The Art of Recursion: Problem Set 2
Due Tuesday, 18 September 2012

Strong Induction and Memoization

Consider the following recursive definition of the Fibonacci numbers:

\[ F_0 = 0 \]
\[ F_1 = 1 \]
\[ F_n = F_{n-1} + F_{n-2} \quad \text{when } n \geq 2. \]

1. Implement a recursive function \( \text{fib}(n) \) which takes as input a natural number and outputs the Fibonacci number \( F_n \), corresponding directly to the above recursive definition. Be sure to stay as faithful to the above definition as possible in your chosen programming language.\(^1\) The idea is to end up with an implementation which obviously corresponds to the above definition, regardless of efficiency.

Empirically, for what inputs are you actually able to run your implementation? Explain your results.

2. Prove that your implementation of the Fibonacci function terminates for all natural number inputs.

3. [optional]\(^2\) What is the big-O time complexity of your implementation (assuming addition takes constant time)?

4. Implement a function which takes as input a natural number \( n \), and constructs an array of all the Fibonacci numbers from \( F_0 \) up to \( F_n \). What is the big-O time complexity of your implementation? Use it to compute \( F_{100} \).

5. Prove that your implementation is correct (that is, that it really does compute an array containing \( F_i \) at index \( i \)).

6. To implement an efficient Fibonacci function, is it necessary to compute and store the entire array of Fibonacci numbers from \( F_0 \) up to \( F_n \)? Why or why not? If not, implement a third variant of a Fibonacci function which is still efficient yet requires less memory than the second implementation.

7. Prove the principle of strong induction using only the usual ("weak") principle of induction. This shows that in fact, the principle of strong induction is not actually any "stronger" than the usual principle of induction with regard to what it allows us to prove, although it may be far more convenient in some cases.

\(^1\) Again, if there is a “big integer” type available, you should use that.

\(^2\) This problem is marked [optional] because I don’t expect you to be able to derive the answer yourself; the idea is that if you wish, you can look up the answer (or try to derive it yourself, if you are the adventurous type) and be prepared to explain the derivation to the class. In general, problems marked [optional] will not be randomly assigned for presenting; instead, you may volunteer to present a solution to an [optional] problem if you wish.
8. Suppose we wish to count how many ways there are to tile a $1 \times n$ rectangle using $1 \times 1$, $1 \times 2$, $1 \times 3$, and $1 \times 4$ blocks. (The order of the tiles matters.) For example, there are 29 ways to tile a $1 \times 6$ rectangle, as shown in Figure 1.

Let $t(n)$ denote the number of ways to tile a $1 \times n$ rectangle using blocks of length 1, 2, 3, and 4. For example, $t(6) = 29$.

(a) Write down a recurrence for $t(n)$.

(b) Find $t(100)$.

Structural recursion and induction

Consider the algebra $\Omega = (\Sigma, A)$ which has operations $\Sigma = \{Z, S\}$ and arities $A(Z) = 0$ and $A(S) = 1$.

9. Write down a few small terms of this algebra. Four is enough.

10. Describe the set $\llbracket \Omega \rrbracket$ of terms.

11. Write down the induction principle for this algebra. Have you seen this principle anywhere before?

Now consider the algebra $\Phi = (\Sigma_\Phi, A_\Phi)$, defined by

$$\Sigma_\Phi = \{R, S\}$$

$$A(R) = 0$$

$$A(S) = 2$$

12. Write down four or five small terms of $\llbracket \Phi \rrbracket$.

13. Describe the set $\llbracket \Phi \rrbracket$ of terms.

14. Write down the induction principle for $\Phi$.

15. Consider the following functions, defined recursively over the set of terms $\llbracket \Phi \rrbracket$. $\text{foo}$ is a function from $\llbracket \Phi \rrbracket$ to $\mathbb{N}$, whereas $\text{bar}$ is a function from $\llbracket \Phi \rrbracket$ to the set of Boolean values $\{\text{True, False}\}$.

$$\text{foo}(R) = 0$$

$$\text{foo}(S(x_1, x_2)) = 1 + \text{foo}(x_1) + \text{foo}(x_2)$$

$$\text{bar}(R) = \text{True}$$

$$\text{bar}(S(x_1, x_2)) = \text{bar}(x_1) \land \text{bar}(x_2) \land (\text{foo}(x_1) = \text{foo}(x_2))$$

State and prove an interesting theorem relating $\text{foo}$ and $\text{bar}$.  

$^3$What counts as “interesting” is left up to your judgement.