Modeling and Programming 2018-1: Seventh lab practice

Luis Daniel Aragon Bermudez 416041271

November 7th, 2017

Contents

The Stern-Brocot representation of rational numbers

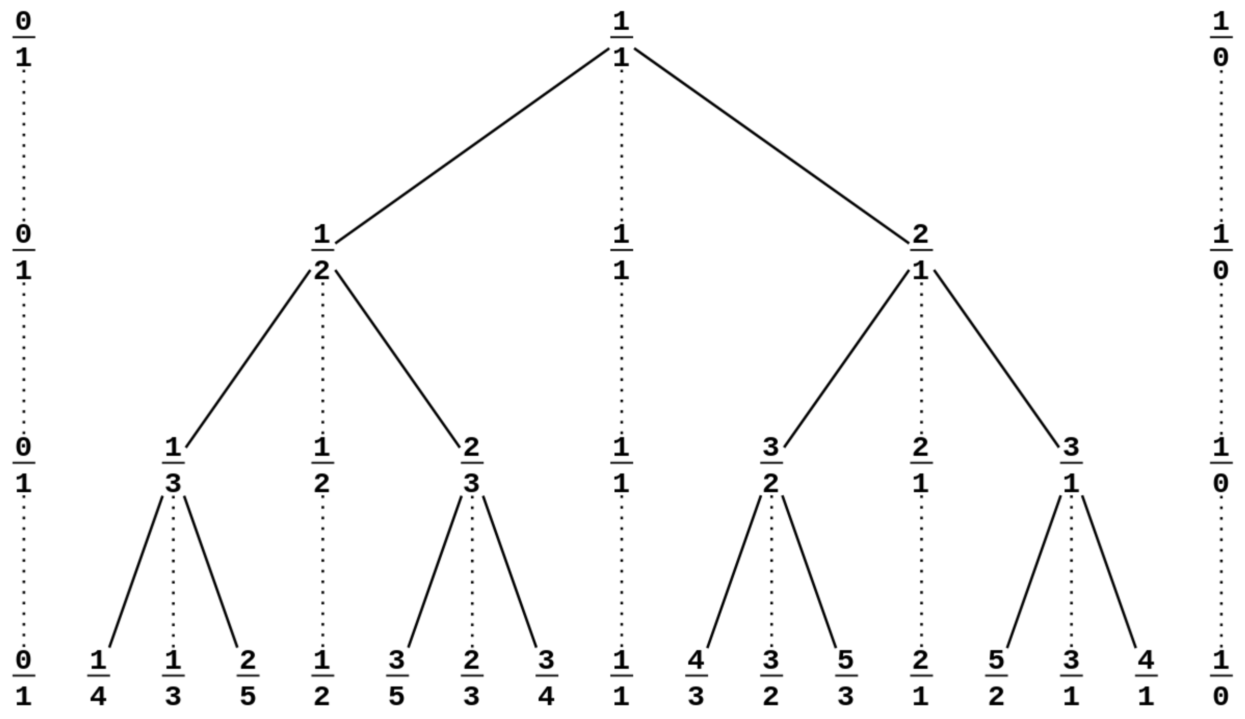

Figure 1: Shows the first few iterations of a Stern-Brocot tree

The Stern-Brocot (SB) Tree can be seen as a method for constructing all positive fractions $\frac{m}{n}$ (where m and n are relative primes). The idea is to begin with two imaginary ratios $\frac{0}{1}$ and $\frac{1}{0}$, then perform the following operation: **Insert $\frac{m+m'}{n+n'}$ between two adjacent fractions $\frac{m}{n}$ and $\frac{m'}{n'}$** as many times as needed. This constructs an infinite binary tree that preserves order, hence no fraction can appear more than once.

We can define the SB representation of all positive rational numbers as the path taken to reach that number in the SB Tree, ie. a string of 'R's and 'L's corresponding to a right or left descent starting from the root ($\frac{1}{1}$). We won't prove this but the right and left descents in the SB Tree can be represented as matrices (that, intrestingly enough, form a group!) which are also defined in the provided library. Hence each LR string corresponds to a product of matrices, from which a fraction can be easily extracted. If nodes are represented as matrices, then the root of the SB Tree is the identity matrix of dimension 2, fractions are encoded into these integral matrices and can be extracted by using the sum of the first row as the numerator and the sum of the second row as the denominator. The provided library and its corresponding app exploit these properties.

The Stern-Brocot Representation Library implements conversion functions to and from the Fraction datatype and the included demo app provides a command-line utility that takes two arguments: `challenge02 <input file> <output file>`. The input file should contain a pair of numbers that represent a fraction in every line, the first number should be the numerator and, the second number, the denominator. The output file will then contain the corresponding Stern-Brocot representation of each fraction.

Installing and running the demo app

Installation is made very simple by using `stack` and `git`. If your system is correctly configured then, installing the demo should be possible using the following one liners:

1. To install globally:

```
> git clone https://github.com/tormenteddan/MP-2018-1-p07.git
...
> cd MP-2018-1-p07 ; stack install
```

2. To install locally use

```
> git clone https://github.com/tormenteddan/MP-2018-1-p07.git
...
> cd MP-2018-1-p07 ; stack build
```

Running the demos is also really simple:

1. If you installed globally, use `challenge02 <input file> <output file>` to run the app.
2. If you installed locally, use `stack exec challenge02 <input file> <output file>` to run the app.

Acknowledgements

For more information on the tools used to build, create and run this program refer to the following links:

- Haskell is probably one of the most popular functional programming languages.
- Stack must the be the most popular build and project manager for haskell modules and programs.
- Numeric Quest is a collection of Haskell modules useful for Mathematics in general, and Quantum Mechanics in particular. The included Fraction datatype was fundamental for developing this app and library.
- Data.Matrix is a haskell library that implements matrices and common matrix operations.