

## uRe: implementing the quadratic sieve

**Source:** <http://coding.derkeiler.com/Archive/General/comp.programming/2004-11/0110.html>

---

**From:** Michael Jørgensen (*ingen\_at\_ukendt.dk*)

**Date:** 11/02/04

Date: Tue, 2 Nov 2004 15:40:17 +0100

"Marlene Stebbins" <marlene@mail.com> wrote in message  
news:1fxhd.90671\$Pl.86607@pd7tw1no...

- > *I am trying to write a toy implementation of the quadratic sieve.*
- > *The following has been gleaned from several sources. If any of you*
- > *are familiar with the quadratic sieve, I would appreciate you looking*
- > *this over and telling me if I've got it right. FYI, I am a 9th grader*
- > *with a very limited math background.*

Hi Marlene,

That sounds exciting! I've just done the same thing myself with a slightly different variant of the same algorithm. Let me try to give you some hints/comments:

- > *Quadratic Sieve factoring algorithm:*
- >
- > *to factor n:*
- >
- > *1. Select a set of prime numbers called the factor base.*
- > *(Selecting the factor base requires further explanation.)*

Easy, just select the smallest prime numbers, for instance all primes less than 100 (that gives 25 primes).

- > *2. For each integer f close to the square root of n,*
- > *try to factor  $(f^2 - n) \pmod n$  with the primes in the factor base.*
- > *("Close to" is defined as an arbitrary range.)*

You really don't need f to be close to the square root of n. This requirement is really only required for optimizing the algorithm.

The real time crunching happens when you want to factor the number  $g = (f^2 - n) \pmod n$  [which is the same as  $f^2 \pmod n$ ], because this value is of the same size as n. When f is close to  $\sqrt{n}$  then g becomes "small". However, if you're working with numbers less than ten (or so) digits, this won't matter.

## comp.programming: uRe: implementing the quadratic sieve

The algorithm works by assuming that all \*factors\* of  $g$  are small, i.e. that  $g$  may be completely factored over the factor base. If  $g$  does not completely factor over the factor base, then it is simply discarded. The smaller  $g$  is, the more likely all the prime factors are small too.

However, it does not take long to factor  $g$ , and you simply keep trying a new pair  $(f,g)$  until you have enough.

- > 3. If any or all of the primes in the factor base are factors
- > of  $(f^2) - n \pmod n$ , save  $f$ .

Note, you must keep track of the sign.  $96^2 \pmod{4633}$  is  $-50$  not  $50$ .

- > 4. Choose several of the saved  $f$ 's such that when their squares
- > are multiplied together, the prime factors of the product  $\pmod n$
- > all have even exponents.

This is another tricky part. So, for each pair  $(f,g)$  you must keep a record of the exponents of the primes in the factor base. In fact, you only need to keep a list of the primes that have odd exponents. The trick is to write  $g = p_1 * p_2 * p_3 * \dots * q^2 \pmod n$ , where  $p_1, \dots$  are the primes with odd exponent, and  $q^2$  is the rest. You should be able to figure out  $q$  from the factorization of  $g$ .

For instance:  $85^2 \pmod{4633} = 2^5 * 3^4 = 2 * 36^2$ . So for  $f=85$  and  $n=4633$  we have  $p_1=2$  and  $q = 36$ .

- > 5. Call the product of the  $f$ 's  $x$ .
- >
- > 6. Call the square root of the product of the prime factors  $y$ .
- >
- > 7. Then,  $\gcd(x+y, n)$  and  $\gcd(x-y, n)$  are factors of  $n$ .

This all looks very good.

>  
>

-----  
---  
>  
> Example:  
>  
> to factor  $n=4633$ :  
>  
> 1. factor base = {2, 3, 5}  
>  
> 2.  $\sqrt{4633} = 68.xxx$ , so  $f$ 's "close to" 68 are, say, 38 to 98  
>  
> 3. For  $f$ 's  $((38, 98)^2) - 4633 \pmod n$ , the following have 2, 3 or 5  
> as prime factors: 59, 67, 68, 69, 85, 96  
No. For the numbers 59, 67, 68, 96 the value of  $f^2 - n \pmod n$  is negative.  
When considering modular arithmetic in general, then  $13 = -2 \pmod{15}$ . So a small negative number is "the same" as a much larger number. However, when

## comp.programming: uRe: implementing the quadratic sieve

factoring you may not neglect the negative sign. You have two possible approaches:

1) Convert all your small negative numbers into the corresponding positive (and larger) numbers; by adding  $n$ . Then factor these positive numbers. This is the simplest. It is not the best performance, because you suddenly have to factor a large number instead of a small number, but if you're only working with small numbers, i.e. less than say ten digits, then execution speed is probably not your top priority.

2) The alternative is to remember the negative sign whenever it appears. This can be achieved by inserting an extra number "-1" into your prime factor base. If this is not entirely obvious, just forget about it and go back to option (1) above.

Let me know if you have any further questions. Do come back here again and share your experiences: What was easy, what was difficult, what worked, what didn't work, etc.

-Michael.

```
>
> 4.  $(68 * 69 * 96)^2 \pmod n = 2^8 * 3^2 * 5^2$  (the exponents are all even).
>
> 5.  $x = \text{sqrt}((68 * 69 * 96)^2 \pmod n)$ ;  $y = \text{sqrt}(2^8 * 3^2 * 5^2)$ 
>
> 6.  $x = (68 * 69 * 96) = 450432$ ;  $y = ((2^4) * 3 * 5) = 240$ 
>
> 7.  $\text{gcd}(450432+240, 4633) = 41$ ;  $\text{gcd}(450432-240, 4633) = 113$ 
>
> 8.  $4633 = 41 * 113$ 
```