How can I improve this Pollard's rho algorithm to handle products of semi-large primes?

Below is my implementation of Pollard's rho algorithm for prime factorization:

```
#include <vector>
#include <queue>

#include <gmpxx.h>

// Interface to the GMP random number functions.
gmp_randclass rng(gmp_randinit_default);

// Returns a divisor of N using Pollard's rho method.
mpz_class getDivisor(const mpz_class &N)
{
    mpz_class c = rng.get_z_range(N);
    mpz_class x = rng.get_z_range(N);
    mpz_class y = x;
    mpz_class d = 1;
    mpz_class z;

    while (d == 1) {
        x = (x*x + c) % N;
        y = (y*y + c) % N;
        y = (y*y + c) % N;
        z = x - y;
        mpz_gcd(d.get_mpz_t(), z.get_mpz_t(), N.get_mpz_t());
    }

    return d;
}

// Adds the prime factors of N to the given vector.
void factor(const mpz_class &N, std::vector<mpz_class> &factors)
{
    std::queue<mpz_class> to_factor;
    to_factor.push(N);

    while (!to_factor.empty()) {
```

It's essentially a straight translation of the pseudocode on Wikipedia, and relies on GMP for big numbers and for primality testing. The implementation works well and can factor primes such as

```
1000036317378699858851366323 = 1000014599 * 1000003357 * 1000018361
```

but will choke on e.g.

```
1000000000000023221400000000048599822299 = 1000000000002301019 * 1000000000000021121
```

My question is: Is there anything I can do to improve on this, short of switching to a more complex factorization algorithm such as Quadratic sieve?

I know one improvement could be to first do some trial divisions by pre-computed primes, but that would not help for products of a few large primes such as the above.

I'm interested in any tips on improvements to the basic Pollard's rho method to get it to handle larger composites of only a few prime factors. Of course if you find any stupidities in the code above, I'm

interested in those as well.

For full disclosure: This is a homework assignment, so general tips and pointers are better than fully coded solutions. With this very simple approach I already get a passing grade on the assignment, but would of course like to improve.

Thanks in advance.

c++ algorithm prime-factoring

asked Nov 2 '13 at 12:37
estan
162 1 9

The wikipedia-article seem to hint that changing your `f(x)` will change it's behaviour. Maybe that could help you progress faster on larger primes? – PureW Nov 2 '13 at 14:51

Your specific example of 1000000000000023221400000000048599822299 = 1000000000002301019 * 1000000000000021121 should be easily handled by Fermat factorization, since the difference of the factors is only 2279936. – Ilmari Karonen Nov 2 '13 at 18:44

add comment

1 Answer

You are using the original version of the rho algorithm due to Pollard. Brent's variant makes two improvements: Floyd's tortoise-and-hare cycle-finding algorithm is replaced by a cycle-finding algorithm developed by Brent, and the gcd calculation is delayed so it is performed only once every hundred or so times through the loop instead of every time. But those changes only get a small improvement, maybe 25% or so, and won't allow you to factor the large numbers you are talking about. Thus, you will need a better algorithm: SQUFOF might work for semi-primes the size that you mention, or you could implement quadratic sieve or the elliptic curve method. I have discussion and implementation of all those algorithms at my blog.

answered Nov 2 '13 at 15:49
user448810
6,144 2 5 18

add comment

Not the answer you're looking for? Browse other questions tagged c++ algorithm prime-factoring or ask your own question.