CoCoALib - Feature #40

Feature # 39 (Closed): Squarefree factorization

Squarefree factorization - Alessio d'Ali`

30 Nov 2011 16:17 - John Abbott

Status:	Closed	**Start date:**	30 Nov 2011
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	New Function	**Estimated time:**	100.00 hours
Target version:	CoCoALib-0.99531	**Spent time:**	101.60 hours

Description

Alessio d'Ali` is implementing some stages of the squarefree factorization task, under the supervision of John Abbott.

The time counted here is just that spent by d'Ali`; time spent by Abbott is counted under overhead.

Related issues:

Related to CoCoA-5 - Support #242: CoCoA-5 Projects for students (e.g. credit...	**In Progress**	**28 Sep 2012**
Related to CoCoALib - Feature #47: Squarefree factorization - multivariate po...	**Closed**	**30 Nov 2011**

History

#1 - 30 Nov 2011 16:56 - John Abbott

d'Ali` has completed a first impl of Bernardin's algorithm.
Ported to C5 by JAA. Tests taken from Bernardin's paper all pass.

#2 - 06 Dec 2011 12:49 - John Abbott

D'Ali` has almost finished refinement and testing of his impl of Bernardin's algm.
He will check the details about the assumption of primitivity -- our tests suggest that it may not be necessary for correctness (but it might be beneficial for speed of computation).

Execution times do display some unexpected variation; the source of this strange behaviour is not yet known (how does C5 compute GCDs in ZZ/(p)[x,y,z]?)

#3 - 20 Dec 2011 12:09 - John Abbott

Explained peculiar variation noted last time: simply that some of the test cases were much denser modulo certain primes than modulo other primes.

D'Ali` has also implemented the case for characteristic 0. Execution speed is competitive with the built-in "factor" command for univariate polynomials; D'Ali`'s implementation is significantly faster for multivariate inputs (in QQ[x,y]). RECALL also that the built-in factor command is working with DUPZ polynomials while D'Ali`'s code was computing with multivariate polynomials and rational coefficients!!

The case of degree 0 inputs must be handled -- a simple initial check suffices.

D'Ali` has completed an initial verification that the precondition that the input be primitive is apparently unnecessary. He will write up his analysis in LaTeX (possibly to be sent to ACM Communications in Computer Algebra?)

#4 - 20 Dec 2011 12:18 - John Abbott

JAA has been writing a short note where he uses a more flexible definition of sqfr factorization which allows any refinement of the standard sqfr factorization. D'Ali` commented that Bernardin's algorithm does sometimes multiply together factors which had been separated. Using the wider definition, these multiplications can be skipped. D'Ali` will check whether the algorithm can easily be modified so that it performs only those products strictly necessary.

D'Ali` will also modify the implementation so that input polynomials are split into primitive parts. We are now fairly sure this is not necessary for correctness, but expect that it will improve execution speed in some cases (without causing significant slowing in other cases).

#5 - 23 Dec 2011 17:02 - John Abbott

John Abbott wrote:

> D'Ali` has also implemented the case for characteristic 0. Execution speed is competitive with the built-in "factor" command for univariate polynomials; D'Ali`'s implementation is significantly faster for multivariate inputs (in QQ[x,y]).

JAA reports trouble reproducing the "comparable speed" result; the last tests indicate that the builtin "factor" command is at least 10 times faster. JAA will investigate!

#6 - 23 Dec 2011 17:17 - John Abbott

Addendum: timing varies depending on the ring, just compare the following:

In a **BIVARIATE** ring

```
Use QQ[x,y];
f := (x^2+3)^11 * (x^2+2*x+3)^5 * (x^3+3*x^2+3*x+3)^3;
t0:=CpuTime(); f1024 := f^64; PrintLn "Time to compute power: ", DecimalStr(CpuTime()-t0);

t0:=CpuTime(); Squarefree(f1024); PrintLn "Time to compute sqfr: ", DecimalStr(CpuTime()-t0);
t0:=CpuTime(); factor(f1024); PrintLn "Time to compute factor: ", DecimalStr(CpuTime()-t0);

Time to compute sqfr: 7.911
...
Time to compute factor: 3.215
```

Same example in a **UNIVARIATE** ring

```
Time to compute sqfr: 34.927
...
Time to compute factor: 3.119
```

Anna thinks strange behaviour is due to the way GCDs are computed in poly rings.

#7 - 19 Jan 2012 10:51 - John Abbott

Alessio has completed the following:

- handles case f=0 --> gives error
- handles case f=const --> appears as factor with mult=1 (should be unfactored part)
- new algm *Squarefree2* avoids multiplying together separated factors having the same multiplicity (algm is simpler, but seems to be slightly slower?!?)
- algm seems to be **unacceptably slow** over QQ -- JAA suspects that this is due to slow gcd, he will investigate.

#8 - 19 Jan 2012 10:52 - John Abbott

Next tasks:

1. investigate whether there is a "clever" way to choose the order in which the indets are to be considered
2. investigate the effect of primitivization on speed of computation (order of indets?)
3. JAA will make the C4 "fast" gcd available in CoCoALib+CoCoA5

#9 - 31 Jan 2012 12:23 - John Abbott

- % Done changed from 20 to 40

Alessio has a first prototype impl (not yet working -- JAA's fault).
He recalls the need for Coefficients(F,x).
Automatic ring homs would make the impl cleaner and easier.
JAA promises faster GCD will soon be available in C5.

Some preliminary tests have shown that a good choice of order in which to consider the indets can lead to much faster times than a poor choice (even a factor of 10 for higher degree inputs). He is looking for a guideline to help make a good choice.

#10 - 07 Feb 2012 12:18 - John Abbott

- % Done changed from 40 to 50

Alessio has implemented a function which factorizes a polynomial into its primitive parts. The idea was to use this as a cheap first step before applying the sqfr factorization algorithm, but tests showed that in CoCoA4 it takes longer to compute the primitive factors than it does to compute the sqfr factorization.

John has a first prototype of coefficients in CoCoALib; it appears to be almost 10 times as fast as the equivalent in CoCoA4. This implies that Alessio's code for computing the primitive factors might be faster than sqfr factor in CoCoA-5. We shall test this as soon as coefficients becomes available in C5.

#11 - 07 Mar 2012 13:01 - John Abbott

Alessio and John have both implemented content free factorization.
The CoCoALib impl is slightly faster -- but we ran just 1 test case.
Alessio's impl of Bernardin's algm for sqfree factorization was much faster than either of the content free impls (on the same single test case).

Alessio has a second impl of content free factorization (in primitivization3.cocoa5); it was about twice as slow as his "easy" impl.

Alessio may look at the effects of considering the indets in different orders with the hope of find a simple way of choosing the best order.

He will proceed with the LaTeX report about Bernardin's algm not requiring the input to be primitive.

He will also look over his (best) impl of Bernardin's algm with the aim of making it as clean as possible.

#12 - 19 Mar 2012 16:11 - John Abbott

Alessio has not found any criterion for a good order in which to "eliminate" the indets.

He did suggest offering an interface which allows the user to specify the order (otherwise the code makes some choice of order, probably increasing indet index).

Alessio did construct at least one family of polynomials which show that some simple criterions do not work in general.

He will write a report summarising his findings.

He also did some code cleaning. We did some more during the meeting.

#13 - 16 Apr 2012 15:50 - John Abbott

- % Done changed from 50 to 60

Alessio did numerous timing experiments to understand whether reordering the variables could lead to a useful gain in speed. His results indicated that a factor of 10 improvement is possible. **HOWEVER**, re-running some of the tests using the newly adjoined CoCoA-4 GCD code quashed the earlier result. The differences in computation time were far less (both in relative and absolute terms).

Next meeting will be in 2 weeks (30th April). Alessio will write up his proof that Bernardin's method does not require the "primitive" hypothesis. John will try to improve multivariate GCD computation.

#14 - 07 May 2012 15:46 - John Abbott

- % Done changed from 60 to 70

Alessio has started writing his report on Bernardin's article.

There are some typos in the article. JAA has given him a copy of Bernardin's thesis as it may help understand the paper.

Alessio will continue with his report.

JAA's new impl is not yet ready -- failed on some simple GCD computations.

#15 - 21 May 2012 15:59 - John Abbott

Alessio has a first solid version of his report.
We spoke of how to generate tables of timings automatically; Alessio will try to produce a CoCoA program for generating the tables.

#16 - 28 May 2012 15:54 - John Abbott

- % Done changed from 70 to 80

Alessio has written a prototype test program; he will improve it.
He has added some detail to the report.

#17 - 26 Jun 2012 15:09 - John Abbott

Alessio has:

- continued with the report
- reorganized the tests so that they are largely automatic

He will check whether *make primitive then do sqfr* is the same as applying *sqfr* directly; ideally we want a proof or some counter-examples.

Next meeting 14:30 on wed 2012-07-11 (might possibly slip one day).

#18 - 11 Jul 2012 15:49 - John Abbott

- % Done changed from 80 to 90

Alessio has added some paragraphs about primitivization, including examples to show that Bernardin's algorithm does not always produce a full primitive factorization.

We discussed the report, and made some suggestions for improving it.

Next meeting to be decided.

#19 - 31 Jul 2012 09:40 - John Abbott

Next meeting will be in September; exact day still to be decided.

#20 - 21 Sep 2012 12:34 - John Abbott

Alessio has produced two prototype implementations in CoCoA: one computes the square "in place", the other produces a new polynomial. The polynomials are actually represented as vectors of coefficients. The study is to help the revised impl of univariate polys over small finite fields (see #127).

The "in place" code is significantly more complicated, and seems to be about the same overall speed.

Testing has highlighted some curious (& undesirable) behaviour of the CoCoA interpreter, *e.g.* doubling the degree causes the computation time to increase by rather more than a factor of 4.

#21 - 21 Sep 2012 14:20 - Anna Maria Bigatti

> Testing has highlighted some curious (& undesirable) behaviour of the CoCoA interpreter, *e.g.* doubling the degree causes the computation time to increase by rather more than a factor of 4.

maybe this is related with issue "theValue makes copies"

#22 - 28 Sep 2012 12:05 - John Abbott

- Status changed from New to Resolved

- % Done changed from 90 to 100

Alessio produced a slightly improved impl for computing squares, except that it actually runs slightly more slowly than the unimproved version -- not clear why!

The squarefree report is finished.

#23 - 23 Oct 2013 20:41 - John Abbott

- Status changed from Resolved to Closed

JAA has now ported d'Ali's code into CoCoALib (and added doc & tests).

Closing this issue.

#24 - 29 Oct 2013 12:32 - Anna Maria Bigatti

- Target version set to CoCoALib-0.99531