Contents

1 Classes **3**

1.1 poly.uniutil – 一変数多項式のためのユーティリティ 3

1.1.1 RingPolynomial – 可換環上の多項式 4

1.1.1.1 getRing 5

1.1.1.2 getCoefficientRing 5

1.1.1.3 shift_degree_to 5

1.1.1.4 split_at 5

1.1.2 DomainPolynomial – 整域上の多項式 5

1.1.2.1 pseudo_divmod 7

1.1.2.2 pseudo_floordiv 7

1.1.2.3 pseudo_mod 7

1.1.2.4 exact_division 7

1.1.2.5 scalar_exact_division 7

1.1.2.6 discriminant 8

1.1.2.7 to_field_polynomial 8

1.1.3 UniqueFactorizationDomainPolynomial – UFD 上の多項式 8

1.1.3.1 content 8

1.1.3.2 primitive_part 8

1.1.3.3 subresultant_gcd 9

1.1.3.4 subresultant_extgcd 9

1.1.3.5 resultant 9

1.1.4 IntegerPolynomial – 有理整数環上の多項式 9

1.1.5 FieldPolynomial – 体上の多項式 10

1.1.5.1 content 11

1.1.5.2 primitive_part 11

1.1.5.3 mod 11

1.1.5.4 scalar_exact_division 11

1.1.5.5 gcd 11

1.1.5.6 extgcd 11

1.1.6 FinitePrimeFieldPolynomial – 有限素体上の多項式 12

1.1.6.1 mod_pow – モジュロとべき乗 13

1.1.6.2 pthroot 13

1.1.6.3 squarefree_decomposition 13

1.1.6.4 distinct_degree_decomposition 13

1.1.6.5 split_same_degrees 14
1.1.6.6 factor 14
1.1.6.7 isirreducible 14
1.1.7 polynomial – さまざまな多項式に対するファクトリ関数 . 14

Chapter 1

Classes

1.1 poly.uniutil – 一変数多項式のためのユーティリティ

- **Classes**
  - **RingPolynomial**
  - **DomainPolynomial**
  - **UniqueFactorizationDomainPolynomial**
  - **IntegerPolynomial**
  - **FieldPolynomial**
  - **FinitePrimeFieldPolynomial**
  - OrderProvider
  - DivisionProvider
  - PseudoDivisionProvider
  - ContentProvider
  - SubresultantGcdProvider
  - PrimeCharacteristicFunctionsProvider
  - VariableProvider
  - RingElementProvider
- **Functions**
  - **polynomial**

1.1.1 RingPolynomial – 可換環上の多項式

Initialize (Constructor)

RingPolynomial(`coefficients`: ***terminit***, `coeffring`: ***CommutativeRing***, `**keywords`: ***dict***)
→ ***RingPolynomial object***

多項式を与えられた係数環 `coeffring` で初期化.

このクラスは **SortedPolynomial**, **OrderProvider** そして **RingElementProvider** から継承.

`coefficients` の型は **terminit**. `coeffring` は **CommutativeRing** のサブクラスのインスタンス.

Methods

1.1.1.1 getRing

getRing(`self`) → ***Ring***

多項式の所属する `Ring` のサブクラスのオブジェクトを返す.
(このメソッドは RingElementProvider 内の定義をオーバーライドする)

1.1.1.2 getCoefficientRing

getCoefficientRing(`self`) → ***Ring***

全ての係数が所属する `Ring` サブクラスのオブジェクトを返す.
(このメソッドは RingElementProvider 内の定義をオーバーライドする)

1.1.1.3 shift_degree_to

shift_degree_to(`self`, `degree`: ***integer***) → ***polynomial***

次数が与えられた `degree` である多項式を返す. より正確に, $f(X) = a_0 + ... + a_n X^n$ とすると, `f.shift_degree_to(m)` は以下を返す:

- もし f が零多項式なら, 零多項式を返す
- $a_{n-m} + ... + a_n X^m$, $(0 \leq m < n)$
- $a_0 X^{m-n} + ... + a_n X^m$, $(m \geq n)$

(このメソッドは OrderProvider から継承される)

1.1.1.4 split_at

split_at(`self`, `degree`: ***integer***) → ***polynomial***

与えられた次数で分割された二つの多項式のタプルを返す. 与えられた次数の項は, もし存在するなら, 下の次数の多項式の側に属する.
(このメソッドは OrderProvider から継承される)

1.1.2 DomainPolynomial – 整域上の多項式

Initialize (Constructor)

DomainPolynomial(`coefficients`: ***terminit***, `coeffring`: ***CommutativeRing***, `**keywords`: ***dict***)
→ ***DomainPolynomial object***

与えられた整域 `coeffring` に対し多項式を初期化.

基本的な多項式の演算に加え, それは擬除算を持つ.

このクラスは **RingPolynomial** と **PseudoDivisionProvider** を継承.
`coefficients` の型は **terminit**. `coeffring` は `coeffring.isdomain()` を満たす **CommutativeRing** のサブクラスのインスタンス.

Methods

1.1.2.1 pseudo_divmod

pseudo_divmod(`self`, `other`: ***polynomial***) → ***tuple***

以下のような多項式 Q, R のタプル `(Q, R)` を返す:

$$d^{deg(f)-deg(other)+1}f = other \times Q + R,$$

d は `other` の主係数.
(このメソッドは PseudoDivisionProvider から継承される)

1.1.2.2 pseudo_floordiv

pseudo_floordiv(`self`, `other`: ***polynomial***) → ***polynomial***

以下のような多項式 Q を返す:

$$d^{deg(f)-deg(other)+1}f = other \times Q + R,$$

d は `other` の主係数.
(このメソッドは PseudoDivisionProvider から継承される)

1.1.2.3 pseudo_mod

pseudo_mod(`self`, `other`: ***polynomial***) → ***polynomial***

以下のような多項式 R を返す:

$$d^{deg(f)-deg(other)+1}f = other \times Q + R,$$

d は `other` の主係数.
(このメソッドは PseudoDivisionProvider から継承される)

1.1.2.4 exact_division

exact_division(`self`, `other`: ***polynomial***) → ***polynomial***

(割り切れるとき) 除算の商を返す.
(このメソッドは PseudoDivisionProvider から継承される)

1.1.2.5 scalar_exact_division

scalar_exact_division(`self`, `scale`: ***CommutativeRingElement***)
→ ***polynomial***

各係数を割り切る `scale` による商を返す.
(このメソッドは PseudoDivisionProvider から継承される)

1.1.2.6 discriminant

discriminant(self) → ***CommutativeRingElement***

多項式の判別式を返す.

1.1.2.7 to_field_polynomial

to_field_polynomial(self) → ***FieldPolynomial***

整域 D 上の多項式環を D の商体へ埋め込むことにより得られる `FieldPolynomial` オブジェクトを返す.

1.1.3 UniqueFactorizationDomainPolynomial – UFD 上の多項式

Initialize (Constructor)

UniqueFactorizationDomainPolynomial(coefficients: ***terminit***, coeffring: ***CommutativeRing***, **keywords: ***dict***)
→ ***UniqueFactorizationDomainPolynomial object***

与えられた UFD `coeffring` において多項式を初期化.

このクラスは **DomainPolynomial**, **SubresultantGcdProvider** そして **ContentProvider** から継承する.

`coefficients` の型は **terminit**. `coeffring` は `coeffring.isufd()` を満たす **CommutativeRing** のサブクラスのインスタンス.

1.1.3.1 content

content(self) → ***CommutativeRingElement***

多項式の内容を返す.
(このメソッドは ContentProvider から継承される)

1.1.3.2 primitive_part

primitive_part(self) → ***UniqueFactorizationDomainPolynomial***

多項式の原始的部分を返す.
(このメソッドは ContentProvider から継承される)

1.1.3.3 subresultant_gcd

subresultant_gcd(`self`, `other`: ***polynomial***) → ***UniqueFactorizationDomainPolynomial***

与えられた多項式の最大公約数を返す．これらは多項式環に入っていなければならず, その係数環は UFD でなければならない.
(このメソッドは SubresultantGcdProvider から継承される)
Reference: [1]Algorithm 3.3.1

1.1.3.4 subresultant_extgcd

subresultant_extgcd(`self`, `other`: ***polynomial***) → ***tuple***

$A \times self + B \times other = P$ である (A，B，P) を返す．P は与えられた多項式の最大公約数．これは多項式環に入っていなければならず, その係数環は UFD でなければならない.
参考: [2]p.18
(このメソッドは SubresultantGcdProvider から継承される)

1.1.3.5 resultant

resultant(`self`, `other`: ***polynomial***) → ***polynomial***

`self` と `other` の終結式を返す.
(このメソッドは SubresultantGcdProvider から継承される)

1.1.4 IntegerPolynomial – 有理整数環上の多項式

Initialize (Constructor)

IntegerPolynomial(`coefficients`: ***terminit***, `coeffring`: ***CommutativeRing***, `**keywords`: ***dict***)
→ ***IntegerPolynomial object***

与えられた可換環 `coeffring` において多項式を初期化.

組み込みの int/long へ特別な初期化がされなければならないので, このクラスは必要とされる.

このクラスは **UniqueFactorizationDomainPolynomial** から継承.

`coefficients` の型は **terminit**．`coeffring` は **IntegerRing** のインスタンス.
冗長なように思えるが, 有理整数環を与える必要がある.

1.1.5 FieldPolynomial – 体上の多項式

Initialize (Constructor)

FieldPolynomial(`coefficients`: ***terminit***, `coeffring`: ***Field***, **`keywords`: ***dict***)
→ ***FieldPolynomial object***

与えられた体 `coeffring` において多項式を初期化.

体上の多項式環はユークリッド整域なので, 除算が提供される.

このクラスは **RingPolynomial**, **DivisionProvider** そして **ContentProvider** から継承.

`coefficients` の型は **terminit**. `coeffring` は **Field** のサブクラスのインスタンス.

Operations

operator	explanation
`f // g`	切り捨て除算の商
`f % g`	余り
`divmod(f, g)`	商と余り
`f / g`	有利関数体上での除算

Methods

1.1.5.1 content

content(`self`) → ***FieldElement***

多項式の内容を返す.
(このメソッドは ContentProvider から継承される)

1.1.5.2 primitive_part

primitive_part(`self`) → ***polynomial***

多項式の原始的部分を返す.
(このメソッドは ContentProvider から継承される)

1.1.5.3 mod

mod(`self`, `dividend`: ***polynomial***) → ***polynomial***

$dividend$ mod $self$ を返す.
(このメソッドは DivisionProvider から継承される)

1.1.5.4 scalar_exact_division

scalar_exact_division(`self`, `scale`: ***FieldElement***)
→ ***polynomial***

各係数を割り切る `scale` による商を返す.
(このメソッドは DivisionProvider から継承される)

1.1.5.5 gcd

gcd(`self`, `other`: ***polynomial***) → ***polynomial***

self と other の最大公約数を返す.

返される多項式はすでにモニック多項式です.
(このメソッドは DivisionProvider から継承される)

1.1.5.6 extgcd

extgcd(`self`, `other`: ***polynomial***) → ***tuple***

タプル `(u, v, d)` を返す; 二つの多項式 `self` と `other` の最大公約数 d と以下となる u, v である

$$d = self \times u + other \times v$$

extgcd を参照.
(このメソッドは DivisionProvider から継承される)

1.1.6 FinitePrimeFieldPolynomial – 有限素体上の多項式

Initialize (Constructor)

FinitePrimeFieldPolynomial(`coefficients`: ***terminit***, `coeffring`: ***FinitePrimeField***, **`keywords`: ***dict***)
→ ***FinitePrimeFieldPolynomial object***

与えられた可換環 `coeffring` において多項式を初期化.

このクラスは **FieldPolynomial** と **PrimeCharacteristicFunctionsProvider** から継承する.

`coefficients` の型は **terminit**. `coeffring` は **FinitePrimeField** のサブクラスのインスタンス.

Methods

1.1.6.1 mod_pow – モジュロとべき乗

mod_pow(`self`, `polynom`: ***polynomial***, `index`: ***integer***)
→ ***polynomial***

$polynom^{index} \bmod self$ を返す.

`self` を法としていることに注意.
(このメソッドは PrimeCharacteristicFunctionsProvider から継承される)

1.1.6.2 pthroot

pthroot(`self`) → ***polynomial***

X^p を X に渡すことにより得られる多項式を返す. p は標数. もし多項式が p 乗された項のみ成さなければ, 結果は無意味.
(このメソッドは PrimeCharacteristicFunctionsProvider から継承される)

1.1.6.3 squarefree_decomposition

squarefree_decomposition(`self`) → ***dict***

平方因子を含まない多項式分解を返す.

返される値は,keys が整数で values が対応したべき乗因子の辞書. 例えば, もし

Examples

```
>>> A = A1 * A2**2
>>> A.squarefree_decomposition()
{1: A1, 2: A2}.
```

(このメソッドは PrimeCharacteristicFunctionsProvider から継承される)

1.1.6.4 distinct_degree_decomposition

distinct_degree_decomposition(`self`) → ***dict***

多項式を相異なる次数で因数分解したものを返す.

返される値は keys が整数で values が対応した次数の因数の積である辞書. 例えば, もし $A = A1 \times A2$, で, そして $A1$ の全ての既約因子が次数 1 を持ち, $A2$ の既約因子は次数 2 を持つ, そして結果は: `{1: A1, 2: A2}`.

与えられた多項式は平方因子をもなたいものでなければならず, その係数環は有限体でなければならない.
(このメソッドは PrimeCharacteristicFunctionsProvider から継承される)

1.1.6.5 split_same_degrees

split_same_degrees(`self`, `degree`: *)* → ***list***

多項式の既約因子を返す.

多項式は与えられた次数の既約因子の積でなければならない.
(このメソッドは PrimeCharacteristicFunctionsProvider から継承される)

1.1.6.6 factor

factor(`self`) → ***list***

多項式を因数分解する.

返される値は, 最初の成分は因数で次の成分はその重複度であるタプルのリストです.
(このメソッドは PrimeCharacteristicFunctionsProvider から継承される)

1.1.6.7 isirreducible

isirreducible(`self`) → ***bool***

もし多項式が既約なら `True` を返し, さもなくば `False` を返す.
(このメソッドは PrimeCharacteristicFunctionsProvider から継承される)

1.1.7 polynomial – さまざまな多項式に対するファクトリ関数

polynomial(`coefficients`: ***terminit***, `coeffring`: ***CommutativeRing***) → ***polynomial***

多項式を返す.

† 関数を呼ぶ前に以下を設定することにより, 係数環から多項式の型を選ぶ方法をオーバーライドすることができる:

```
special_ring_table[coeffring_type] = polynomial_type
```

.

Bibliography

[1] Henri Cohen. *A Course in Computational Algebraic Number Theory*. GTM138. Springer, 1st. edition, 1993.

[2] Kida Yuuji. Integral basis and decomposition of primes in algebraic fields (Japanese). http://www.rkmath.rikkyo.ac.jp/~kida/intbasis.pdf.