The Art of Recursion: Problem Set 1
Due Tuesday, 11 September 2012

Consider the following recursive definition of the factorial function on natural numbers:

\[
\begin{align*}
\text{fact}(0) &= 1 \\
\text{fact}(n) &= n \cdot \text{fact}(n - 1) \quad (n > 0)
\end{align*}
\]

It is common to use the notation \( n! \) as an abbreviation for \( \text{fact}(n) \).

1. Implement the above definition directly as a recursive function, using any programming language you like.\(^1\)

2. What is the big-\( O \) time complexity of computing \( \text{fact}(n) \) using this definition?\(^2\) Empirically, how large can you make \( n \) and still reasonably compute \( \text{fact}(n) \) with your implementation?

3. Prove by induction that given any natural number \( n \) as input, evaluation of \( \text{fact}(n) \) will finish after a finite amount of time (that is, \( \text{fact}(n) \) terminates for all inputs).

4. Write down a recursively defined function on the natural numbers which does not terminate for some (or all) inputs. What goes wrong if you try to prove that it always terminates using induction?

5. Prove that \( n! \geq 2^n \) for all \( n \geq 4 \).

6. What goes wrong if you try using induction to prove \( n! \geq 2^n \) for all \( n \geq 0 \)?

7. What goes wrong if you try using induction to prove \( n! \geq 2^n \) for all \( n \geq 2 \)?

8. Prove by induction: for all \( n \geq 0 \),

\[
\sum_{k=0}^{n} k \cdot (k!) = 0 \cdot 0! + 1 \cdot 1! + 2 \cdot 2! + \cdots + n \cdot n! = (n + 1)! - 1.
\]

9. Consider the following infinite sequence of positive integers, with the first ten terms shown:

\[0, 1, 3, 6, 10, 15, 21, 28, 36, 45, \ldots\]

- Write down a recursive definition of this sequence.
- Derive a (non-recursive) formula for computing the \( n \)th term of the sequence, and prove that it gives the same results as your recursive definition from part (a).

10. Write down a recursive definition for the following sequence:

\[0, 1, 3, 13, 183, 33673, 1133904603, 1285739649838492213 \ldots\]
The *Towers of Hanoi* is a classic puzzle with a solution that can be described recursively. Disks of different sizes are stacked on three pegs; the goal is to get from a starting configuration with all disks stacked on the first peg to an ending configuration with all disks stacked on the last peg, as shown in Figure 1.

The only rules are

- you may only move one disk at a time, and
- a larger disk may never be stacked on top of a smaller one.

For example, as the first move all you can do is move the topmost, smallest disk onto a different peg, since only one disk may be moved at a time.

From this point, it is *illegal* to move to the configuration shown in Figure 3, because you are not allowed to put the green disk on top of the smaller blue one.

11. Describe a recursive procedure for solving the puzzle.

12. How many steps does your procedure take to solve the puzzle with $n$ disks? Prove your answer by induction.

13. Figure 4 shows several circles, cut by chords into one, two, four, and six regions, respectively.

The pictures with zero, one, and two chords show the maximum possible number of regions (one, two, and four, respectively) which can be created with that many chords. However, using three chords it is possible to create more regions than shown.

In general, what is the maximum number of regions that can be created using $n$ chords? Prove your answer.