Contents

Chapter 1

Classes

1.1 poly.termorder – 項順序

- **Classes**
  - †**TermOrderInterface**
  - †**UnivarTermOrder**
  - **MultivarTermOrder**
- **Functions**
  - **weight_order**

1.1.1 TermOrderInterface – 項順序のインターフェース

Initialize (Constructor)

TermOrderInterface(`comparator`: ***function***) → ***TermOrderInterface***

項順序は主に二つの項 (または単項式) の優先順位を決定する関数です. 優先順位により, 全ての項は順序付けられます.

`Python` の項でより正確に, 項順序は整数の二つのタプルをとり, それぞれは項のべき指数を表し, 組み込み関数の cmp のようにただ 0, 1 または -1 を返す.

A `TermOrder` オブジェクトは優先順位関数だけでなく, 次数を出す, 主係数, などの文字列を多項式へフォーマットするためのメソッドも提供する.

`comparator` は整数の二つのタプルのようなオブジェクトをとり, それぞれは項のべき指数を表し, 組み込み関数 `cmp` のようにただ 0, 1 または -1 を返す.

このクラスは抽象クラスで例示化されるべきではない. k のメソッドは下にオーバーライドされなければならない.

Methods

1.1.1.1 cmp

cmp(`self`, `left`: *tuple*, `right`: *tuple*) → *integer*

二つのインデックスタプル `left` と `right` を比較し優先順位を決定する.

1.1.1.2 format

format(`self`, `polynom`: *polynomial*, **`keywords`: *dict*)
→ *string*

多項式 `polynom` のフォーマットされた文字列を返す.

1.1.1.3 leading_coefficient

leading_coefficient(`self`, `polynom`: *polynomial*) → *CommutativeRingElement*

多項式 `polynom` の項順序に関連する主係数を返す.

1.1.1.4 leading_term

leading_term(`self`, `polynom`: *polynomial*) → *tuple*

多項式`polynom`の主項を項順序に関連したタプル`(degree index, coefficient)`として返す.

1.1.2 UnivarTermOrder – 一変量多項式に対する項順序

Initialize (Constructor)

UnivarTermOrder(`comparator`: *function*) → *UnivarTermOrder*

一変量多項式に対しては一意的な項順序がある. 次数として知られている.
一変量の場合への特別なことは, べき数はタプルではないが, 最低限の整数であるということです. 事実によると, メソッドシグナチャーもまた TermOrderInterface 内の定義から変換する必要がありますが, それは簡単なため説明は省略します.

`comparator` は二つの整数をとり,`cmp` のようにただ 0, 1 または -1 を返すために呼ばれ得る, すなわち, もしそれらが 0 を返す, 最初は 1 より大きい, そしてさもなくば -1. 理論上は期待できる比較器は `cmp` 関数のみです.

このクラスは TermOrderInterface を継承する.

Methods

1.1.2.1 format

format(`self`, `polynom`: ***polynomial***, `varname`: ***string***=**'X'**, `reverse`: ***bool***=**False**)
→ ***string***

多項式 `polynom` のフォーマットされた文字列を返す.

- `polynom` は一変量多項式でなければならない
- `varname` は変数名の配置をされ得る.
- `reverse` は `True` と `False` のどちらかになり得る. もしそれが `True` なら, 項は逆 (降) 順で現れる.

1.1.2.2 degree

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

多項式 `polynom` の次数を返す.

1.1.2.3 tail_degree

tail_degree(`self`, `polynom`: ***polynomial***) → ***integer***

`polynom` の全ての項の中での最小次数を返す.

このメソッドは *experimental* です.

1.1.3 MultivarTermOrder – 多変数多項式に対する項順序

Initialize (Constructor)

MultivarTermOrder(`comparator`: ***function***) → ***MultivarTermOrder***

このクラスは **TermOrderInterface** を継承する.

Methods

1.1.3.1 format

format(self, polynom: *polynomial*, varname: *tuple*=None, reverse: *bool*=False, **kwds: *dict*)
→ *string*

多項式 polynom のフォーマットされた文字列を返す.

追加の独立変数である varnames は変数名が必要とされる.

- polynom は多変数多項式です.
- varnames は変数名の列.
- reverse は True と False のどちらかになり得る. もしそれが True, 項は逆 (降) 順で現れる.

1.1.4 weight_order – 重量順序付け

weight_order(weight: *sequence*, tie_breaker: *function*=None)
→ *function*

weight による重量比較付けの比較器を返す.

w を weight をします. 重量順序付けは独立変数 x と y によって定義され, それらは以下を満たします. もし $w \cdot x < w \cdot y$ or $w \cdot x == w \cdot y$ なら $x < y$ で, タイブレイカーが $x < y$ と教えます.

設定 tie_breaker はもし重量ベクトルのドット積が独立変数 tie から離れたなら使われるもう一つの比較器. もしその設定が None (初期設定) で, 与えられた独立変数を順序付けする必要がタイブレイカーが本当にいるなら, a TypeError が起こる.

Examples

```
>>> w = termorder.MultivarTermOrder(
...     termorder.weight_order((6, 3, 1), cmp))
>>> w.cmp((1, 0, 0), (0, 1, 2))
1
```