PMAT 527/627 - Final Project

Expectations

The project will consist of a written portion and an in class presentation. The length of a project can vary considerably, but here are some rough guidelines:

- an expository essay should be about 10-12 pages (individual) or 16-20 pages (group of 2).
- an essay accompanying a coding project should be long enough; perhaps 4-6 pages is reasonable.

You may work on a project either alone or in a group of two. If working in a group you should include a separate page outlining how each person contributed to the final result.

Structure and Writing

Your essay should contain an abstract and introduction. For coding projects you should include mathematical background on the algorithm, a description of choices you made in implementation (variants, how you chose parameters....) and a section on the performance of your code (theoretical and practical). The structure of an expository essay will depend on the topic you're investigating.

There are many resources available on mathematical writing; see Conrad [3] and Goss [7] for some brief guidelines, or Knuth, Larrabee and Roberts [9] for a more extensive treatment. The quality of your writing will affect your grade on the project.

Documentation

Code without documentation is hard to read and understand. Writing documentation as you write code can also make it easier to think about the algorithms involved. Writing comments in the body of your code is useful. You may also want to consult the Sage Developer's Guide for conventions on docstrings in Sage.

Examples and Tests

You should provide examples of your code working. Some examples may belong in a docstrings of individual functions; others could fit more appropriately into the documentation at the top of your source code.

Programming Language

If you choose a coding project, the default language will be Sage and Python. You are welcome to explore other languages in order to improve the performance of your code, but doing so is not required. Cython is one option that fits in well with Python: see `http://www.sagemath.org/doc/developer/coding_in_cython.html`.

Project Ideas

Coding Projects

One possibility for a final project is to implement an interesting algorithm. The following list gives some ideas for algorithms, each of which has various possible optimizations and choices involved. If you choose this kind of project you will also need to write a short essay giving some mathematical background for the algorithm as well as a description of the choices you made in the implementation.

The following ideas are drawn from our textbooks, but you don't need to limit yourself: if there's a number theoretic algorithm you're interested in implementing that's not found in the textbooks feel free to discuss it with me.

- Quadratic sieve. See Crandall-Pomerance [4, §6.1].
- Number field sieve. See Crandall-Pomerance [4, §6.2].
- Fast elliptic curve primality proving. See Crandall-Pomerance [4, §7.6.3].
- Schoof-Elkies-Atkin algorithm. References in Crandall-Pomerance [4, §7.5.2].

Sage Projects

Another option for a coding project is to try to improve number theory in Sage. Some such projects will focus on implementing a particular algorithm, while others will involve making an algebraic structure better. Here are some ideas for reasonable projects within Sage; you can find more by consulting `trac.sagemath.org` or talking to me.

Some of these projects will require knowledge of Cython, which we won't focus on this semester. If you choose to pursue one of these you should look at the Sage Developer's Guide.

- Improve finite fields in Sage. The case of fields of order p^k with $p > 2$, $k > 1$ and $p^k > 2^{16}$ needs particular attention.
- Isogenies, automorphisms and complex multiplication of elliptic curves over number fields. See trac tickets #7368 and #9977
- Isogenies of elliptic curves over finite fields. See trac ticket #11093.
- Linear dependence of points on elliptic curves. See trac ticket #9335.
- Computing Hilbert class polynomials of real quadratic fields. See Cohen-Roblot [2].

Expository Projects

- Counting primes. See Crandall-Pomerance [4, §3.7], trac ticket #11475, and Shoup [11, Chap. 5].
- Arithmetic with large integers. See Crandall-Pomerance [4, Chap. 9] and the GMP manual [6].
- p-adic methods for point counting. See Satoh [10] and Kedlaya [8].
- Computing Galois groups of number fields. See Cohen [1, §6.3] for background, though you should also look at more recent references.
- Computing class groups of number fields. See Cohen [1, §6.5] and more recent references.
- Computing rings of integers in number fields Cohen [1, §6.1] and more recent references.
- Algorithms for elliptic curves. See Cremona [5, Chap. 3].
- Modular forms. See Stein [12].
- Factoring algorithms for polynomials over $\mathbb{Z}$. See Cohen [1, §3.5] and more recent references.

More resources

Here are some places you can look for more ideas.

- Henri Cohen's book [1] has a lot of algorithms for number theory which could be interesting to implement.
- `http://archives.math.utk.edu/topics/numberTheory.html` has a list of interesting topics in number theory.
- Mathematics of Computation journal has a lot of papers describing algorithms, some of which apply to number theory.
- The number theory arXiv has lots of number theory papers, some of which are computational in nature.

Timeline

Oct. 31: An outline is due (worth 5% of the final grade). For a coding project you should lay out the infrastructure of important classes and functions in your implementation and document what they should do. For an expository project you should collect a list of references you'll be using and lay out the general structure of your paper.

Nov. 26-30: I will meet with each group outside of class to practice your presentations. You don't need to have finished everything, but you should have your presentation in close to final form.

Dec. 3-7: Presentations in class (worth 10% of the final grade). Each group will get 20-30 minutes to describe the highlights of your project.

Dec. 17: Final deadline for the written component of the project (worth 85% of the final grade).

References

[1] H. Cohen. A Course in Computational Algebraic Number Theory. Berlin: Springer-Verlag, 1993.

[2] H. Cohen and X. Roblot. Computing the Hilbert class field of real quadratic fields. Mathematics of Computation **69** (1999), no. 231, pp. 1229-1244.

[3] K. Conrad. Errors in mathematical writing. `http://www.math.uconn.edu/~kconrad/math216/mathwriting.pdf`

[4] R. Crandall and C. Pomerance. Prime Numbers: A Computational Perspective, 2nd edition. New York: Springer, 2005.

[5] J. Cremona. Algorithms for Modular Elliptic Curves, 2nd edition. Cambridge: Cambridge UP, 1997. `http://homepages.warwick.ac.uk/~masgaj/book/fulltext/`

[6] GNU Multiple Precision Arithmetic Library: Algorithms. `http://gmplib.org/manual/Algorithms.html`.

[7] D. Goss. Some Hints on Mathematical Style. `http://www.math.osu.edu/~goss.3/hint.pdf`.

[8] K. Kedlaya. Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology. Journal of the Ramanujan Mathematical Society **16** (2001), pp. 323-338. `http://arxiv.org/abs/math/0105031`

[9] D. Knuth, T. Larrabee and P. Roberts. Mathematical Writing. `http://tex.loria.fr/typographie/mathwriting.pdf`

[10] T. Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. Journal of the Ramanujan Mathematical Society **15** (2000), pp. 247-270.

[11] V. Shoup. A Computational Introduction to Number Theory and Algebra, 2nd edition. Cambridge: Cambridge UP, 2008. `http://shoup.net/ntb/ntb-v2.pdf`.

[12] W. Stein. Modular Forms, a Computational Approach. Providence: American Math. Society, 2007. `http://modular.math.washington.edu/msri06/refs/stein-book-on-modular-forms.pdf`