CoCoALib - Slug #1756

deg(f) is slow if f is long

20 Jul 2023 09:29 - John Abbott

Status:	In Progress	**Start date:**	20 Jul 2023
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	50%
Category:	Improving	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99880	**Spent time:**	2.50 hours

Description

In a poly ring with a std deg compatible order we can compute deg(f) just by looking at the first PP.
It seems we do not do this:

```
/**/ t0 := CpuTime(); LD := [deg(g) | i in 1..1000]; TimeFrom(t0);
2.620
/**/ t0 := CpuTime(); LD := [deg(g) | i in 1..10000]; TimeFrom(t0);
25.956
/**/ D := deg(f);
/**/ t0 := CpuTime(); LD := [D | i in 1..10000]; TimeFrom(t0);
0.013
```

The original situation was collecting all elements of a list less than deg(f)

History

#1 - 20 Jul 2023 09:32 - John Abbott

Oddly, i believed I had recently sorted this out... evidently not.
We need a function to say whether the ordering on a polyring is std deg compatible.

Here is SparsePolyOps-RingElem.C:275:

```
  // ANNA: add check if ordering is StdDeg compatible and return StdDeg(LPP)
  long SparsePolyRingBase::myStdDeg(ConstRawPtr rawf) const
```

#2 - 05 Aug 2023 11:04 - John Abbott

Is this an easy task which we could quickly resolve?

Do we have a test case?

```
/**/ PrevPrime(10^8);
99999989
/**/ f := cyclotomic(It,x);
/**/ t0 := CpuTime();
/**/ deg(f);
99999988
/**/ TimeFrom(t0);
2.719
```

Maybe try PrevPrime(5*10^7) instead? As the example above requires quite a lot of memory.

#3 - 05 Aug 2023 11:08 - John Abbott

Lex is clearly not std deg compatible, but if the LPP contains only the last indet, we could short-cut. Not sure this is worth it.

If the weights are positive but not std deg compat, we might be able to estimate a lower bound, and so avoid scanning the whole poly? Need to think about this... not sure it is really that important (but it might be fun to think about it).

#4 - 03 Oct 2023 18:38 - John Abbott

- Status changed from New to In Progress

- Assignee set to John Abbott

- % Done changed from 0 to 50

I think I have mostly solved this issue now. Anna had already added a **IsStdGraded** function in 2013. I have modified the impl of myDeg so that it actually uses IsStdGraded. Tests suggest that all is well. Hope to check in soon.

NOTES I am not too convinced by the name IsStdGraded when applied to a PPordering. Should we make convenience variations to be applied to PPMonoid and/or SparsePolyRing?

#5 - 03 Oct 2023 18:54 - John Abbott

What do we want to do if the OrdMat has a first row being all the same but not equal to 1?
The grading is StdDeg compatible, but the weighted degree is not the same as the StdDeg.

#6 - 05 Oct 2023 20:17 - John Abbott

ANNA! I'd like to discuss the last comment above

#7 - 06 Oct 2023 12:23 - Anna Maria Bigatti

We could have, instead of **IsStdGraded** a function **IsStdGradedCompatible**, better for an ordering.
Then, deg(f) would just be wdeg(LT(f))[0] / M[0][0], with the trivial case for StdDeg.

#8 - 13 Jan 2024 22:27 - John Abbott

I have re-run the example from comment 2: it now takes 0.001s.

I'm undecided about the extra generality handling the case of the first row being a multiple of (1,1,1,...,1). How often does this really occur?

It'd be nice to close this soon.

#9 - 18 Mar 2024 20:49 - John Abbott

If the grading is positive and over ZZ^1 then we use the following general cut-off so that we do not need to scan the whole polynomial.

Let the weights be w_1, w_2, ..., w_n and WLOG in decreasing order (so w_1 is biggest, and w_n is smallest).
Let wdeg of LPP be W and its stddeg be D. Then we can stop scanning terms (ordered by wdeg) once we reach a term with wdeg <= D*w_n; if at any point we find a term with higher stddeg then we repplace D but its stddeg. In particular, this general rule tells us that we can stop at the first term if all the weights are equal.

Should I implement this? Will it ever be useful? It's not that complicated, but...

ADDENDUM in practice, with any positive grading, we can stop when we have reached a term whose wdeg is <= that of z^D where z is an indet with minimal grading. *This suggests that the ring should memorize which indet is the smallest (to avoid recomputing it each time we ask for stddeg)*

#10 - 21 Mar 2024 20:40 - John Abbott

- Target version changed from CoCoALib-0.99850 to CoCoALib-0.99880

My previous comment just above is correct, but we can sometimes do better.
I think we need to consider just the "lowest dimension positive grading" (rather than the grading actually specified). Then the critical indet is the largest one having minimal grading (at the "lowest dimension" mentioned above).
It is probably a bit fiddly to determine the relevant indet in general -- easy for stddeg compatible orderings!

The code is now much improved; the generalization is probably now so urgent. So postponing this issue!