A CCLI Type II Idea List

William Stein

September 2, 2009

Funding estimates are of course difficult. An undergraduate working part time (10 hours/week) costs about \$2,000/quarter. An undergraduate or graduate student working fulltime for two months during the summer costs at least \$5,000. A professional with a Ph.D. typically costs at least \$5,000/month, but can be highly effective.

I envision people funded to work on any of these projects being funded to complete the *project* rather than being funded for a certain amount of time. Typically when I give times below, I'm overestimating how long I think the project would take *me*. Though it is notoriously difficult to estimate how long software engineering projects will take, at least I have some experience (from organizing 18 Sage Days workshops) watching people propose and complete projects.

1. Write self-contained *subject-specific tutorials* for the following subject areas: number theory, combinatorics, linear algebra, 2d and 3d graphics, programming (including Cython), group theory, cryptography, and commutative algebra.

   Cost: **\$2,000/each**. About 2 weeks full-time work for each tutorial, including 1 week writing and 1 week editing and incorporating user feedback.

2. Address all known *security vulnerabilities in the Sage notebook*, and implement a more sophisticated user security model. Yoav Aner, a cryptography graduate student in London, has written an extensive threat assessment for the notebook as his Masters Thesis, which provides a list of issues to tackle.

   Cost: **\$4,000**. About 1 month fulltime work: 2 weeks implementing a user security model, and 2 weeks addressing security vulnerabilities. More time would be valuable.

3. Add complete *spreadsheet capabilities* to the Sage notebook. This would be similar to Microsoft Excel spreadsheets, but the formulas would be Sage code. This could be extremely powerful for applications that are very data-centric, yet involve advanced mathematics.

   Cost: **\$8,000**. About 2 months fulltime work – 1 month finding and integrating an existing AJAX spreadsheet library (such as `http://www.simple-groupware.`

`de/cms/Spreadsheet/Home`, but there are about 10 to choose from) into the Sage notebook, and 1 month making it interoperate with the Sage library.

4. Implement *additional 3d plotting functionality.* There are many functions and options for 3d plots that have not yet been implemented, and this could be done. Also improve Sage's ability to render using HTML5's canvas, including animation output, as something that can be embedded in a webpage, or viewed separately.

   Cost: **$4,000**. This is an open-ended project, but 1 month fulltime would provide enough time to improve implicit plotting, options for parametric and function plotting, and for adding text rendering and axes to canvas rendering.

5. *Improve support for testing end-user code* via the Sage doctest system. Support for this exists in Sage, but often has bugs or inconsistencies in behavior. The most important part of this project would be creating a script which **tests** this behavior, and making it a part of the standard Sage test suite.

   Cost: **$1,000**. This is one week of fulltime work.

6. Create a *C-library interface to GAP*. Stein spent 1 week fulltime during Summer 2009 and created a proof of concept of this library interface, which proved that this project is do-able, and that it would directly result in up to 2000 times speedups over the current Sage/GAP pipe-based interface.

   Cost: **$12,000**. 1 month to implement the GAP C library interface, and 1 month to switch Sage over to use it instead of the pipe interface. Sage uses the pipe interface right now for most group theory, and switching over and optimizing the results will uncover many issues, and also require rewriting some of the Sage library. It will also make it reasonable to use GAP for basic arithmetic in many interesting new cases (e.g., Lie Algebras).

7. We have had one "Bug Days" workshop in San Diego, during which the participants fixed over 200 venerable bugs in Sage (over 1/3 of all known bugs), and did a triage of all bugs. Another *Bug Days workshop* would be a great way to substantially improve the quality of Sage, and fix many longstanding issues. This workshop would be organized to encourage undergraduate student participation.

   Cost: **$12,000**.

8. Once there is C-library interface to GAP, sponsor a Sage Days workshop to *expose to Sage as much as possible of GAP's functionality*, e.g., characters of finite groups.

   Cost: **$12,000**. This would be a good topic for a Sage Days workshop, since it is fairly easy to implement a wide range of things in parallel. This workshop would be organized to encourage undergraduate student participation.

9. Fund work by Carl Witty to complete his implementation of *Cylindrical Algebraic Decomposition* for Sage. Cylindrical Algebraic Decomposition (CAD) is an algorithm for finding descriptions of semi-algebraic sets, i.e., the real solution sets of systems of multivariate polynomial equalities and inequalities. Currently CAD is available in Mathematica and in the standalone program QEPCAD ("Quantifier Elimination by Partial Cylindrical Algebraic Decomposition" – see `http://www.usna.edu/Users/cs/qepcad/B/QEPCAD.html`). CAD is used for exactly solving many problems for polynomials or systems of polynomials over the reals, including finding solution sets, finding maxima or minima, quantifier elimination, etc. For example, in Mathematica, all of Minimize, Maximize, Reduce, FullSimplify, and SparseArray indirectly use CAD when operating on systems of real polynomial equations (or certain trigonometric equations that can be automatically transformed into polynomials) and no simpler algorithm is available.

   Cost: **$10,000**.

10. Implement native *statistics* functionality in Sage, beginning with a class for holding data with descriptions, methods for subselection, transformation and assignments, and then more classes with data/object interactions that wrap or use `scipy.stats` and `rpy`.

    Cost: **$5,000**. 1 month fulltime work.