# An idea for making the $\mathbb{F}_2[x]$ approach viable

September 3, 2009

## 1 Introduction

Recall that the top level in our iteration to produce the generating function in $\mathbb{F}_2[x]$ for the primes up to $n$, amounts to computing

$$f(x) := \sum_{d \le \sqrt{n}} \frac{x^{da_d} - 1}{x^d - 1},$$

which can clearly be evaluated mod any polynomial $g(x) \in \mathbb{F}_2[x]$ in time at most

$$\deg(g) n^{1/2+o(1)}.$$

Moreover, compositions $f(f_1(f_2 \cdots (f_k(x)) \cdots))$ can be quickly evaluated, at least if the $f_i$ are lacunary.

In this note, we consider only the case where $k = 1$ and $f_1(x) = x^t$, for various small values of $t$, where we take $g(x)$ to be a fixed, irreducible polynomial of degree about $(\log n)^2$, say, and where the order of $x$ mod this polynomial is huge, say larger than $n^2$.

This sounds like a rather different sort of approach from what I had written in previous polymath postings, where we test divisibility by many different polynomials $g(x)$; but in fact it is in the same spirit: observe that if $f(x^t)$ is divisible by $g(x)$, it means that the $t$th powers of the roots of $g(x)$ are roots of $f(x)$. If we vary over $t$, and had that $f(x^t)$ is always divisible by $g(x)$ (for those $t$ under consideration), it would mean that $f(x)$ is divisible by lots of low-degree polynomials. What fixing $g(x)$ as the modulus does for us, then, is it saves us the trouble of having to do Chinese Remaindering and working with different moduli, among other things.

## 2 The idea

What I think is perhaps the case is that we cannot have that

$$f(1), f(x), f(x^2), ..., f(x^N) \equiv 0 \pmod{2, g(x)} \tag{1}$$

for $N \sim \sqrt{n}$, say. My reasoning is based on the fact that the same thing cannot hold if $f(x)$ were a polynomial with only $N$ or so terms.

Specifically, suppose that

$$f_2(x) := \sum_{d \leq \sqrt{n}} x^{da_d},$$

where the $a_d$ satisfy

$$0 < a_d < n/d,$$

and all the $da_d$ are distinct (in the rational function analogue, even if the $da_d$ were not distinct, you still would have that the rational functions $(x^{da_d} - 1)/(x^d - 1)$ are all distinct). Then, if

$$f_2(1), f_2(x), ..., f_x(x^N) \equiv 0 \pmod{2, g(x)},$$

we would have that

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x^{a_1} & x^{2a_2} & \cdots & x^{Na_N} \\ x^{2a_1} & x^{4a_2} & \cdots & x^{2Na_N} \\ \vdots & \vdots & \ddots & \vdots \\ x^{(N-1)a_1} & x^{2(N-1)a_2} & \cdots & x^{N(N-1)a_N}. \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \pmod{2, g(x)}.$$

But this is impossible, since the matrix is invertible, and has determinant

$$\prod_{1 \leq i < j \leq N} (x^{ia_i} - x^{ja_j}),$$

which is not divisible by $g(x)$, because we chose $g(x)$ so that $x$ has large order mod $g(x)$.

Let's say we could indeed prove that (1) cannot hold. And let's further assume that we could quickly evaluate $f(x^t) \pmod{2, g(x)}$ for $t$ up to $N$,

by perhaps generalizing the Fast Fourier Transform (to work with sums of sparse rational functions), so that everything takes time at most

$$\deg(g)N^{1+o(1)}.$$

Then, this leads to an algorithm to locate primes of size $n$, that only takes $n^{1/2+o(1)}$ computations. So far that is no better than Odlyzko's method; **but**, the $\mathbb{F}_2[x]$ approach might (in fact, *should*) allow us to port over our results that allow us to compute the parity of $\pi(n)$ quickly.

Ok, how would even the $n^{1/2+o(1)}$ algorithm work? The idea would be to apply the above to the generating function $F(x)$ for the primes in $[n, 2n]$, locating an integer $1 \leq t \leq n^{1/2}$ or so, such that

$$F(x^t) \not\equiv 0 \pmod{2, g(x)}.$$

(i.e. $g(x)$ does not divide $F(x^t)$.) Then, writing

$$F(x^t) = F_l(x^t) + F_h(x^t),$$

where $F_l(x)$ is the generating function for the primes in $[n, 3n/2]$, and $F_h(x)$ is the generating function for the primes in $(3n/2, 2n]$, we have that either $F_l(x^t)$ or $F_h(x^t)$ fails to be divisible by $g(x)$. So, we can use this as a basis for iteration, just like in the parity of $\pi(n)$ algorithm; and note that we don't have to re-compute our value for $t$ each time, nor do we have to switch to a different $g(x)$.

# 3 A plan

In order for all this to work, to produce, say, an algorithm that runs in time $n^{5/11+o(1)}$ or better to locate primes, the following three goals must be met:

**Goal 1.** Show that not all of

$$f(1),\ f(x),\ f(x^2),\ ...,\ f(x^N)$$

can be $0 \pmod{2, g(x)}$. Perhaps this has already been worked out in the literature; or, perhaps there is a way to modify the Vandermonde determinant approach somehow. Or perhaps it is false; if so, surely there is a simple modification that makes it true.

**Goal 2.** Generalize FFT's, or perhaps produce some other algorithm, to quickly evaluate $f(1), f(x), ..., f(x^N) \pmod{2, g(x)}$.

**Goal 3.** Find a way to port over the ideas leading to the parity-of-$\pi(n)$ algorithm to the $\mathbb{F}_2[x]$ context.