Bas Edixhoven, Jean-Marc Couveignes and Robin de Jong have proved that $\tau(p)$ can be computed in polynomial time; their approach involves sophisticated techniques from arithmetic geometry (e.g., étale cohomology, motives, Arakelov theory). *This is work in progress and has not been written up in detail yet.* The ideas they use are inspired by the ones introduced by Schoof, Elkies and Atkin for quickly counting points on elliptic curves over finite fields (see [Sch95]).

Edixhoven describes the strategy as follows:

1. We compute the mod ℓ Galois representation ρ associated to Δ. In particular, we produce a polynomial f such that $\mathbb{Q}[x]/(f)$ is the fixed field of $\ker(\rho)$. This is then used to obtain $\tau(p) \pmod{\ell}$ and do a Schoof-like algorithm for computing $\tau(p)$.

2. We compute the field of definition of suitable points of order ℓ on the modular Jacobian $J_1(\ell)$ to do part 1. (This modular Jacobian is the Jacobian of a model of $\Gamma_1(\ell)\backslash\mathfrak{h}^*$ over $\mathbb{Q}$.)

3. The method is to approximate the polynomial f in some sense (e.g., over the complex numbers, or modulo many small primes r), and use an estimate from Arakelov theory to determine a precision that will suffice.

2.7 Fast Computation of Bernoulli Numbers

This section[1] is about the computation of the Bernoulli numbers B_n, for $n \geq 0$, defined in Section 2.1.2 by

$$\frac{x}{e^x - 1} = \sum_{n=0}^{\infty} B_n \frac{x^n}{n!}. \tag{2.7.1}$$

One way to compute B_n is to multiply both sides of (2.7.1) by $e^x - 1$ and equate coefficients of x^{n+1} to obtain the recurrence

$$B_0 = 1, \qquad B_n = -\frac{1}{n+1} \cdot \sum_{k=0}^{n-1} \binom{n+1}{k} B_k$$

This recurrence provides a straightforward and easy-to-implement method for calculating B_n, if one is interested in computing B_n for all n up to some bound. For example,

$$B_1 = -\frac{1}{2} \cdot \left(\binom{2}{0} B_0 \right) = -\frac{1}{2},$$

and

$$B_2 = -\frac{1}{3} \cdot \left(\binom{3}{0} B_0 + \binom{3}{1} B_1 \right) = -\frac{1}{3} \cdot \left(1 - \frac{3}{2} \right) = \frac{1}{6}.$$

[1]This section represents joint work with Kevin McGown.

However, computing B_n via the recurrence is slow; it requires us to sum over many large terms, it requires storing the numbers $B_0, \dots, B_{n-1}$ in memory, and it takes only limited advantage of asymptotically fast arithmetic algorithms.

A second approach to computing B_n is to take advantage of Newton iteration and asymptotically fast polynomial arithmetic to compute $1/(e^x - 1)$. See [**?**] [Buhler et al.] for extensive details on applications of this method modulo a prime p.

A third way to compute B_n is to use Proposition 2.1.6. E.g., one can use the resulting algorithm paper to compute the rational number B_{10^5} (which has over 370000 digits) in a few minutes using the implementation in [BCea]. Much of what we will describe was gleaned from the PARI-2.2.11 source code, which computes Bernoulli numbers using an algorithm based on (2.1.6). This algorithm appears to have been independently invented by several people: by Bernd C. Kellner (see `www.bernoulli.org`); by Bill Dayl; and by H. Cohen and K. Belabas.

The Riemann zeta function has a product representation

$$\zeta(s) = \sum_{m=1}^{\infty} m^{-s} = \prod_{p \text{ prime}} (1 - p^{-s})^{-1}.$$

We compute B_n as an exact rational number by approximating $\zeta(n)$ to very high precision using the Euler product, using (2.1.6), and using the following theorem:

Theorem 2.7.1 (Clausen, von Staudt)**.** *For even $n \geq 2$,*

$$\mathrm{denom}(B_n) = \prod_{p-1 \mid n} p.$$

Remark 2.7.2. The Sloane sequence A103233 is the number of digits of the numerator of B_{10^n}. The following is a new quick way to compute the number of digits of the numerator of B_n. By Theorem 2.7.1 we have $d_n = \mathrm{denom}(B_n) = \prod_{p-1|n} p$. The number of digits of numerator is thus

$$\lceil \log_{10}(d_n \cdot |B_n|) \rceil$$

But

$$\begin{aligned} \log(|B_n|) &= \log\left(\frac{2 \cdot n!}{(2\pi)^n}\,\zeta(n)\right) \\ &= \log(2) + \log(n!) - n\log(2) - n\log(\pi) + \log(\zeta(n)), \end{aligned}$$

and $\zeta(n) \sim 1$ so $\log(\zeta(n)) \sim 0$. Finally, Stirling's formula gives a fast way to compute $\log(n!) = \log(\Gamma(n+1))$:

$$\log(\Gamma(z)) = \frac{1}{\log(2\pi)} + \left(z - \frac{1}{2}\right)\log(z) - z + \sum_{m=1}^{\infty} \frac{B_{2m}}{2m(2m-1)z^{2m-1}}.$$

Using this method we can compute the number of digits of $B_{10^{50}}$ in a second.

We return the problem of efficiently computing B_n. Let

$$K = \frac{2 \cdot n!}{(2\pi)^n}$$

so that $|B_n| = K\zeta(n)$. Write

$$B_n = \frac{a}{d},$$

with $a, d \in \mathbb{Z}$, $d \geq 1$, and $\gcd(a, d) = 1$. It is elementary to show that $a = (-1)^{n/2+1}\ |a|$ for even $n \geq 2$. Suppose that using the Euler product we approximate $\zeta(n)$ from below by a number z such that

$$0 \leq \zeta(m) - z < \frac{1}{Kd}.$$

Then $0 \leq |B_n| - zK < d^{-1}$, hence $0 \leq |a| - zKd < 1$. It follows that $|a| = \lceil zKd \rceil$ and hence $a = (-1)^{n/2+1}\ \lceil zKd \rceil$.

It remains to compute z. Consider the following problem: given $s > 1$ and $\varepsilon > 0$, find $M \in \mathbb{Z}_+$ so that

$$z = \prod_{p \leq M} (1 - p^{-s})^{-1},$$

satisfies $0 \leq \zeta(s) - z < \varepsilon$. We always have $0 \leq \zeta(s) - z$. Also,

$$\sum_{n \leq M} n^{-s} \leq \prod_{p \leq M} (1 - p^{-s})^{-1}$$

so

$$\zeta(s) - z \leq \sum_{n=M+1}^{\infty} n^{-s} \leq \int_M^{\infty} x^{-s}\, dx = \frac{1}{(s-1)M^{s-1}}.$$

Thus if $M > \varepsilon^{-1/(s-1)}$, then

$$\frac{1}{(s-1)M^{s-1}} \leq \frac{1}{M^{s-1}} < \varepsilon,$$

so $\zeta(s) - z < \varepsilon$, as required. For our purposes, we have $s = n$ and $\varepsilon = (Kd)^{-1}$, so it suffices to take $M > (Kd)^{1/(n-1)}$.

Algorithm 2.7.3 (Compute Bernoulli number B_n)**.** *Given an integer $n \geq 0$ this algorithm computes the Bernoulli number B_n as an exact rational number.*

1. [Special cases] If $n = 0$ return 1, if $n = 1$ return $-1/2$, and if $n \geq 3$ is odd return 0.
2. [Factorial factor] Compute $K = \dfrac{2 \cdot n!}{(2\pi)^n}$ to sufficiently many digits of precision so that ceiling in step 6 is uniquely determined (this precision can be determined using Remark 2.7.2).

3. [Denominator] Compute $d = \prod_{p-1\mid n} p$
4. [Bound] Compute $M = \left\lceil (Kd)^{1/(n-1)} \right\rceil$
5. [Approximate $\zeta(n)$] Compute $z = \prod_{p \le M} (1 - p^{-n})^{-1}$
6. [Numerator] Compute $a = (-1)^{n/2+1} \lceil dKz \rceil$
7. [Output B_n] Return $\dfrac{a}{d}$.

In step 5 use a Sieve to compute all primes $p \le M$ efficiently. In step 4 we may replace M by any integer greater than the one specified by the formula, so we do not have to compute $(Kd)^{1/(n-1)}$ to very high precision.

Example 2.7.4. We illustrate Algorithm 2.7.3 by computing B_{50}. Using 135 binary digits of precision, we compute

$$K = 7500866746076957704747736.71552473164563479$$

The divisors of n are $1, 2, 5, 10, 25, 50$, so

$$d = 2 \cdot 3 \cdot 11 = 66.$$

We find $M = 4$ and compute

$$z = 1.00000000000000088817842109308159029835012$$

Finally we compute

$$dKz = 495057205241079648212477524.999999994425778,$$

so

$$B_{50} = \frac{495057205241079648212477525}{66}.$$

Remark 2.7.5. A time-consuming step in Algorithm 2.7.3 is computation of $n!$, though this step does not dominate the runtime. See [] [[fast factorial web page]] for a discussion of several algorithms.