18.783 Elliptic Curves **Spring 2015**
Problem Set #3 **Due: 02/27/2015**

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 just **one** of the first three problems. Then complete Problem 4, which is a survey. When you submit your solutions via e-mail, please give the attached file a name of the form `LastnamePset3.pdf`.

I recommend reading through all three problems before you pick your poison. My take on the problems:

- Problem 1 involves a happy mixture of theory, coding, and complexity analysis. The coding should be easy, but the complexity analysis will require some care. If you have already taken 18.782, please do not choose this problem.
- Problem 2 requires no coding but will involve a longer write up than the other two problems. It is probably a bit more difficult conceptually. If you solve this problem you will have proved a result that does not seem to appear in the literature.
- Problem 3 requires more coding than the others, but should be the easiest to write up (and it has some neat practical applications that we may see later).

Of course your mileage may vary. Whichever problem you choose, I would urge you not to wait until the last minute to start it (none of these problems is particularly hard, but they are probably best solved in more than one sitting).

Problem 1. Root-finding over $\mathbb{Z}$ (100 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 used as a black box 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.

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

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

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

(c) 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 in (c), 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 $B_0 \leq B$. We can also assume $p < 2B_0$, since otherwise the problem is trivial ($k = 0$ and $x_k = x_0$).

(d) 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 (be careful here, the most obvious implementation will not achieve this time bound). Prove that if f has $O(1)$ terms, then the algorithm can be implemented to run in time $O(\mathsf{M}(\log B) + \mathsf{M}(\log B_0)\log d)$.

(e) 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 quasi-quadratic time (meaning $O(n^2(\log n)^e)$ for some e, where $n = \log N$).

(f) Implement your algorithm and report the result and running time on the each of the following inputs: $2^{1000}+297$, 5^{503}, $(2^{500}+55)^2$, $(2^{333}+285)^3$, $(2^{32}+15)^{31}$, $100!+1$. To time your code in sage, use the `time` command (e.g. `time 1+1`).

(g) Prove that the algorithm you gave in (e) can be implemented to run in sub-quadratic time (meaning $o(n^2)$ where $n = \log N$). You may need to modify the algorithm slightly in order to achieve this.[1]

Problem 2. The torsion subgroup of $E(\mathbb{Q})$ (100 points)

Let E be an elliptic curve over $\mathbb{Q}$. The problem of determining the rational points on E is a famously hard problem that is still unsolved. However, determining the rational

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

points of finite order is easy. In this problem you will design (but need not implement) 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 [5] and then by Lutz [4] in the 1930's and is the first half of the Nagell-Lutz Theorem. The standard proof [6, §8.1] relies on a p-adic filtration, but in this problem you will give a shorter and simpler proof that relies only on properties of the division polynomials. As shown in lecture, for any integer n not divisible by m, the x-coordinate x_n of the point $nP = (x_n, y_n)$ is given by $x_n = \phi_n(x_1)/\psi_n^2(x_1)$ where

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

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.

(a) Prove that for any positive integer $n < m$, if x_n is an integer, then x_1 must be an integer. Use this to reduce to the case that m is prime.

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

(c) If m is an odd prime then x_1 is a root of $\psi_m(x) = mx^{(m^2-1)/2} + \cdots \in \mathbb{Z}[x]$. Using this, prove that x_1 is an integer, and then show that y_1 must also be an integer (thus P has integer coordinates as claimed).

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. Reducing the coefficients A and B modulo p then gives an elliptic curve $E_p/\mathbb{F}_p$. Since we know that torsion points in $E(\mathbb{Q})$ have integer coordinates, we can always reduce the coordinates of such a point modulo p to get the coordinates of a point in $E_p(\mathbb{F}_p)$.

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

We now recall Mazur's theorem from Lecture 1, which tells us that the order of a torsion point in $E(\mathbb{Q})$ can be at most 12 (and cannot be 11). Our strategy is to pick a prime $p \geq 11$ of good reduction for E, find all the points of order less than or equal to 12 in $E_p(\mathbb{F}_p)$, and use the algorithm from Problem 1 to try and lift these points to $E(\mathbb{Q})$ (you don't need to implement or prove anything about the algorithm in Problem 1, for the purposes of this problem we will take it as given).

The first step is to find a prime p that does not divide the discriminant Δ. Doing this by trial division is not fast enough to give a quasi-linear running time, so we need to be a bit more clever. We will instead use an algorithm for fast simultaneous modular reduction [3, Alg. 10.16]. to compute $\Delta \bmod p_i$ for the first several primes $p_1, \cdots, p_k$ greater than 11, where k is chosen so that $M = p_1 \cdots p_k > \Delta$ (so we know that $\Delta \bmod p_i$ is nonzero for some p_i, we'll just pick the least one).

This is accomplished using a *product tree*, a binary tree of integers whose bottom level (the leaves of the tree) consists of the primes p_i; for the sake of simplicity let us assume we round k up to a power of 2 so that we have a complete binary tree. Working our way up from the leaves, we set the value of each internal node to the product of its children; eventually we reach the root of the tree, which then has the value $M = p_1 \cdots p_k$. We then replace the root M with $d = |\Delta| \bmod M$, and for each of its children m_1 and m_2 we replace m_i with $d_i = d \bmod m_i$ (which is $|\Delta| \bmod m_i$). Recursively working our way down the tree, we eventually get $|\Delta| \bmod p_i$ in the leaves.

In order to bound the complexity of our algorithm, we define

$$n := \lg |A| + \lg |B|,$$

which represents the bit-size of the input, the elliptic curve $E/\mathbb{Q}$ given as $y^2 = x^3 + Ax + B$ with $A, B \in \mathbb{Z}$. Note that we then also have $\log |\Delta| = O(n)$.

(e) Prove that we can determine the least prime $p \geq 11$ that does not divide Δ in time $O(\mathsf{M}(n) \log n)$, and use the Prime Number Theorem to show that $p = O(n)$.

For each integer $m > 1$, define the polynomial $f_m \in \mathbb{Z}[x]$ as follows:

$$f_m(x) = \begin{cases} x^3 + Ax + B & \text{if } m = 2, \\ \psi_m / \psi_2 & \text{if } m > 2 \text{ is even}, \\ \psi_m & \text{if } m \text{ is odd}, \end{cases}$$

where ψ_m denotes the mth division polynomial of the elliptic curve $E\colon y^2 = x^3 + Ax + B$.

(f) Prove that if $P = (x_1, y_1) \in E(\mathbb{Q})$ has finite order m not divisible by p then we have $f_m(x_1) = 0 \bmod p$ and $f'_m(x_1) \neq 0 \bmod p$.

It follows that their exist suitable starting values x_0 and z_0 to which we can apply the algorithm in Problem 1 to obtain an integer root of $f_m(x)$ that is congruent to x_0 modulo p. By part (c), this root must be equal to x_1. This still leaves the question of how to find such an x_0. We know it must appear as the x-coordinate of some point in $E_p(\mathbb{F}_p)$ of order m, so it suffices to find all such points for all the values of $m \leq 12$ permitted by Mazur's theorem.

(g) Give an algorithm to enumerate all the points $(x_0, y_0) \in E_p(\mathbb{F}_p)$ in time $O(n\mathsf{M}(\log n))$.

(h) Give an algorithm to construct the set S consisting of all points in $E_p(\mathbb{F}_p)$ of order at most 12 in time $O(n\mathsf{M}(\log n))$, and prove that the cardinality of S is $O(1)$ (meaning it is bounded by a constant that does not depend on n).

(i) Prove that there is a bound $H > 0$ with $\log H = O(n)$ such that the coefficients of f_m all have absolute value bounded by H, for $2 \leq m \leq 12$. You don't need to give an explicit value for H, just show that it exists and can be effectively computed.

(j) Using the $O(d\mathsf{M}(\log H))$ complexity bound of the root-finding algorithm from Problem 1 (proved in part (d) of Problem 1), show that for any point $Q \in S$ of order m you can either find a point $P \in E(\mathbb{Q})$ of order m that reduces to Q modulo p, or prove that no such P exists[2] in time $O(\mathsf{M}(n))$.

[2]Note that not every point $Q \in S$ is necessarily the reduction of a point $P \in E(\mathbb{Q})$.

(k) Conclude that you can enumerate the torsion points in $E(\mathbb{Q})$ in $O(\mathsf{M}(n)\log n)$ time.

It is worth noting that the algorithm you have just designed is asymptotically faster than both of the algorithms given in [6]: one is based on the the Lutz–Nagell Theorem [6, Thm. 8.7], which requires factoring Δ and is not polynomial time, and the other uses Doud's algorithm [2] which is quasi-quadratic but not quasi-linear.[3]

Problem 3. Computing division polynomials (100 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

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

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.

[3] Doud gives a quasi-cubic complexity bound in [2] but with fast arithmetic it is quasi-quadratic.

(a) Let $F(x) = 4(x^3 + Ax + B)$. Using the recursion formulas for ψ_{2m} and ψ_{2m+1}, derive recursion formulas 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.

(b) 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*).

(c) 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 several terabytes of memory to store f_n.

(d) Analyze the asymptotic complexity (in terms of time and space) of your program as a function of $\log p$ and n. Use $\mathsf{M}(b)$ to denote the time to multiply two b-bit integers.

(e) 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.

(i) 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).

(ii) 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 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 presentation (1="epic fail", 10="perfection"), the pace (1="way too slow", 10="way too fast"), and the novelty of the material (1="old hat", 10="all new").

Date	Lecture Topic	Material	Presentation	Pace	Novelty
2/19	Isogenies and endomorphisms				
2/24	Division polynomials				

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] D. Doud, *A procedure to calculate torsion of elliptic curves over* $\mathbb{Q}$, Manuscripta Mathematica **95** (1998), 463–469.

[3] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*, third edition, Cambridge University Press, 2013.

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

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

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

MIT OpenCourseWare
http://ocw.mit.edu

18.783 Elliptic Curves
Spring 2015

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.