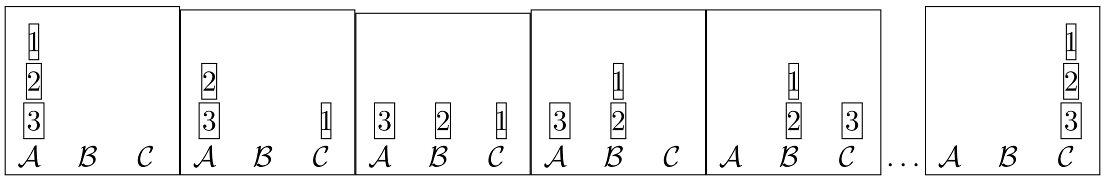

Figure 1: Moving a Hanoï Tower of size 3.

a. Give the recursive algorithm to move a Hanoï Tower of height n from one peg to the other, using only one extra peg.

b. Prove that its worst case performance is $2^n - 1$ unit moves.

c. Prove that this performance is optimal.

   Consider a very simple variant to introduce more diversity in the instances, where we allow disks of same size: we call it the `Disk Pile Problem`. This obviously introduces much easier instances, where the tower can be moved in linear time (see Figure 2).

d. Give a recursive algorithm to move a `Disk Pile` from one peg to the other, using only one extra peg, knowing that $\forall i \in \{1, \ldots, s\}$, n_i is the number of disks of size i. Your algorithm must be efficient for the cases where all the disks are the same size, and where all the disks are of distinct sizes.

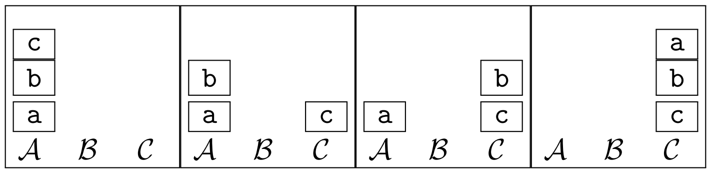

Figure 2: Moving a disk pile of size 3.

e. Give and prove the worst case performance of your algorithm over all instances of fixed s and vector $(n_1, \dots, n_s)$.

f. Prove that a performance of $\sum_{i\in\{1,\dots,s\}} n_i 2^{s-i}$ is optimal.

g. What is the worst case performance of your algorithm over all instances of fixed value s and fixed total number of disks n?

Problem 2 (3 marks)
Consider the following variant: assume that we have three pegs, n disks (of distinct sizes). A legal configuration of the n disks on the three pegs must satisfy the constraint that each disk must rest on a larger disk or an empty peg. Initially, the n disks are on the first peg. The goal is to move them to the third peg. *The only move permitted is to take a disk in middle position of a tower, and to insert it in the middle of a tower* (see Figure 3). This tower is called `Bouncing`, because it can be viewed as if the tower were reposing on a spring, tuned so that the middle of the tower is at constant height.

The "middle" of the tower is formally defined as follow: On a peg containing n disks ranked by increasing sizes, we shall be allowed only to remove the disk of rank $\lfloor \frac{n}{2} \rfloor + 1$ called the *middle* disk, and we can insert a disk only in position $\lfloor \frac{n+1}{2} \rfloor$ in the tower. This means that if n is odd, we remove the center disk, and insert below it; and if n is even, we insert in the middle of the tower, and remove the disk below the middle of the tower. (see Figure 3)

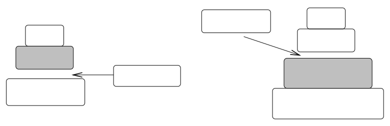
Figure 3: Rules for insertions and removals from a `Bouncing Tower`. If the next operation is a removal, then the shaded disk will be the disk removed. If the next operation is an insertion, then it takes place at the position indicated by the arrow.

We say that an insertion of disk d on peg A is *legal* if d can be inserted in the insertion point of A and leads to a legal configuration. A move is said legal if there is a disk d to remove from the initial peg, and that the insertion of d on the final peg is legal.

a. Give an algorithm to move a tower of n disks of distinct sizes from one peg to another.

b. Analyze the complexity $f(n)$ of your algorithm.

c. What other questions can be asked about this problem?