Contents

1 Functions 2
1.1 ecpp – elliptic curve primality proving 2
1.1.1 ecpp – elliptic curve primality proving 2
1.1.2 hilbert – Hilbert class polynomial 2
1.1.3 dedekind – Dedekind's eta function 3
1.1.4 cmm – CM method 3
1.1.5 cmm_order – CM method with order 3
1.1.6 cornacchiamodify – Modified cornacchia algorithm 4

Chapter 1

Functions

1.1 ecpp – elliptic curve primality proving

The module consists of various functions for ECPP (Elliptic Curve Primality Proving).

It is probable that the module will be refactored in the future so that each function be placed in other modules.

The ecpp module requires mpmath.

1.1.1 ecpp – elliptic curve primality proving

ecpp(`n`: *integer*, `era`: *list*=None) → *bool*

Do elliptic curve primality proving.
If `n` is prime, return True. Otherwise, return False.

The optional argument `era` is a list of primes (which stands for ERAtosthenes).

`n` must be a big integer.

1.1.2 hilbert – Hilbert class polynomial

hilbert(`D`: *integer*) → *(integer, list)*

Return the class number and Hilbert class polynomial for the imaginary quadratic field with fundamental discriminant `D`.

Note that this function returns Hilbert class polynomial as a list of coefficients.

†If the option **HAVE_NET** is set, at first try to retrieve the data in http://hilbert-class-polynomial.appspot.com/. If the data corresponding to `D` is not found, compute the Hilbert polynomial directly (for a long time).

`D` must be negative int or long. See [1].

1.1.3 dedekind – Dedekind's eta function

dedekind(`tau`: *mpmath.mpc*, `floatpre`: *integer*) → *mpmath.mpc*

Return Dedekind's eta of a complex number `tau` in the upper half-plane.

Additional argument `floatpre` specifies the precision of calculation in decimal digits.

`floatpre` must be positive int.

1.1.4 cmm – CM method

cmm(`p`: *integer*) → *list*

Return curve parameters for CM curves.

If you also need its orders, use **cmm_order**.

A prime `p` has to be odd.
This function returns a list of (`a`, `b`), where (`a`, `b`) expresses Weierstrass' short form.

1.1.5 cmm_order – CM method with order

cmm_order(`p`: *integer*) → *list*

Return curve parameters for CM curves and its orders.

If you need only curves, use **cmm**.

A prime `p` has to be odd.
This function returns a list of (`a`, `b`, `order`), where (`a`, `b`) expresses Weierstrass' short form and `order` is the order of the curve.

1.1.6 cornacchiamodify – Modified cornacchia algorithm

cornacchiamodify(d: ***integer***, p: ***integer***) → ***list***

Return the solution $(u,\ v)$ of $u^2 - \mathtt{d}v^2 = 4\mathtt{p}$.

If there is no solution, raise ValueError.

p must be a prime integer and d be an integer such that $\mathtt{d} < 0$ and $\mathtt{d} > -4\mathtt{p}$ with $\mathtt{d} \equiv 0, 1 \pmod 4$.

Examples

```
>>> ecpp.ecpp(300000000000000000053)
True
>>> ecpp.hilbert(-7)
(1, [3375, 1])
>>> ecpp.cmm(7)
[(6L, 3L), (5L, 4L)]
>>> ecpp.cornacchiamodify(-7, 29)
(2, 4)
```

Bibliography

[1] Richard Crandall and Carl Pomerance. *Prime Numbers*. Springer, 1st. edition, 2001.