egcd

Release 0.4.0

Andrei Lapets

Jul 13, 2022

CONTENTS

Easy-to-import library with a basic, efficient, pure-Python implementation of the extended Euclidean algorithm.

CHAPTER

ONE

INSTALLATION AND USAGE

This library is available as a package on PyPI:

```
python -m pip install egcd
```

The library can be imported in the usual way:

```
from egcd import egcd
```

1.1 Examples

The function `egcd` is an efficient implementation of the extended Euclidean algorithm. It accepts two integer inputs `b` and `n`, returning a tuple of the form `(gcd(b, n), a, m)` where the three integers in the tuple satisfy the identity `(a * b) + (n * m) == gcd(b, n)`:

```
>>> egcd(1, 1)
(1, 0, 1)
>>> egcd(12, 8)
(4, 1, -1)
>>> egcd(23894798501898, 23948178468116)
(2, 2437250447493, -2431817869532)
>>> egcd(pow(2, 50), pow(3, 50))
(1, -260414429242905345185687, 408415383037561)
```

CHAPTER

TWO

DEVELOPMENT

All installation and development dependencies are fully specified in `pyproject.toml`. The `project.optional-dependencies` object is used to specify optional requirements for various development tasks. This makes it possible to specify additional options (such as `docs`, `lint`, and so on) when performing installation using pip:

```
python -m pip install .[docs,lint]
```

2.1 Documentation

The documentation can be generated automatically from the source files using Sphinx:

```
python -m pip install .[docs]
cd docs
sphinx-apidoc -f -E --templatedir=_templates -o _source .. && make html
```

2.2 Testing and Conventions

All unit tests are executed and their coverage is measured when using pytest (see the `pyproject.toml` file for configuration details):

```
python -m pip install .[test]
python -m pytest
```

Alternatively, all unit tests are included in the module itself and can be executed using doctest:

```
python src/egcd/egcd.py -v
```

Style conventions are enforced using Pylint:

```
python -m pip install .[lint]
python -m pylint src/egcd
```

2.3 Contributions

In order to contribute to the source code, open an issue or submit a pull request on the GitHub page for this library.

2.4 Versioning

Beginning with version 0.1.0, the version number format for this library and the changes to the library associated with version number increments conform with Semantic Versioning 2.0.0.

2.5 Publishing

This library can be published as a package on PyPI by a package maintainer. First, install the dependencies required for packaging and publishing:

```
python -m pip install .[publish]
```

Ensure that the correct version number appears in the `pyproject.toml` file and in any links to this package's Read the Docs documentation that exist in this README document. Also ensure that the Read the Docs project for this library has an automation rule that activates and sets as the default all tagged versions. Create and push a tag for this version (replacing `?.?.?` with the version number):

```
git tag ?.?.?
git push origin ?.?.?
```

Remove any old build/distribution files. Then, package the source into a distribution archive using the wheel package:

```
rm -rf build dist src/*.egg-info
python -m build --sdist --wheel .
```

Finally, upload the package distribution archive to PyPI using the twine package:

```
python -m twine upload dist/*
```

2.5.1 egcd module

Easy-to-import library with a basic, efficient, pure-Python implementation of the extended Euclidean algorithm.

`egcd.egcd.`**`egcd`**(*b: int*, *n: int*) → Tuple[int, int, int]

Given two integers `b` and `n`, returns `(gcd(b, n), a, m)` such that `a*b + n*m == gcd(b, n)`.

This implementation is adapted from the sources below:

- Extended Euclidean Algorithm on Brilliant.org,
- Modular inverse on Rosetta Code,
- Algorithm Implementation/Mathematics/Extended Euclidean algorithm on Wikibooks, and
- Euclidean algorithm on Wikipedia.

```
>>> egcd(1, 1)
(1, 0, 1)
>>> egcd(12, 8)
(4, 1, -1)
>>> egcd(23894798501898, 23948178468116)
(2, 2437250447493, -2431817869532)
>>> egcd(pow(2, 50), pow(3, 50))
(1, -260414429242905345185687, 408415383037561)
```

The example below tests the behavior of this function over a range of inputs using the built-in `math.gcd` function.

```
>>> from math import gcd
>>> checks = []
>>> for (b, n) in [(b, n) for b in range(200) for n in range(200)]:
...     (g, a, m) = egcd(b, n)
...     checks.append(g == a*b + n*m)
...     checks.append(g == gcd(b, n))
>>> all(checks)
True
```

PYTHON MODULE INDEX

e

INDEX