18.783 Elliptic Curves **Spring 2013**

Problem Set #3 **Due: 02/28/2013**

Description

These problems are related to the material covered in Lectures 5-6. I have made every effort to proof-read the problems, but there may well be errors that I have missed. The first person to spot each error will receive 1-3 points of extra credit on their problem set, depending on the severity of the error.

Instructions: Solve Problem 1 and **one** of Problems 2 and 3. Then complete Problem 4, which is a survey.

All of the problems on this problem set involve at least some programming. While the algorithms involved are straight-forward and you only have two problems to do, I recommend starting early just in case it takes more time to debug your code than you expect (experienced programmers will know that this is good advice; it is even better advice for inexperienced programmers).

Problem 1. Root-finding over $\mathbb{Z}$ (50 points)

In this problem you will develop an algorithm to find integer roots of polynomials in $\mathbb{Z}[x]$ using a p-adic version of Newton's method (also known as Hensel lifting). As an application, this gives us an efficient way to factor perfect powers (a special case that we will need to handle when we come to the elliptic curve factorization method), and it will be needed in Problem 2 to find integer roots of division polynomials.

In the questions below, p can be any integer greater than 1, but you may assume it is a prime power if you wish.

1. Let $x_0 \in \mathbb{Z}$ and $f \in \mathbb{Z}[x]$. Prove that the following equivalence holds in $\mathbb{Z}[x]$:

$$f(x) \equiv f(x_0) + f'(x_0)(x - x_0) \bmod (x - x_0)^2.$$

2. Let $x_0, z_0 \in \mathbb{Z}$ and $f \in \mathbb{Z}[x]$ satisfy $f(x_0) \equiv 0 \bmod p$ and $f'(x_0)z_0 \equiv 1 \bmod p$. Let

$$x_1 \equiv x_0 - f(x_0)z_0 \bmod p^2,$$
$$z_1 \equiv 2z_0 - f'(x_1)z_0^2 \bmod p^2.$$

Prove that that the following three equivalences hold:

$$x_1 \equiv x_0 \bmod p, \tag{i}$$
$$f(x_1) \equiv 0 \bmod p^2, \tag{ii}$$
$$f'(x_1)z_1 \equiv 1 \bmod p^2. \tag{iii}$$

Show that (i) and (ii) characterize $x_1 \bmod p^2$ uniquely by proving that if $x_2 \in \mathbb{Z}$ also satisfies $x_2 \equiv x_0 \bmod p$ and $f(x_2) = 0 \bmod p^2$, then $x_1 \equiv x_2 \bmod p^2$.

Iteratively applying part 2 yields an algorithm that, given an integer k and x_0, z_0, and f satisfying the hypothesis of part 2, outputs an integer x_k that satisfies $f(x_k) \equiv 0 \bmod p^{2^k}$.

3. Prove that if f has an integer root r for which $f'(r)$ is invertible modulo p, then given $x_0 \equiv r \bmod p$, $z_0 \equiv 1/f'(x_0) \bmod p$, and k such that $|r| < p^{2^k}/2$, this algorithm outputs x_k such that r is the unique integer $r \equiv x_k \bmod p^{2^k}$ satisfying $|r| < p^{2^k}/2$.

To apply the result of part 3, we need to know a suitable starting value (or values) for x_0. For the two applications we have in mind, this is will be straightforward, so let us proceed on the assumption that we are given a suitable x_0 with $f'(x_0)$ invertible modulo p.

Let B be the maximum of the absolute values of the coefficients of f, and let B_0 be an upper bound on the absolute value of its largest integer root. It suffices to choose the least k such that $p^{2^k} > 2B_0$, and since any integer root of f must divide its constant coefficient, we can assume that $B_0 \le B$.

4. Prove that with this choice of k the algorithm can be implemented to run in time $O(d\ \mathsf{M}(\log B))$, where d is the degree of f. Prove that if f has $O(1)$ terms, then the algorithm can be implemented to run in time $O(\mathsf{M}(\log B) + \mathsf{M}(\log B_0)\log d)$.

5. Using the primes $p = 2$ and $p = 3$, describe an efficient algorithm that, given an integer N relatively prime to 6, either outputs an integer a and a prime q such that $a^q = N$, or proves that N is not a perfect power. Prove that your algorithm runs in time $O(n^2 \log\log n)$, where $n = \log N$.[1]

6. Implement your algorithm and report the time it takes when run on the each of the following five inputs: $2^{1000} + 297$, 5^{503}, $(2^{500} + 55)^2$, $(2^{333} + 285)^3$, and $(2^{32} + 15)^{31}$.

Problem 2. The torsion subgroup of $E(\mathbb{Q})$ (50 points)

Let E be an elliptic curve over $\mathbb{Q}$. The problem of determining the set of rational points on E is a famously hard problem that is still unsolved. However, determining the rational points of finite order is easy. In this problem you will develop the first part of an efficient algorithm for doing so.

We shall assume that E is defined by a Weierstrass equation $y^2 = x^3 + Ax + B$, where A and B are *integers*. This assumption is not restrictive: we can always pick $u \in \mathbb{Z}$ so that the isomorphic curve $y^2 = x^3 + u^4Ax + u^6B$ has integer coefficients.

Let $P = (x_1, y_1)$ be a point of finite order $m > 0$ in $E(\mathbb{Q})$. Our first goal is to prove that P must have integer coordinates. This was proved independently first by Nagell [3] and then by Lutz [2] in the 1930's and is the first half of the Nagell-Lutz Theorem. The standard proof [4, §8.1] relies on a p-adic filtration.

Here you will give a shorter and simpler proof that relies only on properties of the division polynomials. As shown in lecture, for any positive integer n, the x-coordinate x_n of the point nP is given by $x_n = \phi_n(x_1)/\psi_n^2(x_1)$ where

$$\phi_n(x) = x^{n^2} + \cdots,$$
$$\psi_n^2(x) = n^2x^{n^2-1} + \cdots,$$

with each ellipsis denoting lower order terms; see Problem 3 for the full definition of ϕ_n and ψ_n, which depend on the curve coefficients A and B.

[1]In fact, this problem can be solved in quasi-linear time; see [1].

1. Prove that if x_n is an integer, then x_1 must be an integer. Conclude that if P does not have integer coordinates, then for any prime p dividing m, the point $(m/p)P$ does not have integer coordinates, and thus we may assume that m is prime.

2. Prove that if $m = 2$ then P has integer coordinates.

3. Let m be an odd prime. Then x_1 is a root of the polynomial $\psi_m(x) = mx^{(m^2-1)/2} + \cdots$, which has integer coefficients. Using this and the fact that (x_1, y_1) satisfies the curve equation, prove that x_1 (and therefore y_1) is an integer, so P has integer coordinates.

We now need a few facts about the image of the torsion subgroup under reduction modulo a prime p of good reduction for E. So let $\Delta(E) = -16(4A^3 + 27B^2)$ be the discriminant of E, and let p be a prime that does not divide Δ. Abusing notation, let us write $E(\mathbb{F}_p)$ to denote the group of $\mathbb{F}_p$-rational points on the reduction of E modulo p.

4. Prove that if $P \in E(\mathbb{Q})$ has order m, then its reduction in $E(\mathbb{F}_p)$ also has order m. Moreover, show that the reduction map from $E(\mathbb{Q})$ to $E(\mathbb{F}_p)$ is injective at torsion points.

5. Let N be the last four digits of your student ID. Then set $A = a(N)$ and $B = b(N)$, where $a(t)$ and $b(t)$ are the polynomials[2]

$$a(t) = -27(t^4 - 12t^3 + 14t^2 + 12t + 1),$$
$$b(t) = 54(t^6 - 18t^5 + 75t^4 + 75t^2 + 18t + 1),$$

and let E be the elliptic curve $y^2 = x^3 + Ax + B$ over $\mathbb{Q}$. Let p be the least prime of good reduction for E such that 25 does not divide $\#E(\mathbb{F}_p)$. Use the root-finding algorithm from Problem 1 to lift a point $P$of order 5 from $E(\mathbb{F}_p)$ to $E(\mathbb{Q})$; that is, compute the pre-image of $P \in E(\mathbb{F}_p)$ under the reduction map from $E(\mathbb{Q})$ to $E(\mathbb{F}_p)$.

Problem 3. Computing division polynomials (50 points)

For integers $n \geq 0$, define $\psi_n \in \mathbb{Z}[x, y, A, B]$ by

$$\begin{aligned}
\psi_0 &= 0,\\
\psi_1 &= 1,\\
\psi_2 &= 2y,\\
\psi_3 &= 3x^4 + 6Ax^2 + 12Bx - A^2,\\
\psi_4 &= 4y(x^6 + 5Ax^4 + 20Bx^3 - 5A^2x^2 - 4ABx - 8B^2 - A^3),\\
\psi_{2m} &= \frac{1}{2y}\psi_m(\psi_{m+2}\psi_{m-1}^2 - \psi_{m-2}\psi_{m+1}^2) \qquad (m \geq 3),\\
\psi_{2m+1} &= \psi_{m+2}\psi_m^3 - \psi_{m-1}\psi_{m+1}^3 \qquad (m \geq 2).
\end{aligned}$$

Let $\phi_1 = x$ and $\omega_1 = y$, and for integers $n > 1$ define

$$\phi_m = x\psi_m^2 - \psi_{m+1}\psi_{m-1},$$
$$\omega_m = \frac{1}{4y}(\psi_{m+2}\psi_{m-1}^2 - \psi_{m-2}\psi_{m+1}^2).$$

[2] This parameterization is due to David Zywina.

It is a straight-forward exercise (which you are not required to do) to show that these polynomials have the form

$$\begin{aligned}
\phi_n(x) &= x^{n^2} + \cdots, \\
\omega_n(x,y) &= \begin{cases} y(x^{3(n^2-1)/2} + \cdots) & n \text{ odd}, \\ x^{3n^2/2} + \cdots & n \text{ even}, \end{cases} \\
\psi_n(x,y) &= \begin{cases} nx^{(n^2-1)/2} + \cdots & n \text{ odd}, \\ y(nx^{(n^2-4)/2} + \cdots) & n \text{ even}, \end{cases}
\end{aligned}$$

where each ellipsis denotes terms of lower degree in x.

In practical applications it is more convenient to work with the univariate polynomials

$$f_n(x) = \begin{cases} \psi_n & n \text{ odd}, \\ \psi_n/\psi_2 & n \text{ even}. \end{cases}$$

Note that $\psi_2 = 2y$, and it follows from the formulas above that f_n does not depend on y. If $P = (x_0, y_0)$ is a point on the elliptic curve $y^2 = x^3 + Ax + B$ with $y_0 \neq 0$ (so P is not a 2-torsion point), then $f_n(x_0) = 0$ if and only if $nP = 0$. In this problem you will develop an efficient algorithm to compute f_n.

1. Let $F(x) = 4(x^3 + Ax + B)$. Using the recursion formulas for ψ_{2m} and ψ_{2m+1}, derive recursion fomulas for f_{2m} and f_{2m+1} that involve $f_{m-2}, \ldots, f_{m+2}$ and F. Note that for f_{2m+1} you will need to distinguish the cases where m is odd and even.

2. Show that for any $k \geq 3$, if you are given the polynomials $f_{k-3}, \ldots, f_{k+5}$ and F, you can compute the polynomials $f_{2k-3}, \ldots, f_{2k+5}$ (call this *doubling*), and you can also compute the polynomials $f_{2(k+1)-3}, \ldots, f_{2(k+1)+5}$ (call this *doubling-and-adding*).

3. Implement an algorithm that, given a positive integer n, a prime p, and coefficients A and B, computes the division polynomial $f_n \in \mathbb{F}_p[x]$ for the elliptic curve $E/\mathbb{F}_p$ defined by $y^2 = x^3 + Ax + B$, using a left-to-right binary exponentiation approach. Here are a few tips, but you are free to use any design you like.

   - Work in the polynomial ring $\mathbb{F}_p[x]$, which you can create in Sage by typing `R.<x>=PolynomialRing(GF(p))`. Note that A and B are now scalars in $\mathbb{F}_p$, not variables. Precompute $F = 4(x^3 + Ax + B) \in \mathbb{F}_p[x]$.
   - You need an initial vector of division polynomials $v = [f_{k-3}, \ldots, f_{k+5}]$ to get started. If the leading two bits of n are "11", then let $v = [f_0, \ldots, f_8]$ and $k = 3$. Otherwise, let $[f_1, \ldots, f_9]$ and $k = 4$ if the top three bits of n are "100", and let $v = [f_2, \ldots, f_{10}]$ and $k = 5$ if the top three bits of n are "101".
   - Implement a function that, given k, $v = [f_{k-3}, \ldots, f_{k+5}]$, F, and a bit b, computes $k' = 2k + b$ and $v = [f_{k'-3}, \ldots, f_{k'+5}]$. To perform left-to-right binary exponentiation, call this function repeatedly, passing in the bits of n starting from either 2 or 3 bits from in the top and working down to the low order bit.
   - To test your code, you can compare results with Sage, which already knows how to compute f_n, via

```
FF=GF(p); R.<x>=PolynomialRing(FF)
E=EllipticCurve([FF(A),FF(B)])
E.division_polynomial(n,x,0)
```

- Your program should be quite fast, but be careful not to test it with values of n that are too large — the degree of f_n is quadratic in n, so if n is, say, a million, you would need terabytes of memory to store f_n (which you do not have!).

4. Analyze the asymptotic complexity of your program as a function of $\log p$ and n.

5. Modify your program so that it performs its computations modulo x^7 (to compute $f(x) \bmod x^7$ in Sage use `f.mod(x^7))`. Now let A be the least prime greater than the last two digits of your student ID, let B be the least prime greater than the first two digits of your student ID, and let $p = 65537$. Let $E/\mathbb{F}_p$ be the elliptic curve defined by $y^2 = x^3 + Ax + B$, and let $n = N^{100} + 1$, where N is the integer formed by adding the last three digits of your student ID to 9000.

a. Use your modified program to compute $f_n \bmod x^7$ and record the result in your problem set. Be sure to first test your program with smaller values of n and verify the results with Sage (your answer to this question will be heavily weighted when grading this problem, so please be careful).

b. Time your program using the `time` command in Sage. How long does it take?

Problem 4. Survey

Complete the following survey by rating each of the problems you attempted on a scale of 1 to 10 according to how interesting you found the problem (1 = "mind-numbing," 10 = "mind-blowing"), and how difficult you found the problem (1 = "trivial," 10 = "brutal"). Also estimate the amount of time you spent on each problem to the nearest half hour.

	Interest	Difficulty	Time Spent
Problem 1			
Problem 2			
Problem 3			

Also, please rate each of the following lectures that you attended, according to the quality of the material (1="useless", 10="fascinating"), the quality of the presentation (1="epic fail", 10="perfection"), and the novelty of the material (1="old hat", 10="all new").

Date	Lecture Topic	Material	Presentation	Novelty
2/21	Isogenies and endomorphisms			
2/26	Division polynomials and torsion points			

Feel free to record any additional comments you have on the problem sets or lectures.

References

[1] D. J. Bernstein, *Detecting perfect powers in essentially linear time*, Mathematics of Computation **67** (1998), 1252–1283.

[2] E. Lutz, *Sur l'equation* $y^2 = x^3 - ax - b$ *dans les corps* p*-adic*, J. Reine Angew. Math. **177** (1937), 237–247.

[3] T. Nagell, *Solution de quelque problèmes dans la théorie arithmétique des cubiques planes du premier genre*, Wid. Akad. Skrifter Oslo I **1** (1935).

[4] L. Washington, *Elliptic curves: Number theory and cryptography*, second edtion, CRC press, 2008.