Unit Fractions

March 19, 2017

The equation

$$\frac{5}{8} = \frac{1}{2} + \frac{1}{8}$$

is interesting. It tells us that if you want to share 5 pizzas among 8 people you should cut 4 pizzas in half and the remaining pizza into eight pieces and then given everybody one piece of each size. That make much more sense than trying to give everyone one piece that is 5/8 of a pizza.

We can't always make such cute interpretations of equations like this, but they're interesting for a variety of other reasons as well. We're interested here in writing a fraction — we might as well assume it is less than one — as a sum of fractions whose numerators are all 1. We call those *unit fractions.*

To keep things interesting, since I'm sure you were going to ask, why don't we only use positive numbers and not allow subtraction. And we might as well start off with fractions that are written in lowest terms so their top and bottom have no factors other than 1 in common.

Try a few to get a feel for how things might work.

$2/13 = 1/13 + 1/13$ which isn't all that interesting. How about we agree you can't repeat unit fractions in the decomposition.

Do some more. 2/13 again. Did everybody get the same answer? How did you get your answer?

Folks proceeded in different ways using a number of good ideas. Greedy algorithm, picking denominators with lots of factors, picking denominators that are even or powers of 2. Or have easy to guess decimal representations.

Which solutions are special, interesting, or nice? Some people like solutions with a small number of factions (terms) as opposed to larger numbers. Some people like it if the bottoms are all or mostly small integers. Other people favor huge numbers and lots of terms. Can you find a way to say what's good/bad? Sums of denominators?

What questions can you ask?

1. Can you write every fraction as a sum of unit fractions?
2. Do the methods we mentioned (greedy, greedy even, greedy powers of twos, denominators with lots of factors) always work?

3. How can you find the shortest (smallest number of terms) or smallest (lowest big denominator) sums for a given fraction?

4. How many ways can you write a given fraction as a sum of unit fractions? Here $1 = 1/2 + 1/3 + 1/6$ seemed significant as we could take any decomposition into unit fractions and then multiply one of those fractions by 1 written in this special form to get as decomposition with more fractions in it. Can we make it so the unit fractions are all different?

5. How can we eliminate repeated unit fractions? In $1/13 + 1/13$ can we just move on to replace one of the $1/13$ with a sum of unit fractions with bigger denominators? in some systematic way?

6. Can every fraction be written as a sum of 2 unit fractions?

7. Can you find a fraction that needs 3 unit fractions? Or 4 or n?

8. What about decomposing fractions of particular forms?

   (a) $2/n$ (We can assume n is odd!) Using greedy we get
   $$\frac{2}{2n+1} = \frac{1}{n+1} + \frac{1}{(n+1)(2n+1)},$$
   which says something about some of the other questions we've asked.

   (b) $2/(pq)$ where p, q are prime. Distinct or not.

   (c) $3/n$, $4/n$?

   (d) By the way, no one knows whether every fraction $4/n$ can be written as the sum of just 3 unit fractions. People have tried pretty hard to prove this and it is know for n up to about 10^{14}. So we're talking about some interesting questions here!

9. If we require ourselves to pick denominators for unit fractions in such a way that they get bigger and bigger we might not be able to get them unit fractions to add up to enough.

10. How are the denominators of the unit fractions in decomposition for a given fraction related? Can you predict anything about them?

11. Can you "work forwards" instead of backwards and describe all the fractions you can write with 2 unit fractions, then 3, and so on?

We ended up concentrating on the "greedy powers of 2 in denominator" method and tried to determine whether it always worked or not.

0.1 Egyptian fractions

Ancient Egyptians only worked with unit fractions and would never have written 5/8 (the way they wrote fractions wouldn't even allow it) so they'd write $1/2 + 1/8$ instead. (Well, they did have a symbol for 2/3)

The method they used for multiplication was based on successive doubling. So, to multiply an integer time a fraction $1/n$, they needed to work with (fractions that we'd write as $2/n, 4/n$ and so on. This meant that multiplication required knowing how to write $2/n$ as a sum of unit fractions. It also meant that, in considering different ways to write $1/n$ as a sum of unit fractions, the preferred representations in which the denominators were even and small.

Table of how they wrote $2/n$ can you guess how they got these?

0.2 The greedy algorithm

The idea is to start with a/b and take away from it the largest unit fraction that is smaller than it. The apply this same procedure to the difference until the difference is itself a unit fraction.

Examples:

$$\frac{12}{53} = \frac{1}{5} + \frac{7}{265}$$
$$\frac{7}{265} = \frac{1}{38} + \frac{1}{10070}$$

Showing that
$$\frac{12}{53} = \frac{1}{5} + \frac{1}{38} + \frac{1}{10070}.$$

Also,
$$4/37 = 1/10 + 3/370$$
$$3/370 = 1/124 + 1/22940$$

The question here is to explain why this process always stops after some number of steps by reaching a remainder that is a unit fraction.

We observed in a number of examples that the numerators of the remaining fraction (after the largest possible iunit fraction is subtracted) get smaller with each step. TO explain why this is actually the case,

Starting with a/b with $a < b$, write $b = ak+l$ where $k \geq 1$ and $1 \leq k \leq a-1$. This means that
$$\frac{1}{k+1} \leq \frac{a}{b} < \frac{1}{k}$$
so that $\frac{1}{k+1}$ is the largest unit fraction we can take from a/b.

Then the difference is
$$\frac{a}{b} - \frac{1}{k+1} = \frac{a(k+1)-b}{b(k+1)}$$

and the numerator of this remainder is

$$ak + a - b = ak + a - (ak + l) = a - l < a,$$

showing that the denominators decrease with each step of the algorithm and so must eventually reach 1.

```
def gcd(a, b):
    while b !=0:
       t = b
       b = a % b
       a = t
    return a

#This does greedy powers of 2
#MAY NOT TERMINATE!
def greedy_twos(AA,BB):
    #a=12
    #b=53
    a=AA
    b=BB
    t=1
    while a>1:
        while ( a*t < b) or( a*t==b) :
            t=2*t
        c=a*t-b
        d=b*t
        f=gcd(c,d)
        c=int(c/f)
        d=int(d/f)
        print a,"/",b,"=",1,"/",t,"+",c,"/",d
        a=c
        b=d

#here's the plain greedy algorithm
#ALWAYS TERMINATES
def greedy(AA,BB):
    a=AA
    b=BB
    t=1
    while a>1:
        while (a*t<b) or (a*t==b):
            t=t+1
        c=a*t-b
        d=b*t
        f=gcd(c,d)
        c=int(c/f)
        d=int(d/f)
        print a,"/",b,"=",1,"/",t,"+",c,"/",d
        a=c
        b=d

greedy(12,53)

greedy(4,37)
```