

Re: Basically a sieve method, relation to quantum

Source: <http://sci.tech-archive.net/Archive/sci.math/2005-01/6926.html>

From: David C. Ullrich (ullrich_at_math.okstate.edu)

Date: 01/24/05

Date: Mon, 24 Jan 2005 05:54:35 -0600

On Sun, 23 Jan 2005 23:49:58 -0500, "Tim Peters" <tim.one@comcast.net> wrote:

>[jstevh@msn.com]

>>>>

>>>> To factor an RSA challenge number I'd need a full factorization of some
>>>> number off of it. Now $T = M^2 - j^2$, where M is the target number, and
>>>> j is a number you get to pick.

>

>[Tim Peters]

>>> That's where you often get off track, failing to engage your full
>>> abilities. What you want instead is a method that will factor M given a
>>> factorization of $T = M + j$, where j is a number you get to pick. For
>>> example, pick $j=0$, and then your algorithm will run in linear time.

>

>[David C. Ullrich]

>> Damm. I was going to suggest deriving a factorization of M from a
>> factorization of $2*M$, but your idea is much simpler and more elegant.

>

>Indeed, and yours is just the special case of picking $j=M$ (I know James
>realizes that, I'm just pointing it out for your benefit). What he may not
>have realized yet is that his method is just a special case of picking $j =$
> $M^2 - j^2 - M$.

>

>I wouldn't call it my idea, though -- it's almost certain I wouldn't have
>thought of it without JSH pushing in this direction. Since I'm too stupid
>to make it work anyway, James should get all the credit.

>

>Right now he's got this niggling detail where, to factor M in polynomial
>time, he at least has to assume that factoring integers with twice as many
>digits is cheap.

I just realized it's simply a proof by induction.

Of course it's "upwards" induction instead of downwards, so lesser minds might be concerned about the lack of a "base case". But at some point you run out of storage, and the recursion ends there. (I point this out just to prove that mathematicians

can so solve programming problems, so there.)

*>Some posters are making a big deal out of that, as if it
>were a fundamental problem. Really! It would be easier to prove that M can
>be factored in polynomial time if he got to assume instead that factoring
> $M+0$ takes negligible time. Even the chowderheads on sci.math could
>understand that proof. Heck, even I can write a linear-time Python program
>for it:
>
>def factor(M):
> j = 0
> factors = factorize(M+j) # returns list of factors in negligible time
> print [hex(p) for p in factors]*

I'm `_still_` using 1.5.2, believe it or not – yes I know 2.x is much better, really hate to install new software, 1.5.2 works just fine for the silly little things I use it for, etc. Never mind.

Anyway, I've seen enough discussion to recognize that list comprehension thing, if that's the right term for it. But I hadn't realized there was a "factorize" in 2.x. This may be enough to push me over the edge. As it were.

*>The only subtle part is using hex() to display the factors (Python's hex(n)
>takes time linear in the number of bits in n; converting to decimal instead
>would inject a quadratic-time component).*

That comment is really out of place in this thread, don't you think?

*>> Oh well, not the first time something like this has happened – I guess
>> we math guys should leave the actual programming to the pros.
>
>Ya, you're way out of your depth here again. I am too. We should charge
>admission for visiting this end of the pool.
>*

David C. Ullrich