Math::Prime::Util
Adventures with Math & Crypto

Dana Jacobsen

24 June 2014

Genesis

- In early 2012, needed a lot of primes for another module
- Pure Perl sieve
  - far too slow
- Math::Prime::XS
  - still not fast enough, too much memory
- Math::Prime::FastSieve
  - Close. Submitted patch, now fast enough
- Decided to make my own module
- First release June 2012, now on 42nd release

Features

- Contains almost all functionality of:

- Math::Prime::XS
- Math::Prime::FastSieve
- Math::Factor::XS
- Math::Big::Factors
- Math::Factoring
- Math::Primality
- Math::Prime::TiedArray
- Crypt::Primes
- Math::ModInt::ChineseRemainder
- Integer::Partition

- And many things not in other modules

Functions

primes
next_prime
prev_prime
forprimes
prime_iterator
prime_iterator_object
prime_count
prime_count_lower
prime_count_upper
prime_count_approx
nth_prime
nth_prime_lower
nth_prime_upper
nth_prime_approx
twin_prime_count
twin_prime_count_approx
nth_twin_prime
nth_twin_prime_approx

factor
factor_exp
divisors
fordivisors
divisor_sum

prime_precalc
prime_memfree
prime_get_config
prime_set_config

is_prime
is_prob_prime
is_provable_prime
is_provable_prime_with_cert
prime_certificate
verify_prime
is_pseudoprime
is_strong_pseudoprime
is_lucas_pseudoprime
is_strong_lucas_pseudoprime
is_almost_extra_strong_lucas_pseudoprime
is_extra_strong_lucas_pseudoprime
is_frobenius_underwood_pseudoprime
is_aks_prime
miller_rabin_random

random_prime
random_ndigit_prime
random_nbit_prime
random_strong_prime
random_proven_prime
random_proven_prime_with_cert
random_maurer_prime
random_maurer_prime_with_cert
random_shawe_taylor_prime
random_shawe_taylor_prime_with_cert

Math::Prime::Util::PrimeArray
Math::Prime::Util::PrimeIterator

primorial pn_primorial
moebius mertens
euler_phi jordan_totient
carmichael_lambda
exp_mangoldt
liouville
chebyshev_theta
chebyshev_psi
lucas_sequence
partitions
forpart
ExponentialIntegral
LogarithmicIntegral
RiemannZeta
RiemannR
consecutive_integer_lcm
gcd lcm
gcdext chinese
valuation
invmod
vecsum
is_power
kronecker
binomial
znorder
znprimroot
znlog
legendre_phi

Design decisions (1)

- Functions

```
say prime_count(1e11);
```

  OO

```
my $obj = new MPU(…);
say $obj->prime_count(1e11);
```

- Inputs
  - Initially was for native ints only
  - Now supports bigints in Perl with Math::BigInt. Portable but very slow.
  - Math::Prime::Util::GMP allows fast bigints

Design decisions (2)

- Many modules or one module?   I chose one.
- Input validation.   `is_prime(1.5); is_prime("foo");`
- Calling overhead can dominate time
  - As much as possible, only XS, including validation
  - Load and call Perl only when necessary
  - Environment variable to disable XS if desired.

Design decisions (3)

- Portability. This is a *library*.
  - 32-bit and 64-bit
  - With or without GMP
  - MSWin32, AIX, Solaris, HP-UX, Linux
  - gcc, clang, etc.
  - Thread safe

- Support back to 5.6.2
  - Very painful for 64-bit
  - Perl and many modules often turn UVs into NVs
  - This is disastrous for number theory

Applications

- Simple operations: Primes, primality, factoring
- Simple tasks: RosettaCode, OEIS, Project Euler
- Number Theory: record prime gaps, primality proofs
- Debugging other packages
  - Crypt::Primes can return composites
  - FLINT / SAGE n_is_prime can return true for composites.
  - Math::Pari isprime can return true for composites
  - Perl6 #.is-prime can return true for composites
  - Sympy Similar to Perl6: slow & known counterexamples
- Crypto

Crypto

- Math::Pari
  - Lots of CPAN crypto modules use this.
  - Based on Pari 2.1 – about 10 years out of date
  - Doesn't build correctly on 64-bit Windows
  - Would like an alternative
- Crypt::Random => Bytes::Random::Secure (or other)
- Crypt::Primes => Math::Prime::Util
- Crypt::RSA => Alt::Crypt::RSA::BigInt
  - Drop in replacement
  - Fixes 10+ open defects
- Crypt::DSA => Crypt::DSA::GMP
  - Same API so mostly a simple change of module
  - Fixes 20+ open defects
  - Adds FIPS 186-4 functionality, interoperability tests, and more
  - Requires GMP for performance
  - Faster

A Perl success story

- Fastest open source prime count and nth_prime
  - Many orders of magnitude faster than sieving

- Fastest 64-bit primality testing (and deterministic)

- Random [proven] primes and primality tests needed for crypto modules

- With Math::Prime::Util::GMP installed:

- Can generate proofs for 128-bit primes in milliseconds
- Fastest bigint probable prime test to 10k digits
- Fastest open source AKS and ECPP (primality proofs)
- ECPP verifier being used at factordb
- First known occurrence prime gaps
  - Using this module in Perl for last seven months
  - Over 20% of all current record gaps from this.

Conclusion

Math::Prime::Util

Math::Prime::Util::GMP

Available on CPAN now!

Thanks: Wojciech Izykowski, Christian Bau, Kim Walisch, bulk88, William Hart, Paul Zimmerman, and lots of mathematicians.

Examples

- `say prime_count(1e14);    # 3204941750802 in ~2 seconds`

- ```
  say join " ",
  factor("1591437872382662009392357398799382773784842577362757");
  # 435905229083 889976781491420 3 4102219867919815386812 93 in ~1s
  ```

- `forprimes { say } 1e18, 1e18+1000`

- ```
  my $nsemiprimes = 0;           # Count semiprimes by brute force
  forcomposites {
    $nsemiprimes++ if scalar factor($_) == 2
  } 1e8-1;
  say "Number of semiprimes less than 1e8: $nsemiprimes";
  ```

- ```
  my ($limit, $sum, $pc) = (1e8-1, 0, 1);       # Clever way
  forprimes {
    $sum += prime_count(int($limit/$_)) + 1 - $pc++;
  } int(sqrt($limit));
  say $sum;
  ```
  ```