COMMUNICATION

Beyond the Black Box

Jeroen Demeyer, William Stein, and Ursula Whitcher

"Commercial computer algebra systems are black boxes, and their algorithms are opaque to the users," complained a trio of mathematicians whose "misfortunes" are detailed in a recent *Notices* article [2]. "We reported the bug on October 7, 2013…By June 2014, nothing had changed …All we could do was wait."

In open-source software the code underlying each operation is available to anyone who chooses to look at it. Because open-source code can be checked directly, it is highly valuable for replicable, peer-reviewable research. We are users and developers of SageMath, open-source software by and for the mathematical community (see [3]). SageMath is a full computer algebra system that can be installed locally or accessed freely in a web browser through the SageMathCloud. Sage also provides a consistent interface to many free libraries of mathematical software.

the code underlying each operation is available to anyone

Open source is not a panacea: errors can arise in open-source software as well as in closed-source software. We trace the history of a representative bug in Sage to illustrate the role the mathematical community plays in detecting and fixing bugs in open-source software. The unfortunate trio [2] found a bug in Mathematica's algorithm for determinants of large integer matrices. SageMath evaluates that determinant correctly. For the sake of comparison, we describe a different past problem with Sage's computation of determinants and the process followed in resolving it.

The first step in fixing a bug is that somebody must notice a problem. In our case, SageMath release 5.6 stalled when computing the determinant of a large integer matrix; the computation never completed.

Next, the person who notices the bug must report it. Typically, a user would post a question to the sage-support or sage-devel Google group. Asking questions simultaneously warns active SageMath users about the problem and advertises for developers who might fix it. Bugs can also be reported by emailing `bugs@sagemath.com`. A very confident user might move straight to the next step, creating a ticket.

Proposed changes to the SageMath code are tracked on the SageMath trac server. Individual bugs or code enhancements are called tickets. In January 2013 our determinant bug became ticket 14032, with the initial title *determinant() of large integer matrices broken* (see [4]). The person who creates a ticket chooses a priority for the fix from the options of "trivial", "minor", "major", "critical", and "blocker". Most SageMath tickets are given the middle ranking of "major"; our determinant bug was "critical". A new version of SageMath cannot be released if there are open tickets marked "blocker".

Once the ticket is created, developers can start hunting down the bug's origins. In our case, the first clue was that stalling arose when computing determinants of integer matrices of size greater than 50×50. This was the threshold where SageMath 5.6 invoked a p-adic algorithm to calculate determinants of integer matrices. SageMath includes several different implementations of algorithms to compute integer determinants. The default algorithm depends on the input but can be explicitly overridden. Choosing a different algorithm made the computation work, so it was clear that the problem was with the p-adic algorithm.

How is the p-adic algorithm supposed to work? Suppose we are given an $n \times n$ integer matrix A. Checking whether the determinant of A is 0 is relatively quick, so let us assume that $\det(A) \neq 0$. The first step is to choose a random integer vector v and a prime number p. We may find a rational vector x that solves the matrix equation $Ax = v$ using p-adic approximation. That is, we solve

Jeroen Demeyer is postdoctoral assistant at the University of Ghent (Belgium) and research engineer for the European project OpenDreamKit at the University Paris-Sud (France). His email address is `jdemeyer@cage.ugent.be`.

William Stein is professor of mathematics at the University of Washington. His email address is `wstein@uw.edu`.

Ursula Whitcher is associate editor for Math Reviews *and associate professor at the University of Wisconsin–Eau Claire. Her email address is* `whitchua@uwec.edu`.

For permission to reprint this article, please contact: `reprint-permission@ams.org`.

DOI: http://dx.doi.org/10.1090/noti1408

$A\bar{x} = b \pmod{p^m}$ for some m; if we take m to be sufficiently large, we may recover x from $\bar{x}$. Now, one can prove, using Cramer's rule, that the least common multiple of the denominators of the entries of x will be a divisor d of $\det(A)$. With high probability, $\det(A)/d$ will be not only an integer but a tiny integer. We can reconstruct $\det(A)$ completely by finding its value modulo a few different prime numbers using the row-echelon form of A in fields of prime order and then applying the Chinese Remainder Theorem. The Hadamard bound on the determinant of a matrix, which compares the determinant to the product of the Euclidean norms of its columns, bounds the size of the prime numbers we need to check; this guarantees that we can complete the reconstruction in a finite amount of time.

Further testing of the p-adic determinant algorithm in SageMath showed that it worked for very large integer matrices as well as for small ones. One of us, Jeroen Demeyer, a postdoc at Ghent University, discovered that in fact the computation stalled only for $51 \leq n \leq 63$. He changed the title of the ticket to *determinant() of integer matrices of size in [51,63] broken* and began trying to figure out why SageMath treated these matrices differently. The problem, he discovered, lay in the implementation of the last part of the p-adic algorithm, where SageMath tried to find $\det(A)$ "modulo a few prime numbers." When a prime p is large, SageMath computed determinants $\pmod{p}$ using the code for determinants over $\mathbb{Z}$. A recent tweak to another part of Sage's matrix code had changed the definition of "large p" to be $p > 2^{23}$ (that is, prime numbers greater than 8388593). When $n \leq 63$, Sage's integer matrix determinant function called code that asked for $\det(A)$ modulo primes greater than 8388593. This produced an infinite loop: computing the determinant of an integer matrix called for the determinant $\pmod{p}$, which called for the determinant of an integer matrix.

Once he found the bug's origin, Demeyer quickly wrote a patch to fix it. The next step is peer review: before someone's code is incorporated into Sage, another developer must test it and sign off. Review often entails multiple rounds of suggestions and fixes, but in this case Demeyer's fix worked well the first time, and the mathematical physicist Volker Braun (then a postdoc at the Dublin Institute for Advanced Studies) approved it for inclusion in Sage just two days after the bug was reported. Less than a month later, SageMath 5.7, which included the fix for this bug, was released.

Part of Demeyer's patch was a test to ensure similar problems would not arise in the future. Each new release of SageMath goes through a series of tests to ensure that it works as advertised. Demeyer's patch requires new versions of the SageMath code to compute the determinants of random integer matrices up to 80×80 successfully. Furthermore, SageMath now checks that $\det(A^2) = \det(A)^2$ for a random square integer matrix A. Automatic testing provides SageMath with resilience and makes it easier for multiple developers to contribute to different parts of the code base. Currently, about 94 percent of the functions in SageMath are automatically tested. By default, new releases of SageMath use FLINT, the Fast Library for Number Theory, to compute the determinants of integer matrices. For most integer matrices A of size 24×24 and higher, FLINT computes the determinant $\det(A)$ by calculating $\det(A)$ modulo small primes. The primes are chosen so that their product is greater than twice the Hadamard bound. When the entries of A are small in comparison to the size of the matrix, however, FLINT still uses the p-adic lifting algorithm. For more information about SageMath's current and past determinant algorithms, see [1].

References

[1] Jeroen Demeyer, William Stein, and Ursula Whitcher, *Beyond the black box*, preprint, April 2016. arXiv:1604.08472 [math.HO]

[2] Antonio J. Durán, Mario Pérez, and Juan L. Varona, The misfortunes of a trio of mathematicians using computer algebra systems, *Notices Amer. Math. Soc.* **61** (2014), 1249–1252. MR3155350

[3] David Joyner and William Stein, Open source mathematical software, *Notices Amer. Math. Soc.* **54** (2007), 1279.

[4] SageMath ticket #14032, *determinant() of integer matrices of size in [51,63] broken*, `trac.sagemath.org/ticket/14032`.

Credits

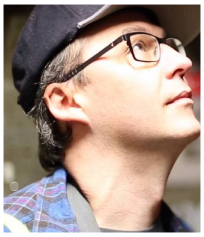
Photo of Jeroen Demeyer courtesy of Jeroen Demeyer.
Photo of William Stein courtesy of Dennis Stein.
Photo of Ursula Whitcher courtesy of UWEC Learning and Technology Services Media Production.

ABOUT THE AUTHORS

Jeroen has been a SageMath developer since 2010 and release manager of SageMath from 2011 to 2013.

Jeroen Demeyer

William is the founder of the SageMath mathematical software project.


William Stein

Ursula is a SageMath developer. She has been active in the community since 2008 and leads projects for Women in Sage.

Ursula Whitcher