CoCoALib - Feature #800

PPMonoidSparse: impl of sparse PPs

09 Nov 2015 13:33 - John Abbott

Status:	Closed	**Start date:**	09 Nov 2015
Priority:	Urgent	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	New Function	**Estimated time:**	8.99 hours
Target version:	CoCoALib-0.99850	**Spent time:**	9.00 hours

Description

Implement a PPMonoid which uses a sparse repr for the PPs.

This is useful when there are many indets.

Related issues:

Related to CoCoA-5 - Slug #798: use poly ring with many variables is too slow	**Closed**	**09 Nov 2015**
Related to CoCoALib - Feature #802: DivMask: extend interface?	**In Progress**	**10 Nov 2015**
Related to CoCoALib - Feature #803: PPOrdering: use it to compute WDeg?	**In Progress**	**11 Nov 2015**
Related to CoCoALib - Slug #842: PPMonoidSparse: comparisons are VERY SLOW	**New**	**15 Feb 2016**
Related to CoCoA-5 - Support #242: CoCoA-5 Projects for students (e.g. credit...	**In Progress**	**28 Sep 2012**

History

#1 - 09 Nov 2015 13:35 - John Abbott

I had assumed that this issue already existed, but a quick search did not find it.

There is already a partial implementation (dating back to 2005/2007). Complete it!

#2 - 10 Nov 2015 14:49 - John Abbott

- Status changed from New to In Progress

- Assignee set to John Abbott

- % Done changed from 0 to 50

There was a very incomplete impl in PPMonoidSparse.C, which was already recognized by Makefile.

Apparently the impl has long exponents; might a "big exp" version also be interesting? Anyway, I'll proceed with the long version; if we want a BigInt version later, it should be largely a copy-and-paste job :-)

Currently the impl uses internally a std::list of pairs (index, exp). The advantage of using a std::list over a std::vector is apparent only in myMulByIndetPower where a new factor may need to be inserted.

I now think that a std::vector based impl would probably be better (assuming that myMulByIndetPower is not called often when it is necessary to insert a new factor in the "middle" of the PP).

Another problem is how to handle well the various term orderings. Currently the code assumes lex which is clearly too limiting!

#3 - 10 Nov 2015 15:16 - John Abbott

The currently missing functions are: myCmp, myWDeg and myCmpWDeg (& a partial version). Also myComputeDivMask.

How to implement these functions reasonably efficiently? Is efficiency even important?

Surely efficiency is important for myCmp, especially with the most common PP ordering (namely, StdDegRevLex). How to achieve this? One possible solution is to create a temporary vector into which the exponents will be placed, and then use a standard "dense" algorithm, but that seems to defeat the purpose of a sparse repr.

Might it be important to store the StdDeg along with the list of (index, exp) pairs? If it is not stored, then it must be computed (linear cost) for each comparison.

#4 - 10 Nov 2015 16:45 - John Abbott

For Wdeg I am inclined to compute it "on the fly" every time it is needed. If someone reports poor execution speed, and profiling fingers WDeg as the culprit then I can consider storing the WDeg explicitly in the PP repr.

The question for the standard degree is probably different...

#5 - 10 Nov 2015 18:39 - Anna Maria Bigatti

John Abbott wrote:

> Might it be important to store the StdDeg along with the list of (index, exp) pairs? If it is not stored, then it must be computed (linear cost) for each comparison.

well, as you always tell me: make the clean implementation first.... ;-)

#6 - 24 Nov 2015 15:15 - John Abbott

- % Done changed from 50 to 60

I have a first (complete) version which seems to work OK -- just simple tests.
I even managed to create a sparse poly ring with 1000000 indets! :-)

#7 - 24 Nov 2015 17:26 - John Abbott

Cleaned the PPMonoidSparse code; there were still a couple of incomplete fns.

#8 - 24 Nov 2015 17:28 - John Abbott

Should the NewPolyRing fns automatically choose PPMonoidSparse if there are "lots of" indets? If so, how many are "lots of"? 20? 30? 40?

Comments? Suggestions?

#9 - 02 Feb 2016 14:15 - John Abbott

- Priority changed from Normal to Urgent

#10 - 15 Feb 2016 13:54 - John Abbott

- Related to Slug #842: PPMonoidSparse: comparisons are VERY SLOW added

#11 - 16 Jun 2016 17:09 - John Abbott

- Target version changed from CoCoALib-0.99540 Feb 2016 to CoCoALib-0.99560

As reported in issue #842 execution speed is sometimes very disappointing, so currently it may not be such a good idea to arrange for NewPolyRing to use a PPMonoidSparse yet. Nevertheless, the principle is good; I did look at the code hoping to make the change quickly, but it looked to be trickier than I'd expected :-(

#12 - 06 Nov 2017 15:08 - John Abbott

- Target version changed from CoCoALib-0.99560 to CoCoALib-0.99600

#13 - 12 Jun 2018 18:29 - John Abbott

- Target version changed from CoCoALib-0.99600 to CoCoALib-0.99650 November 2019

#14 - 01 Oct 2019 12:01 - John Abbott

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

#15 - 09 Feb 2022 20:12 - John Abbott

- Status changed from In Progress to Resolved

- % Done changed from 60 to 80

The code is there, and there is a test: test-PPMonoidSparse1.C. However many lines are commented out in the test.
Documentation? Pah! D is for wimps! Well, I've added a few lines to PPMonoidSparse.C

I'm tempted to close this issue for 0.99720, and make a new one for improving the impl.
Ideally the repr should depend on the term order: e.g. DegRevLex favours having the IndexExp entries in reverse order!
Exercise for student?

#16 - 09 Feb 2022 20:13 - John Abbott

- Related to Support #242: CoCoA-5 Projects for students (e.g. crediti F and tesi) added

#17 - 14 Feb 2022 18:23 - John Abbott

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

#18 - 09 Mar 2023 22:15 - John Abbott

- Status changed from Resolved to Closed

- % Done changed from 80 to 100

- Estimated time set to 8.99 h

It make sense to close this issue, and let issue #842 deal with the disappointing execution speed.