Contents

Chapter 1

Functions

1.1 squarefree – Squarefreeness tests

There are two method groups. A function in one group raises **Undetermined** when it cannot determine squarefreeness. A function in another group returns `None` in such cases. The latter group of functions have "_ternary" suffix on their names. We refer a set $\{$`True`, `False`, `None`$\}$ as *ternary*.

The parameter type *integer* means either *int*, *long* or **Integer**.

This module provides an exception class.

Undetermined : Report undetermined state of calculation. The exception will be raised by **lenstra** or **trivial_test**.

1.1.1 Definition

We define squarefreeness as:
n is squarefree $\iff$ there is no prime p whose square divides n.

Examples:

- 0 is non-squarefree because any square of prime can divide 0.
- 1 is squarefree because there is no prime dividing 1.
- 2, 3, 5, and any other primes are squarefree.
- 4, 8, 9, 12, 16 are non-squarefree composites.
- 6, 10, 14, 15, 21 are squarefree composites.

1.1.2 lenstra – Lenstra's condition

lenstra(`n`: *integer*) → *bool*

If return value is True, `n` is squarefree. Otherwise, the squarefreeness is still unknown and **Undetermined** is raised. The algorithm is based on [1].

†The condition is so strong that it seems `n` has to be a prime or a Carmichael number to satisfy it.

Input parameter `n` ought to be an odd **integer**.

1.1.3 trial_division – trial division

trial_division(n: ***integer***) → ***bool***

Check whether `n` is squarefree or not.

The method is a kind of trial division and inefficient for large numbers.

Input parameter `n` ought to be an **integer**.

1.1.4 trivial_test – trivial tests

trivial_test(n: ***integer***) → ***bool***

Check whether `n` is squarefree or not. If the squarefreeness is still unknown, then **Undetermined** is raised.

This method do anything but factorization including Lenstra's method.

Input parameter `n` ought to be an odd **integer**.

1.1.5 viafactor – via factorization

viafactor(n: ***integer***) → ***bool***

Check whether `n` is squarefree or not.

It is obvious that if one knows the prime factorization of the number, he/she can tell whether the number is squarefree or not.

Input parameter `n` ought to be an **integer**.

1.1.6 viadecomposition – via partial factorization

viadecomposition(n: ***integer***) → ***bool***

Test the squarefreeness of `n`. The return value is either one of `True` or `False`; `None` never be returned.

The method uses partial factorization into squarefree parts, if such partial factorization is possible. In other cases, It completely factor `n` by trial division.

Input parameter `n` ought to be an **integer**.

1.1.7 lenstra_ternary – Lenstra's condition, ternary version

lenstra_ternary(n: *integer*) → *ternary*

Test the squarefreeness of `n`. The return value is one of the ternary logical constants. If return value is `True`, `n` is squarefree. Otherwise, the squarefreeness is still unknown and `None` is returned.

†The condition is so strong that it seems `n` has to be a prime or a Carmichael number to satisfy it.
This is a ternary version of **lenstra**.

Input parameter `n` ought to be an odd **integer**.

1.1.8 trivial_test_ternary – trivial tests, ternary version

trivial_test_ternary(n: *integer*) → *ternary*

Test the squarefreeness of `n`. The return value is one of the ternary logical constants.

The method uses a series of trivial tests including **lenstra_ternary**.
This is a ternary version of **trivial_test**.

Input parameter `n` ought to be an **integer**.

1.1.9 trial_division_ternary – trial division, ternary version

trial_division_ternary(n: *integer*) → *ternary*

Test the squarefreeness of `n`. The return value is either one of `True` or `False`; `None` never be returned.

The method is a kind of trial division.
This is a ternary version of **trial_division**.

Input parameter `n` ought to be an **integer**.

1.1.10 viafactor_ternary – via factorization, ternary version

viafactor_ternary(`n`: ***integer***) → ***ternary***

Just for symmetry, this function is defined as an alias of **viafactor**.

Input parameter `n` ought to be an **integer**.

Bibliography

[1] H. W. Lenstra, Jr. Miller's primality test. *Information processing letters*, Vol. 8, No. 2, 1979.