CoCoALib - Design #620

Redesign CRTMill

11 Sep 2014 12:47 - John Abbott

Status:	In Progress	**Start date:**	11 Sep 2014
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	20%
Category:	New Function	**Estimated time:**	40.00 hours
Target version:	CoCoALib-1.0	**Spent time:**	2.25 hours

Description

I have some ideas for alternative implementations of CRTMill, but they do not fit well with the current user interface: in particular the functions residue(crt) and modulus(crt) might become quite costly (and have some "undesirable" side-effects).

Define a new user interface which allows efficient implementations of all the various CRT ideas.
Implement it (with examples, tests, etc)

Related issues:

Related to CoCoALib - Design #619: Modulus (for CRTMill) ambiguous	**Closed**	**09 Sep 2014**
Related to CoCoALib - Design #778: CRTMill::myAddInfo accept modulus 1 or not?	**In Progress**	**17 Sep 2015**

History

#1 - 11 Sep 2014 15:40 - John Abbott

- Status changed from New to In Progress

- Assignee set to John Abbott

- % Done changed from 0 to 10

I envisage two types of use of a CRTMill:

1. keep adding residue-modulus pairs until the combined modulus exceeds an *a priori* bound
2. conduct occasional checks of "stability", and stop when result is regarded as "sufficiently stable"

With use (1) it would make sense to specify the bound at the very start when creating the CRTMill; then there could be a function which says whether the bound has been reached/surpassed. This function should be **quick**, but may be **slightly imprecise**: it is acceptable for the function to say the bound has not been reached when the combined modulus is very slightly larger than the bound (thus wasting just 1 CRT iteration).

With use (2) it is necessary to compute the combined residue each time we check for stability; in any case, when stability has not been achieved, new residue-modulus pairs will be added subsequently. The details of the stability check should probably reside **outside** the CRTMill.

#2 - 11 Sep 2014 15:59 - John Abbott

I expect the main case of adding residue-modulus pairs is when both are machine long values, and we could reasonably expect that the modulus is at least 2, and that the residue is reduced. Here are two possible definitions of *reduced*:

- **weak symmetric** -m <= r <= m
- **symmetric** -(m/2) < r <= m/2

For robustness the code should accept non-reduced residues as well (which will be reduced internally). Since reducing the residue may incur a significant cost, there should be an optional flag to say that the user promises that the residue is already reduced. What should be default value for the flag be?

I also expect that normally the user knows that the modulus of a pair to be added is coprime to all previously added moduluses; since checking this can become very costly, there should be an optional flag to say that the user promises that the modulus is coprime. What should be default value for

the flag be?

#3 - 12 Sep 2014 16:55 - John Abbott

I have had a quick look at documentation for NTL and FLINT.
NTL has a very simple interface (like the existing one for CoCoALib).
FLINT has 2 interfaces: the simple one, and another where the user creates a "comb" which is then used repeatedly for many sets of residues (*e.g.* when reconstructing all the coeffs of a poly).

Think about FLINT's "comb" interface; and maybe implement something similar.

#4 - 13 Sep 2014 23:47 - John Abbott

It could be interesting to have a vector version of CRTMill which accepts vectors of residues along with a single (common) modulus. Is this any better than FLINT's comb? What about in a parallel setting?

#5 - 16 Sep 2014 13:44 - John Abbott

- % Done changed from 10 to 20

Maybe the vector version needs a little more thought.

In a deterministic algorithm, where we lift to an *a priori* bound, my proposal in note 4 should work fine. It may not work so well in a *heuristic* setting where lifting proceeds until "stability" is observed.

In the heuristic setting I envisage the following probable usage pattern. One particular coeff is used as a test case; that coeff alone is lifted until it becomes "stable", then (all) other coeffs are lifted "in parallel" and tested for stability... If some coeffs are not stable then more lifting is done.

The main question is whether a serial CRT can achieve comparable efficiency to one which knows from the start all moduli...

#6 - 11 May 2015 14:14 - John Abbott

- Target version changed from CoCoALib-0.99536 June 2015 to CoCoALib-0.99540 Feb 2016

#7 - 21 Mar 2016 14:30 - John Abbott

- Target version changed from CoCoALib-0.99540 Feb 2016 to CoCoALib-1.0

#8 - 21 Mar 2016 15:15 - John Abbott

- Related to Design #778: CRTMill::myAddInfo accept modulus 1 or not? added