Lab: First Computational Exploration of Boomerang Fractions

Introduction

You've had a chance to investigate this problem with paper and pencil, and perhaps you've gained some insights into which sorts of fractions behave better than others. Is there a pattern to this better behavior that is shared among these fractions?

For most problems that we might tackle computationally, there are multiple approaches we can follow; this is certainly the case for boomerang fractions. What are some of the strengths of computers?

- Speed in performing calculations. Keep in mind, however, that for some problems, sheer speed or processing power isn't enough to answer all the questions we might have.
- Graphing and other sorts of visualization (graphics that help you see patterns and relationships that are hard to see in lists of number or other symbols).

Getting started

Let's take a step back to math circle activity and look at the 2 operations used in boomerang fraction sequences: **addition** and **inversion**.

1. Can you think of a simple way to choose randomly between two choices? For example, these 2 operations of **addition** and **inversion** at each step in a boomerang fraction sequence?
2. Use the method you came up with to generate the first 10 terms of a sequence for the fraction 4/5, and write it down.

   $$1 \to 9/5 \to ? \to ? \to ? \to ? \to ? \to ? \to ? \to ?$$

3. Use the same process again to generate the first 10 terms of another sequence for 4/5. Did you get the same sequence?
4. How many different sequences of 10 terms (or fewer, if a sequence returns to 1 before the 10^{th} term) are possible, if we randomly choose between addition and inversion after the first 2 terms? Does this number depend on the fraction we're using?

This looks like it will be quite time consuming to do for one fraction, much less ten or twenty.

Nick Bennett & James Taylor
Math Circles Collaborative of New Mexico 28 May 2017

Digging deeper

1. How can we do this on a computer? Once you've learned a bit about how a spreadsheet like Excel works (any spreadsheet will do), especially how to use formulas, functions and replicated cells, you are ready to explore this on a tool available to pretty much anyone with a computer. You need to know just a few functions to allow you to automate the sequences you got in the previous section. You'll need the **`RAND()`** and **`IF()`** functions and eventually want to understand the importance of *absolute addressing*.
2. Open a new spreadsheet and begin. Pick your fraction. Start with 4/5 and make a column of 10 sequence elements.
3. Now make 15 more columns just like it. What do you notice?
4. Save your document with a good filename!
5. See if you can find the "recalculate" button or function for your spreadsheet and press it. What happens?
6. See if you can you generalize your computer-aided process, so that it can be used for fractions other than 4/5, without a lot of extra typing.
7. Can you extend the maximum length of these generated sequences to 20 terms? 50 terms? 100 terms? How would extending the length in this fashion affect the number of different sequences that could be generated?
8. Is it possible to generate a large number of random sequences in this fashion, while keeping track of the shortest generated sequence (if any) that returns to 1? How can you add a column that helps you track where 1 first appears?
9. How can you make sure that you have found the shortest sequence back to 1?

Questions for discussion

1. Just as we know that the first operation in a boomerang fraction sequence is always addition, what do we know (if anything) about the *last* operation in a sequence that returns to 1?
2. Besides the first and last operations, are there other conditions under which we might be able to exclude either addition or inversion from consideration for the next operation to be performed at some points in the sequence? If so, how might we modify our approach to address this?
3. Is it possible for the approach developed so far (i.e. one that generates sequences via random selection of operations) to miss a solution – that is, not to generate the shortest sequence returning to 1 – even after generating a large number of sequences? If so, is there a way to generate *all* possible sequences of a specified length? (In general, this process is called *enumeration* – that is, numbering and listing all possible alternatives. In this case, thinking of the either-or choice between addition and inversion as part of a numbering system can be helpful.). Discuss with your entire group!