CoCoA-5 - Slug #1556

DivAlg slower than NR

22 Dec 2020 13:57 - John Abbott

Status:	New		Start date:	22 Dec 2020
Priority:	Normal		Due date:	
Assignee:	Anna Maria Bigatti		% Done:	10%
Category:	enhancing/improving		Estimated time:	0.00 hour
Target version:	CoCoA-5.4.2		Spent time:	0.55 hour

Description

Andraschko reported the following by email:
Consider the following code which computes the minimal polynomial of sqrt(2)+sqrt[3](3)+sqrt[5](5):

```
S ::= QQ[x,t1,t2,t3];
Use S;
L := [x-(t1+t2+t3), t1^2-2, t2^3-3, t3^5-5];
I := ideal(S,L);
J := elim([t1,t2,t3],I);
f := gens(J)[1]; -- some ugly poly of deg 30 that is in I
```

Now let's just compute for fun NR. This should be 0 since f is in I and L is a Gröbner basis by the coprime LT criterion.
It does and it is very fast (**not even a second**). Now if we do the same with DivAlg, it is VERY slow - although it should normally do the same and just store some information on its way.
Why is it so slow? It took **16 minutes** on my system until it was finished.

History

#1 - 22 Dec 2020 14:03 - John Abbott

I verify that the problem is present in the current version on my computer.

NR is a built-in function; **DivAlg** actually calls **ModElDivAlg** in hilop.cpkg5.

While it is reasonable that DivAlg takes longer (since it must compute the quotients too); being a factor of about 20000 slower is unreasonable!

#2 - 22 Dec 2020 14:21 - John Abbott

Just for the record: NR took about 0.063s, DivAlg took about 1280s.

Presumably DivAlg should also be built-in, and should use geobuckets to accumulate the factors?

#3 - 22 Dec 2020 17:19 - Anna Maria Bigatti

- Assignee set to Anna Maria Bigatti

Check the code for NR and see how to make the data type for the output.

#4 - 08 Jan 2021 11:20 - Anna Maria Bigatti

- % Done changed from 0 to 10

I checked: not as easy as I thought.
NR is defined in SparsePolyOps-RingElem.C and make use of ReductionCog. This means it cannot be extended to keep track of the quotients. It

has to be copied, and rewritten.