Lab 1: Recursion Versus Iteration

Week of September 17, 2018

Objective

To compare the efficiency of a recursive implementation of an algorithm and an iterative version.

The Rolen Numbers

The Rolen numbers are defined as follows: $R(0) = 1$, $R(1) = 1$,

$$R(n) = R(n-1) + \sum_{k=0}^{n-2} (R(k) \times R(n-2-k)), \quad \text{for } n > 1.$$

For example, here is how you would compute $R(2)$:

$$\begin{aligned}
R(2) &= R(2-1) + \sum_{k=0}^{2-2} (R(k) \times R(2-2-k)) \\
&= R(1) + \sum_{k=0}^{0} (R(k) \times R(-k)) \\
&= R(1) + (R(0) \times R(0)) \quad \text{(the sum only has a } k = 0 \text{ term)} \\
&= 1 + (1 \times 1) \quad \text{(because } R(0) = 1 \text{ and } R(1) = 1\text{)} \\
&= 1 + 1 \\
&= 2
\end{aligned}$$

And here is how you would compute $R(3)$:

$$\begin{aligned}
R(3) &= R(3-1) + \sum_{k=0}^{3-2} (R(k) \times R(3-2-k)) \\
&= R(2) + \sum_{k=0}^{1} (R(k) \times R(1-k)) \\
&= R(2) + (R(0) \times R(1-0)) + (R(1) \times R(1-1)) \quad \text{(the sum has } k = 0 \text{ and } k = 1 \text{ terms)} \\
&= 2 + (1 \times 1) + (1 \times 1) \quad \text{(because } R(0) = R(1) = 1 \text{ and } R(2) = 2\text{)} \\
&= 2 + 1 + 1 \\
&= 4
\end{aligned}$$

Here are some values in the series:

n	0	1	2	3	4	5	6	7	8	9
R(n)	1	1	2	4	9	21	51	127	323	835

Exercises

Exercise 1: Drawing a calling tree: Draw the calling tree, showing all the recursive calls that would be made if a recursive method is computing $R(4)$. Note that the recursive part of a recursive method needs to make a number of recursive calls — every time it needs the value of $R(\text{anything})$ when it is computing $R(n)$, it makes a recursive call to get the value of $R(\text{anything})$. (Note that "anything" is any value less than n.)

Exercise 2: Complete a program: The file `Rolen.java` contains a nearly-complete program, except it needs TWO methods added (details below).

The two `static` methods you will write:

`recursiveRolen`: This method is passed an `int n` and recursively computes and returns the n^{th} Rolen number, $R(n)$, for $n \geq 0$.

`iterativeRolen`: This method is passed an `int n` and iteratively computes the n^{th} Rolen number, $R(n)$, for $n \geq 0$. When $n > 1$, this iterative method uses a loop (instead of recursion) and an array (to store Rolen numbers as you compute them) to compute the n^{th} Rolen number, for $n \geq 0$. The idea if $n > 1$: First, put $R(0)$ and $R(1)$ into the first two positions of the array. Then use a loop to compute $R(i)$ and put it into position i of the array, for i going from 2 to n. Return $R(n)$ after the loop is done.

Note that you must use type `long`, not `int`, to store Rolen numbers, because Rolen numbers get very big very fast!