CoCoALib - Feature #1210

New Fn: make public "coefficients" (of poly wrt a given PP basis)

03 Aug 2018 17:18 - John Abbott

Status:	Closed	**Start date:**	03 Aug 2018
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	New Function	**Estimated time:**	3.99 hours
Target version:	CoCoALib-0.99850	**Spent time:**	3.95 hours

Description

In SparsePolyOps-MinPoly.C (near line 70) there is a private fn for computing the coeffs of a poly wrt to a PP basis.

Clean up the interface, and make the fn public.

Related issues:

Related to CoCoALib - Slug #1009: coefficients for MinPoly e Frobenius	**Closed**	**16 Feb 2017**

History

#1 - 03 Aug 2018 17:28 - John Abbott

What should the fn signature be? Should there be more than one signature?

- **(1)** void coeffs(vector<RingElem& C, ConstRefRingElem f, const vector<PPMonoidElem>& PPbasis)
- **(2)** vector<RingElem> coeffs(ConstRefRingElem f, const vector<PPMonoidElem>& PPbasis)

Signature (1) might work well if we want to used a fixed "workspace".
Signature (2) should not be inefficient if we use C++ "move" operator.
JAA has a preference for (2) at the moment.

The impl can be efficient if PPbasis is in strictly decr order:

- **(A)** if we use vector<PPMonoidElem> then it requires some work to guarantee that the elements are distinct and strictly decreasing (and in the same PPMonoid)
- **(B)** we could repr the PPbasis as the sum of its entries: this is automatically sorted, the elements are distinct, and they are guaranteed in the same PPMonoid

Approach **(B)** seems to be more foolproof, but possibly a little unnatural?

Discuss; decide; implement.

#2 - 03 Aug 2018 17:28 - John Abbott

- Related to Slug #1009: coefficients for MinPoly e Frobenius added

#3 - 01 Oct 2019 12:02 - John Abbott

- Target version changed from CoCoALib-0.99650 November 2019 to CoCoALib-0.99800

#4 - 27 Feb 2020 14:55 - John Abbott

- Status changed from New to Resolved

- Assignee set to John Abbott

- Target version changed from CoCoALib-0.99800 to CoCoALib-0.99700

- % Done changed from 0 to 80

- Estimated time set to 1.99 h

Name: **CoeffVecWRTBasis** (or CoeffListWRTBasis in CoCoA-5)

I have a first (KISS) impl: it takes 2 RingElem, and uses the support of the second arg as the PP basis.

Documented, visible in CoCoA-5 (with doc). About to check in.

The fn gives error if the poly is not in the span of the PP basis.

#5 - 04 Mar 2020 17:15 - Anna Maria Bigatti

John Abbott wrote:

> Name: **CoeffVecWRTBasis** (or CoeffListWRTBasis in CoCoA-5)

I suggest calling it **CoeffVecWRTSupport** (List in CoCoA-5) because it is very similar to
calling coefficients(f, support(g));.
Should it be **CoeffVecWRTRevSupport**?

More precisely I suggest not to use the word Basis which could be misleading (or we should check there are no extra PPs in f)

#6 - 04 Mar 2020 18:20 - John Abbott

OK, I have chosen **CoeffVecWRTSupport**. Changes already made (but not yet checked in).

NOTE I prefer not CoeffVecWRTRevSupport because it is really too long.

#7 - 06 Mar 2020 15:39 - John Abbott

- Target version changed from CoCoALib-0.99700 to CoCoALib-0.99800

- Estimated time changed from 1.99 h to 3.99 h

I think there is good reason to make the other version of this function available, incl. the one Anna proposed, namely CoeffVec(f, QB) where QB is an ordered list of PPs.

It could also be useful to have even more general versions available (possibly with a different interface). For this reason I prefer to **postpone** to the next version, so that we have time to think about the design (& to implement).

#8 - 05 Apr 2021 15:19 - John Abbott

How to impl neatly and efficiently the version CoeffsWRT(f,QB)?

Perhaps create a list of pairs (PP,index); sort it by the PP-ordering.
Then use the existing fn to obtain the coeffs, and finally undo the permutation.
A bit tedious, but should not be too slow... I hope.

#9 - 04 Feb 2022 22:01 - John Abbott

- Target version changed from CoCoALib-0.99800 to CoCoALib-0.99850

#10 - 17 Feb 2022 19:37 - John Abbott

If we require that the supplied QB be in order (incr or decr?)
then the question about permutations goes away.

Can anyone think of a circumstance where a non-ordered QB would be useful?

#11 - 08 Aug 2022 20:41 - John Abbott

- Status changed from Resolved to Closed

- % Done changed from 80 to 100

I have opted for the KISS solution -- we can make fancier versions in the future is there is need for them!
I have cleaned the code slightly (and added some comments).
I have added a simple test (to test-SparsePolyRing2.C).

Closing!