Math 581d, Fall 2010, Homework 3

William Stein

October 13, 2010

Do the following 3 problems, and turn them in by email (`wstein@gmail.com`) before the beginning of class on Wednesday, October 20, 2010. As usual, you can find the latex of this file at `http://wstein.org/edu/2010/581d/hw/`.

1. Write "a page" of Python code that does something of interest to you which uses *all* of the following Python features:
   - generator expressions, e.g., `(a*a for a in xrange(10))`
   - classes, e.g., `class Foo:`
   - comments, e.g., `# this is a comment`
   - list comprehensions, e.g., `[a*a for a in range(10) if a%2 == 0]`.
   - decorators, e.g., `@foo`
   - regular expressions, e.g., `import re; # do something with re.[tab]`

2. In this problem, you will compare integer arithmetic using the MPIR library (as wrapped in Sage) with the integer arithmetic implemented in Python itself. Please state exactly which version of Sage you are using, and on what computer.

   (a) Figure out how to create a Sage integer n with 5 digits and a Python integer m with 5 digits. The following must be true about their types if you did this problem correctly:

   ```
   sage: type(n)
   <type 'sage.rings.integer.Integer'>
   sage: type(m)
   <type 'int'>
   ```

   (b) What is the type of $n + m$?

   (c) For each of the following values of d, make a d-bit Sage integer n and a d-bit Python integer m and time how long computing `n*n` and `m*m` takes. You can time evaluation of an expression as follows (type `timeit?` for more help):

   ```
   sage: timeit('n*n')
   625 loops, best of 3: 114 ns per loop
   ```

   The values of d are: 4, 31, 64, 128, 512, 4096, 1048576.

(d) What algorithms for integer multiplication does Python implement? You might have to look at the source code of Python to determine this (maybe it is documented online). Remark: Some interesting problems in math can be encoded as "multiply these two integers"; this problem helps you see that the software you choose to do the multiplication can have an impact on how long it takes! Search for "A Trillion Triangles" in Google to see an example.

(e) Express an opinion: Why do *you* think Python doesn't implement asymptotically fast integer multiplication? Why is it acceptable to the Python developers that multiplying some million digit integers can easily take a hundred times longer in Python than in Sage (or using the gmpy library).

3. (a) Construct (in Sage) your instructor's favorite elliptic curve, which is defined by the equation

$$y(y+1) = (x-1)x(x+2).$$

If you don't know what to do, type `EllipticCurve?` in Sage.

(b) Find a pair of rational numbers (x, y), where each of x and y have numerator and denominator with at least 5 digits, such that $y(y+1) = (x-1)x(x+2)$.

(c) Use the `integral_points` method of an elliptic curve to find all pairs (x, y) of integers such that $y(y+1) = (x-1)x(x+2)$. (Warning: Note that points are only returned up to sign, so if you get back a point P in the output list, you do not get the point $-P$, unless of course $P = -P$.)