CoCoALib - Design #1716

Qn: factor for BigInt

29 Nov 2022 11:23 - John Abbott

Status:	Closed	**Start date:**	29 Nov 2022
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	1.49 hour
Target version:	CoCoALib-0.99850	**Spent time:**	1.45 hour

Description

What should **factor** for BigInt do if there are large prime factors (too large to be found)?

Currently, it gives error (and in so doing discards any factors it may have found).

Would it be reasonable to return an "incomplete" factorization?
So maybe RemainingFactor could be set to the unfactored part?
I don't know if it could happen that two composite factors were found, but that they cannot be further factorized (I suppose it is possible, just rather unlikely). If this happens then we would have two "unfactored" parts...

My current application could use a small prime factor if one can be found quickly/easily. I prefer not to use PollardRho directly because that stops as soon as any factor is found, whereas factor does try to recurse.

Comments? Ideas? Would it be OK to return composite factors? (maybe they could be marked with minus signs?)

Related issues:

Related to CoCoALib - Design #950: factor and SmoothFactor for integers --> F...	**Closed**	**20 Oct 2016**

History

#1 - 29 Nov 2022 21:14 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

I had thought about identifying non-prime factors by making them negative.
But a colleague said he found that "not very transparent"... though it might well prompt the user to look at the manual.

An even simpler option would simply be to state in the doc that the factors may be composite. Though I suppose a user might find it "unpleasantly surprising" if some factors are composite, when smaller/easier inputs always give prime factors.

#2 - 30 Nov 2022 17:08 - John Abbott

- Assignee set to John Abbott

I have checked the code (& updated it).
It seems that the PollardRho loop can anyway output composite factors, though it does attempt to factorize them.
I have revised the manual entry to help make this clearer.

I have now modified the code so that a large composite factor after the "TrialDiv" loop which cannot be split by the PollardRho loop, is placed in the RemainingFactor field.

#3 - 30 Nov 2022 17:13 - John Abbott

Should we permit the user to impose a time limit on **factor**/**FactorINT**?
Is this sensible? Doubling the amount of time increases the range of detectable factors only slightly -- presumably up to about 4 times value (**not** 4 times log).

I was also wondering whether the automatic PollardRho limit, should depend on the size of the input.
I did a quick experiment, and found that the time taken is roughly linear in FloorLog2(R).
A tolerable limit (on my computer) was about 10^11/FloorLog2(R), which corresponded to a max time of about 60 seconds.

#4 - 13 Mar 2023 22:02 - John Abbott

- Related to Design #950: factor and SmoothFactor for integers --> FactorINT, FactorINT_TrialDiv, FactorINT_PollardRho added

#5 - 13 Mar 2023 22:03 - John Abbott

- % Done changed from 10 to 60

It is now possible to impose a timeout.

#6 - 14 Mar 2023 21:32 - John Abbott

- Status changed from In Progress to Closed

- % Done changed from 60 to 100

- Estimated time set to 1.49 h

The current state is reasonable. Some improvements could be made, but I'm not sure it is worth investing time in this "peripheral" aspect of CoCoALib. What we have now works well enough for number which are not too big (and whose prime factors are smaller than about 10^12).

Closing.