# CoCoALib - Design #950

## factor and SmoothFactor for integers --> FactorINT, FactorINT_TrialDiv, FactorINT_PollardRho

20 Oct 2016 20:10 - John Abbott

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 20 Oct 2016 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | John Abbott | **% Done:** | 100% |
| **Category:** | Safety | **Estimated time:** | 0.00 hour |
| **Target version:** | CoCoALib-0.99850 | **Spent time:** | 9.10 hours |

**Description**

I can make factor and SmoothFactor for integers faster in some cases by calling IsProbPrime.
The risk is that a factor returned as irreducible (prime) is not; this risk is very low.

Is this acceptable?
Should the user be allowed to stop these "risky" short-cuts?

Discuss.

**2022-02** Now called FactorINT, FactorINT_TrialDiv and FactorINT_PollardRho

**Related issues:**

| | | |
|---|---|---|
| Related to CoCoALib - Feature #796: CoCoALib function for radical (or SqFree)... | **Closed** | **05 Nov 2015** |
| Related to CoCoALib - Design #1159: Add global enum "verify/DontVerify" | **Closed** | **22 Feb 2018** |
| Related to CoCoALib - Slug #1170: SmoothFactor: slow when a factor is found | **Closed** | **28 Mar 2018** |
| Related to CoCoALib - Design #1716: Qn: factor for BigInt | **Closed** | **29 Nov 2022** |

### History

#### #1 - 20 Oct 2016 20:10 - John Abbott

*- Related to Feature #796: CoCoALib function for radical (or SqFree) of a polynomial added*

#### #2 - 21 Oct 2016 07:46 - Anna Maria Bigatti

ProbSmoothFactor?
or SmoothFactor(n) and SmoothFactor(n, "NoProb")?

#### #3 - 21 Oct 2016 14:14 - John Abbott

*- Status changed from New to In Progress*

*- % Done changed from 0 to 10*

Does that mean that you are in favour of offering the user the choice between a fast (and probably correct) answer, or a potentially (very) slow but guaranteed correct answer?

What should the default choice be? Perhaps there should be a global flag (set in the GlobalManager?) which says what the default behaviour should be?

#### #4 - 06 Nov 2017 14:03 - John Abbott

*- Target version changed from CoCoALib-0.99560 to CoCoALib-0.99600*

#### #5 - 09 Mar 2018 18:16 - John Abbott

*- Related to Design #1159: Add global enum "verify/DontVerify" added*

#### #6 - 09 Mar 2018 18:18 - John Abbott

*- Description updated*

Maybe there could be a fn called **IsPrime3** which is guaranteed (reasonably) fast, and returns a bool3?

#### #7 - 28 Mar 2018 20:25 - John Abbott

*- Related to Slug #1170: SmoothFactor: slow when a factor is found added*

#### #8 - 12 Jun 2018 16:42 - John Abbott

*- Target version changed from CoCoALib-0.99600 to CoCoALib-0.99650 November 2019*

#### #9 - 08 Feb 2019 21:05 - John Abbott

*- Target version changed from CoCoALib-0.99650 November 2019 to CoCoALib-1.0*

#### #10 - 14 Mar 2019 13:35 - John Abbott

Here is an effect of the idea of testing the remaining factor for primality.

```
>>> N := NextPrime(2^60);
>>> SmoothFactor(N, 1100000000);
record[RemainingFactor := 1152921504606847009, factors := [], multiplicities := []]
--->  time about 0.01s

>>> Pmax := 1100000000;
>>> p1 := NextPrime(Pmax);
>>> p2 := NextPrime(p1);
>>> NN := p1*p2;
>>> SmoothFactor(NN, 1100000000);
record[RemainingFactor := 1210000028600000153, factors := [], multiplicities := []]
time is about 4.6s
```

There is a big difference in speed depending on whether the remaining factor is prime or not... This is a natural consequence, but is it desirable?

#### #11 - 14 Oct 2020 13:26 - John Abbott

*- % Done changed from 10 to 20*

I have just checked what the code currently does: if many primes have been tried without finding a factor **and** if the remaining is small (less than 2^64) then IsPrime is called, otherwise no primality test is applied.

One reason for this strategy is for instance trying SmoothFactor(1+factorial(10000), 10000).
Applying IsProbPrime to 1+factorial(100000) takes about 84s on my computer; simply trying all primes up to 10000 takes about 0.02s. So even 1 call to IsProbPrime is highly disadvantageous. See also #1170.

This is not really a main feature of CoCoALib; it would be better to delegate factorization to some external library (which?)

What to do?

#### #12 - 20 Oct 2021 15:56 - John Abbott

Robbiano would like factor to work for integers (or some integers, at least).
How can we do this reasonably? Maybe it should simply call SmoothFactor with some predefined bound? Perhaps also call PollardRho?

#### #13 - 10 Jan 2022 17:15 - John Abbott

*- Assignee set to John Abbott*

I have now a prototype fn for finding **one** factor using PollardRhoSeq.
It takes as input N>1 to be factorized, and an upper bound on the number of iterations.
If no factor is found, the return value is 0 (which is obviously impossible).

An interesting test case is N := 4087 which requires PollardRho to try a=2 and a=3 (where the poly is x^2+a).
[I did not find any requiring a>3; but did not look that hard].
Also N=13861

An example with 3 factors: N=28214231

#### #14 - 20 Jan 2022 20:05 - John Abbott

*- % Done changed from 20 to 30*

Currently my prototype function is called **factor_PollardRho**.
It takes 2 args: N to be factorized (non-zero integer),
iters number of iterations [time is roughly linear in number of iters]

Should there be a variant with timeout? Maybe that makes sense only in CoCoALib,
and if there is a structure which can resume computing from where timeout occurred?

If we want an easy-to-use factorization function for integers, it should probably
do just a few iterations of SmoothFactor (upto 1000? Surely not more) and then
switch to PollardRho. I do not really want to invest time and effort into making
a more sophisticated integer factorizer.

#### #15 - 21 Jan 2022 12:12 - John Abbott

Anna thinks a name closer to SmoothFactor would be more appropriate because the fns both take 2 args...
No final decision.

#### #16 - 21 Jan 2022 12:14 - Anna Maria Bigatti

Drastic change: FactorBound, FactorIter or FactorINTBound, FactorINTIter or FactorINT_bound, FactorINT_iter?

#### #17 - 26 Jan 2022 12:46 - John Abbott

*- Target version changed from CoCoALib-1.0 to CoCoALib-0.99850*

Another proposal:

- **FactorINT(N,TimeOut)** tries both TrialDiv and PollardRho; maybe also does IsProbPrime check?
- **FactorINT_TrialDiv(N,pmax)** trial division upto pmax
- **FactorINT_PollardRho(N, iters)** PollardRho method with upto iters iterations (gives 0 if no factor found)

Opinions?

#### #18 - 27 Jan 2022 12:29 - Anna Maria Bigatti

John Abbott wrote:

> - **FactorINT_TrialDiv(N,pmax)** trial division upto pmax
> Opinions?

maybe just **FactorINT_div(N,pmax)** or **FactorINT_smooth(N,pmax)** (to recall its previus name)?

#### #19 - 27 Jan 2022 14:24 - John Abbott

I suggest making SmoothFactor obsolescent, and just forward the call to FactorINT_TrialDiv.

I think I prefer FactorINT_TrialDiv to FactorINT_smooth because the first name is clearer to everyone, also FactorINT_PollardRho indicates the method used so it would be more coherent to use the name FactorINT_TrialDiv since that also identifies the method used.

I do wonder whether there would be any sense in letting the user specify a TimeOut (instead of some other upper limit). Or perhaps both sorts of limit at the same time?

#### #20 - 31 Jan 2022 22:12 - John Abbott

*- % Done changed from 30 to 60*

I have now implemented **FactorINT**, **FactorINT_TrialDiv** and **FactorINT_PollardRho**.
SmoothFactor is now obsolete.
Doc has been updated. Tests and examples have been updated (but no new tests added).

#### #21 - 01 Feb 2022 08:38 - Anna Maria Bigatti

*- Subject changed from factor and SmoothFactor for integers to factor and SmoothFactor for integers --> FactorINT, FactorINT_TrialDiv, FactorINT_PollardRho*

*- Description updated*

**#22 - 02 Feb 2022 11:16 - John Abbott**

Oh joy!

Have fun if you decide to document functions whose name contain an underscore: LaTeX gives its usual cryptic, unhelpful error messages (sigh!)

I have reworded the doc to avoid the underscores.

Does anyone object to having fn names containing underscores?
If so, what do you propose as an alternative?

**#23 - 13 Mar 2023 22:02 - John Abbott**

*- Status changed from In Progress to Closed*

*- % Done changed from 60 to 100*

The current impl is good enough for most purposes.
Some factors may be reducible if PollardRho is used.
Too bad.
Closing.

**#24 - 13 Mar 2023 22:02 - John Abbott**

*- Related to Design #1716: Qn: factor for BigInt added*