18.783 Elliptic Curves **Spring 2013**
Problem Set #5 **Due: 03/19/2013**

Description

These problems are related to the material covered in Lectures 9-10. As usual, the first person to spot each non-trivial typo/error will receive one point of extra credit.

Instructions: Solve **one** of Problems 1-2 and one of Problems 3-4. Then do Problem 5, which is a survey.

Problem 1. Rubik's pie (60 points)

In this problem you will write an optimal solver for the Rubik's pie puzzle that you received in class. The Rubik's pie consists of 18 pieces: 8 corner pieces, 8 edge pieces, and 2 center pieces. Every piece of the puzzle can be uniquely identified by specifying its shape (corner, edge, center), and the color of the stickers on it; for example, there is exactly one white-blue corner piece. The puzzle has 6 faces. The two circular faces, which we will call the *front* and *back* faces, each contain 9 pieces: 4 corners, 4 edges, and a center piece. The four side faces, which we will call the *up* ,*down*, *right*, and *left* faces, each contain 6 pieces: 4 corners and 2 edges. Note that each side face intersects each circular face in 3 pieces (2 corners and an edge), and adjacent side faces intersect in 2 corner pieces.

There are seven permissible *moves*, all of which are performed while looking directly at the front face (this determines the meaning of "clockwise" – you are assumed to be looking at the front face while turning the back face).

1. A clockwise quarter-turn of the back face (b), which moves the back pieces on the right face to the down face.

2. A counter-clockwise quarter-turn of the back face (f), which moves the back pieces on the right face to the up face.

3. A half-turn of the back face (B).

4. Half-turns of any of the four side faces (u, d, r, l).

All moves keep the center front piece in a fixed orientation and each has a unique inverse. We do not include rotations of the front face; up to orientation, these are equivalent to a rotation of the back face (the counter-clockwise quarter-turn of the back face is labelled f because it is equivalent to a clockwise quarter-turn of the front face).

We define a *solved* puzzle to be one in which the front face is white and the colors of the stickers running clockwise along the side of the front layer starting from the blue corner are blue, blue, orange, orange, red, red, green, green, and the colors of the stickers along the side of the back layer match those in the front. Figure 1 shows a solved puzzle.[1]

[1] Unfortunately the manufacturer did not deliver all of the puzzles in a solved state, according to our definition (the order of the colors may vary). You can deal with this either by taking your puzzle apart (rotate an side face 45 degrees and gently force a corner out) and reassembling it, or by using your solver to put the puzzle in a solved state before answering the questions that follow.

Figure 1. Two views of a solved puzzle.

A *configuration* of the puzzle specifies the location of each edge and corner piece in a fixed orientation with the white center piece in front; for example, a particular configuration might have the yellow-red edge is in the down-back position. There are four possible configurations of a solved puzzle. We say that a solved puzzle is in the *standard configuration* if the white-blue corner is in the up-left-front position (which also means that the yellow-red edge is in the down-back position).

Below are four views of a puzzle that had the move sequence **`frBul`** applied to a solved puzzle in the standard configuration. The view on the far left has the same orientation as the standard configuration. You may wish to verify this with your own puzzle.

Figure 2. Views of a solved puzzle in the standard configuration after applying **`frBul`**.

Your task is to implement an *optimal solver* (also known as "God's algorithm") for the Rubik's pie. This is an algorithm that, given any starting configuration, outputs a shortest sequence of moves that leads to a solved configuration (not necessarily the standard one).[2]

Consider the graph $G = (V, E)$ whose vertex set V consists of all possible configurations of the puzzle and whose edges (v_1, v_2) are labelled with one of the seven permissible moves $m \in \{u, d, r, l, b, B, f\}$, where applying the move m to configuration v_1 yields the configuration v_2. This is a bi-directed graph in which each vertex has degree 7; the move labelling the edge (v_1, v_2) is the inverse of the move labelling the edge (v_2, v_1).

Given two vertices s and t in this graph, we wish to find a (not necessarily unique) shortest path from s to t. The edge labels on this path give us a sequence of moves $w = m_1 m_2 \cdots m_k$ that will take the puzzle from configuration s to configuration t in k moves, where k is the distance from s to t. Reversing the path and inverting each move yields a sequence $w^{-1} = m_k^{-1} m_{k-1}^{-1} \cdots m_1^{-1}$ that takes the puzzle from t to s.

To find such a path you will use a *bidirectional search*. Let $N(s, r)$ denote the *r-neighborhood* of s, the set of vertices v whose distance from s is at most r. For each vertex in $v \in N(s, r)$ we include a path w from s to v of length r (it does not matter which path is chosen) that we store together with v, so we view $N(s, r)$ as a set of pairs (v, w) where the v's are all distinct, and we index this set by v (in Python this can be conveniently implemented using a dictionary).

The bi-directional search algorithm works by alternately expanding neighborhoods of s and t until they intersect. An outline of the algorithm is given below. We use ϵ to denote the empty path.

[2]Unlike a Rubik's cube, with the Rubik's Pie, every starting configuration can be solved; this means that if your puzzle falls apart it does not matter how you put it back together.

1. Set $N(s,0) = \{(s,\epsilon)\}$ and $N(t,0) = \{(t,\epsilon)\}$, and set $r_s = r_t = 0$.
2. Repeat until $N(s,r_s)$ and $N(t,r_t)$ contain a common vertex v:
   a. If $r_s = r_t$, compute $N(s,r_s+1)$ by extending $N(s,r_s)$ and then increment r_s; Otherwise, compute $N(t,r_t+1)$ by extending $N(t,r_t)$ and then increment r_t.
3. Output the path $w_1 w_2^{-1}$, where $(v,w_1) \in N(s,r_s)$ and $(v,w_2) \in N(t,r_t)$.

Note that when extending a neighborhood you may encounter the same vertex multiple times, but you should only keep one pair (v,w) for each v (alternatively you could keep them all and compute every shortest path from s to t). Once you have implemented and tested your algorithm, use it to answer the following questions:

1. Find an optimal solution to the configuration obtained by applying the move sequence

`blurdbrBrdflblfrBrBrBdrbub`

to the standard puzzle. Below are four views of a standard puzzle after applying this move sequence. The view on the left has the same orientation as the standard configuration.

Figure 3. Standard puzzle after applying **`blurdbrBrdflblfrBrBrBdrbub`**.

Your solution should be a shortest move sequence that, when applied to the puzzle pictured on the left, yields a solved puzzle. Equivalently, it should be a shortest inverse of the move sequence above. Also record how long it took your algorithm to find a solution.

2. Next, generate a "random" sequence of moves m using the Python code snippet

```
m = ''.join(['fbBudlr'[d] for d in (N^10).digits(7)])
```

where N is the first four digits of your student ID. Find a shortest inverse to m.

3. By linearly extrapolating from the time it took your program to solve part 1, give a rough estimate (to within an order of magnitude) of the time it would take your program to find an optimal solution to the puzzle in Figure 3 if you had instead used a breadth-first search rather than a bidirectional search (i.e. just expand a neighborhood of s until it contains t). Assume that memory is not a limiting factor.

Problem 2. The image of Galois (60 points)

Let $E/\mathbb{Q}$ be an elliptic curve, let ℓ be a prime, and let $K = \mathbb{Q}(E[\ell])$ be the Galois extension of $\mathbb{Q}$ obtained by adjoining the coordinates of all the points in the ℓ-torsion subgroup $E[\ell]$ to $\mathbb{Q}$. The Galois group $\mathrm{Gal}(K/\mathbb{Q})$ acts linearly on the vector space

$$E[\ell] \;\simeq\; \mathbb{Z}/\ell\mathbb{Z} \oplus \mathbb{Z}/\ell\mathbb{Z} \;\simeq\; \mathbb{F}_\ell^2,$$

thus there is a group homomorphism

$$\rho_E : \mathrm{Gal}(K/\mathbb{Q}) \to \mathrm{GL}_2(\mathbb{F}_\ell)$$

that maps each field automorphism $\sigma \in \mathrm{Gal}(K/\mathbb{Q})$ to an element of the general linear group $\mathrm{GL}_2(\mathbb{F}_\ell)$, which we may view as an invertible 2×2 matrix with coefficients in $\mathbb{F}_\ell$ (after choosing a basis for $E[\ell]$).

As you may recall, a homomorphism from a group G to a group of linear transformations is called a (linear) *representation* of G. The map ρ_E is a representation of the group $\mathrm{Gal}(K/\mathbb{Q})$, known as the *mod-$\ell$ Galois representation* attached to E.[3]

For each prime $p \neq \ell$ where E has good reduction there is a *Frobenius element* Frob_p of $\mathrm{Gal}(K/\mathbb{Q})$, which reduces to the Frobenius map $x \mapsto x^p$ modulo a prime of K lying above p. Let E_p denote the reduction of E modulo such a prime p. The Frobenius element is mapped by ρ_E to an element of $\mathrm{GL}_2(\mathbb{F}_\ell)$ corresponding to π_ℓ, the restriction of the Frobenius endomorphism of E_p to the ℓ-torsion subgroup $E_p[\ell]$. The *Frobenius element* Frob_p is only determined up to conjugacy (and is usually identified with its conjugacy class), since it depends on a choice of basis, but we can unambiguously determine the characteristic polynomial of $\rho_E(\mathrm{Frob}_p) = \pi_\ell$. In particular, the trace of $\rho_E(\mathrm{Frob}_p)$ is the trace of Frobenius $t = p + 1 - \#E_p(\mathbb{F}_p)$ modulo ℓ, and the determinant of $\rho_E(\mathrm{Frob}_p)$ is simply $p \bmod \ell$ (note that $p \neq \ell$).

The Chebotarev density theorem tells us that for any conjugacy class C of $\mathrm{Gal}(K/\mathbb{Q})$, the proportion of primes p for which Frob_p lies in C is exactly the ratio $\#C/\#\mathrm{Gal}(K/\mathbb{Q})$. Asymptotically, we can think of each prime p as being assigned a uniformly random Frobenius element $\mathrm{Frob}_p \in \mathrm{Gal}(K/\mathbb{Q})$ which is mapped by ρ_E to a uniformly random element of the image of ρ_E in $\mathrm{GL}_2(\mathbb{F}_\ell)$. For a typical elliptic curve $E/\mathbb{Q}$, the representation ρ_E is surjective and its image is all of $\mathrm{GL}_2(\mathbb{F}_\ell)$, but this is not always the case. Number theorists (and others) are very interested in understanding these exceptional cases. The image of ρ_E has a direct impact on the statistical behavior of $E_p[\ell]$ as p varies. For instance, the proportion of primes p for which $E_p[\ell] = E_p(\mathbb{F}_p)[\ell]$ is precisely $1/\#\operatorname{im}\rho_E$, since this occurs if and only if $\rho_E(\mathrm{Frob}_p) = \pi_\ell$ is the identity.

The purpose of this exercise is for you to attempt to determine the image of ρ_E for various elliptic curves $E/\mathbb{Q}$ by analyzing the statistics of π_ℓ as $p \neq \ell$ varies over primes of good reduction, by comparing these statistics to the corresponding statistics for various candidate subgroups of $\mathrm{GL}_2(\mathbb{F}_\ell)$. Not every subgroup of $\mathrm{GL}_2(\mathbb{F}_\ell)$ can arise as the image of ρ_E, since, for example, $\operatorname{im}\rho_E$ must contain matrices with ever possible nonzero determinant (as p varies, $\det\pi_\ell$ will eventually hit every element of $\mathbb{F}_\ell^*$).

For $\ell = 3$ there are, up to conjugacy, 8 candidate subgroups G of $\mathrm{GL}_2(\mathbb{F}_\ell)$ for the image of ρ_E. These are listed in Table 1, and can also be found in the Sage worksheet 18.783 Problem Set 5 Problem 2.sws.

[3] Typically K is replaced by the algebraic closure of $\mathbb{Q}$, but for our purposes we just need $E[\ell](K) = E[\ell]$.

group	order	description	generators
C_2	2	cyclic	$\begin{pmatrix}2&0\\0&1\end{pmatrix}$
D_2	4	dihedral	$\begin{pmatrix}2&0\\0&1\end{pmatrix}, \begin{pmatrix}2&0\\0&2\end{pmatrix}$
$D_3 = S_3$	6	dihedral	$\begin{pmatrix}2&1\\2&0\end{pmatrix}, \begin{pmatrix}1&0\\1&2\end{pmatrix}$
C_8	8	cyclic	$\begin{pmatrix}1&1\\1&0\end{pmatrix}$
D_4	8	dihedral	$\begin{pmatrix}2&0\\0&1\end{pmatrix}, \begin{pmatrix}0&2\\1&0\end{pmatrix}$
D_6	12	dihedral	$\begin{pmatrix}1&2\\1&0\end{pmatrix}, \begin{pmatrix}0&1\\1&0\end{pmatrix}$
Q_{16}	16	semi-dihedral	$\begin{pmatrix}1&1\\2&1\end{pmatrix}, \begin{pmatrix}0&1\\1&0\end{pmatrix}$
$\mathrm{GL}_2(\mathbb{F}_3)$	48	general linear	$\begin{pmatrix}2&0\\0&1\end{pmatrix}, \begin{pmatrix}2&1\\2&0\end{pmatrix}$

Table 1. Candidates for the image of ρ_E in $\mathrm{GL}_2(\mathbb{F}_3)$.

1. The determinant $\det A$, trace $\operatorname{tr} A$, and the multiplicative order $|A|$ of a matrix in $\mathrm{GL}_2(\mathbb{F}_\ell)$ are all invariant under conjugation. Show that the pair $(\det A, \operatorname{tr} A)$ does not determine the conjugacy class of A in $\mathrm{GL}_2(\mathbb{F}_3)$, but then prove that the triple $(\det A, \operatorname{tr} A, |A|)$ does determine the conjugacy class of A in $\mathrm{GL}_2(\mathbb{F}_3)$.

Thus we can get more information about π_ℓ if, in addition to computing its trace, we also compute its multiplicative order in the ring $\mathrm{End}(E_p[\ell])$.

2. Modify the function `trace_mod` in the Sage workhseet 18.783 Lecture 9: Schoof's algorithm.sws so that it also computes the order of π_ℓ and returns both the trace and the order of π_ℓ. You don't need to handle $\ell = 2$, but if you wish, you can use that fact that the order of π_2 is 1, 2, or 3, depending on whether $f(x)$ has $3, 1$, or 0 roots in $\mathbb{F}_p$, respectively, where $y^2 = f(x)$ is the Weierstrass equation for E.

Note: **The order of π_ℓ must be computed modulo the full division polynomial ψ_ℓ.** So compute $|\pi_\ell|$ before the step that computes q_ℓ, which is the first place where a division-by-zero error could occur, causing h to be replaced by a proper factor. Also, be sure to compute $|\pi_\ell|$ only the first time through the loop when you know that $h = \psi_\ell$, don't accidentally recompute it if the loop repeats more than once.

Now address the first part of part 1 in a different way: pick an elliptic curve $E/\mathbb{Q}$ and find two primes p and p' for which $\pi_3 \in \mathrm{End}(E_p[3])$ and $\pi_3' \in \mathrm{End}(E_{p'}[3])$ have the same characteristic polynomial but different multiplicative orders.

3. Write a program that, given an elliptic curve E, a prime ℓ, and an upper bound N, enumerates the primes $p \le N$ distinct from ℓ and for which E has good reduction, and for each E_p, computes the triple $(\det \pi_\ell, \operatorname{tr} \pi_\ell, |\pi_\ell|)$. Keep a count of how often each distinct triple occurs (use a dictionary, as in the `group_stats` function in 18.783 Problem Set 5 Problem 2.sws. Then normalize the counts by dividing by the number of primes p used, yielding a ratio for each triple.

For $\ell = 3$, use your program to provisionally determine the image of ρ_E for each of the ten elliptic curves below, by comparing the statistics computed by your program with the corresponding statistics for each of the 8 candidate subgroups of $\mathrm{GL}_2(\mathbb{F}_3)$. With N

around 5000 or 10000 you should be able to easily distinguish among the possibilities. The curves below are also listed in the worksheet 18.783 Problem Set 5 Problem 2.sws.

$$y^2 = x^3 + x \qquad y^2 = x^3 + 1$$
$$y^2 = x^3 + 432 \qquad y^2 = x^3 + x + 1$$
$$y^2 = x^3 + 21x + 26 \qquad y^2 = x^3 - 112x + 784$$
$$y^2 = x^3 - 3915x + 113670 \qquad y^2 = x^3 + 4752x + 127872$$
$$y^2 = x^3 + 5805x - 285714 \qquad y^2 = x^3 + 652509x - 621544482$$

4. Note that if a given triple $(\det \pi_\ell, \operatorname{tr} \pi_\ell, |pi_\ell|)$ occurs for some E_p but does not occur in a candidate subgroup $G \subset \mathrm{GL}(\mathbb{F}_\ell)$, you can immediately rule out G as a possibility for the image of ρ_E. Analyze the 8 candidate subgroups in Table 1 to find a pair of triples that arise in $\mathrm{GL}_2(\mathbb{F}_3)$ but do not both arise in any of its proper subgroups. If for a given curve $E/\mathbb{Q}$ you can find both of these triples for some E_{p_1} and E_{p_2}, then you have unconditionally *proven* that ρ_E is surjective for $\ell = 3$.

Use this to devise an algorithm that attempts to prove ρ_E is surjective for $\ell = 3$. Your algorithm should return `true` as soon as it is able to determine $\operatorname{im} \rho_E = \mathrm{GL}_2(\mathbb{F}_3)$ (this should happen quite quickly, if it is true). If this fails to happen after computing triples for E_p for every prime up to, say, 10000, then your algorithm should give up and return `false`. You can think of this as a Monte Carlo algorithm with one-sided error: the "randomness" comes from the assumption that each π_ℓ is uniformly and independently distributed over the image of ρ_E for each prime p. If your program returns `true`, then ρ_E is definitely surjective; if it returns `false` it is almost certainly not surjective, but there is a small probability of error.

Generate 1000 random elliptic curves $E/\mathbb{Q}$ of the form $y^2 = x^3 + Ax + B$, with A and B uniformly distributed over the interval $[1, 10^6]$, and use your program to test whether their mod-3 Galois representation ρ_E is surjective or not. List any and all curves for which your program returns `false`, and provisionally identify the image of ρ_E in each such case as in part 3 above.

Problem 3. Schoof's algorithm (40 points)

In this problem you will analyze the complexity of Schoof's algorithm, as described in the notes for Lecture 9 and implemented in the Sage worksheet

18.783 Lecture 9: Schoof's algorithm.sws.

In your complexity bounds, use $\mathsf{M}(n)$ to denote the complexity of multiplying two n-bit integers. Recall that the complexity of multiplying polynomials in $\mathbb{F}_p[x]$ of degree d is $O(\mathsf{M}(d \log p))$, and the complexity of inverting a polynomial of degree $O(d)$ modulo a polynomial of degree d is $O(\mathsf{M}(d \log p) \log d)$. The space complexity of both operations is $O(d \log p)$.

1. Analyze the time and space complexity of computing t_ℓ as described in Algorithm 9.5 of the lecture notes and implemented in the `trace_mod` function in the worksheet. Give separate bounds for each of the four non-trivial steps in Algorithm 9.5 as well as overall bounds for the entire algorithm. Express your bounds in terms of ℓ and $n = \log p$, using $\mathsf{M}(m)$ to denote the cost of multiplying two m-bit integers.

2. Analyze the overall time and space complexity of Schoof's algorithm, as described in Algorithm 9.1 of the lectures notes and implemented in the `Schoof` function of the worksheet above, as a function of $n = \log p$. Give your answer in two forms, first using $\mathsf{M}(m)$ to express the cost of multiplication, and then using the explicit Schönhage-Strassen bound $\mathsf{M}(m) = O(m \log m \log\log m)$.

3. In your answer to part 1, you should have found that the running time of one particular step strictly dominates the running time of all the other steps of Algorithm 9.5. Explain how to modify Algorithm 9.5 to improve the running time of this step so that its time complexity matches that of the next most time-consuming step in Algorithm 9.5.

4. Revise your time and space complexity estimates in part 2 to reflect part 3.

Problem 4. A Las Vegas algorithm to compute $E(\mathbb{F}_p)$. (40 points)

In this problem you will use the extended discrete logarithm to design a Las Vegas algorithm to determine the structure of $E(\mathbb{F}_p)$ as a sum of two cyclic groups $\mathbb{Z}/N_1\mathbb{Z} \oplus \mathbb{Z}/N_2\mathbb{Z}$, with $N_1|N_2$. We assume that the group order N has already been computed, either by Schoof's algorithm or by the Las Vegas algorithm that you implemented in Problem Set 3. We also assume that we can readily compute (or are given) the prime factorization of N.

Our strategy is to determine the structure of the ℓ-Sylow subgroups of $E(\mathbb{F}_p)$ for each prime ℓ dividing N. If ℓ divides N but ℓ^2 does not, then the ℓ-Sylow subgroup is obviously isomorphic to $\mathbb{Z}/\ell\mathbb{Z}$, so we only need to consider primes whose square divides N. Additionally, we know that if ℓ does not divide $p-1$, then the ℓ-Sylow subgroup must be cyclic (if not, $E[\ell] = E(\mathbb{F}_p)[\ell]$ and π_ℓ is the identity matrix, which implies $\det \pi_\ell \equiv p \equiv 1 \bmod \ell$).

This yields the following high-level algorithm to compute N_1 and N_2, given N.

1. Compute (if not given) the prime factorization N.
2. Set $N_1 = 1$ and $N_2 = 1$, and for each maximal prime power ℓ^e dividing N:
   (a) If $e = 1$ or ℓ does not divide $p-1$, then set $N_2 = \ell^e N_2$ and continue.
   (b) Otherwise, compute the structure $\mathbb{Z}/\ell^{e_1}\mathbb{Z} \oplus \mathbb{Z}/\ell^{e_2}\mathbb{Z}$ of the ℓ-Sylow subgroup of $E(\mathbb{F}_p)$ as described below, with $e_1 \le e_2$, and set $N_1 = \ell^{e_1} N_2$ and $N_2 = \ell^{e_2} N_2$.
3. Output N_1 and N_2

All we need now is an algorithm to compute the ℓ-Sylow subgroup G_ℓ of $E(\mathbb{F}_p)$, given the orders ℓ^e and N of G_ℓ and $E(\mathbb{F}_p)$, respectively. Our strategy is to first pick two random points $P_1, P_2 \in G_\ell$, by generating random points in $E(\mathbb{F}_p)$ and multiplying them by N/ℓ^e. We hope that these points generate G_ℓ. Next, we reduce them to what we hope is a *basis* for G_ℓ, that is, points Q_1 and Q_2 such that $G_\ell \simeq \langle Q_1\rangle \oplus \langle Q_2\rangle$. We then have $G_\ell \simeq \mathbb{Z}/\ell^{e_1}\mathbb{Z} \oplus \mathbb{Z}/\ell^{e_2}\mathbb{Z}$ where $\ell^{e_1} = |Q_1|$, $\ell^{e_2} = |Q_2|$. Note that we can quickly compute the order of any element of G_ℓ, since it must be a power of ℓ. Provided that we know the points Q_1 and Q_2 are *independent*, which means that $\langle Q_1, Q_2\rangle \simeq \langle Q_1\rangle \oplus \langle Q_2\rangle$, to verify that we actually have computed a basis for G_ℓ (and not some proper subgroup), we just need to check that $e_1 + e_2 = e$. If this does not hold, we try again with two new random points P_1 and P_2, secure in the knowledge that we must eventually succeed.

Your job is to flesh out this strategy and analyze the resulting algorithm. We first recall the definition of the extended discrete logarithm given in class.

Definition 1. For elements α and β of a finite group G, the *extended discrete logarithm* of β with respect to α, denoted $\mathrm{DL}^*(\alpha,\beta)$, is the pair of positive integers (x,y) with $\alpha^x = \beta^y$, where y is minimal subject to $\beta^y \in \langle\alpha\rangle$, and $x = \log_\alpha \beta^y$.

Prove each of the following statements, in which G is an arbitrary finite abelian ℓ-group containing elements α and β.

(**1**) If G has ℓ-rank at most 2 and α and β are random elements uniformly distributed over the elements of G, then the probability that $G = \langle\alpha,\beta\rangle$ is at least 3/8.

(**2**) If $(x,y) = \mathrm{DL}^*(\alpha,\beta)$ then y is a power of ℓ.

(**3**) For $(x,y) = \mathrm{DL}^*(\alpha,\beta)$ the following are equivalent:

(a) $x = |\alpha|$ and $y = |\beta|$;

(b) $\langle\alpha,\beta\rangle$ has order $|\alpha|\cdot|\beta|$.

(c) α and β are independent;

(**4**) If $|\alpha| \geq |\beta|$ and $(x,y) = \mathrm{DL}^*(\alpha,\beta)$ then $y|x$ and $\gamma = \beta - (x/y)\alpha$ and α are independent.

The key fact is (4), which tells us that we should order the P_i so that $|P_1| \leq |P_2|$ and then let $Q_1 = P_1 - (x/y)P_2$ and $Q_2 = P_2$, where $(x,y) = \mathrm{DL}^*(P_2,P_1)$. If we then compute $\ell^{e_1} = |Q_1|$ and $\ell^{e_2} = |Q_2|$, it follows from (3) that $G_\ell = \langle Q_1, Q_2\rangle$ if and only if $e_1 + e_2 = e$. Fact (1) tells that we expect this to occur within less than 3 iterations, on average. By (2), we can compute $(x,y) = \mathrm{DL}^*(P_2,P_1)$, by attempting to compute $x = \log_{P_2} \ell^i P_1$ for $i = 0,1,2,\ldots$ until we succeed, at which point we have $y = \ell^i$.[4]

To compute $\log_{P_2} \ell^i P_1$, we use the prime-power case of the Pohlig-Hellman algorithm to reduce the problem to a discrete logarithm computation in a group of prime order ℓ, where we use the baby-steps giant-steps method.

Now do the following:

5. Write down a step-by-step description (not a program) of an algorithm to compute the structure of the ℓ-Sylow subgroup G_ℓ of $E(\mathbb{F}_p)$ in the form $\mathbb{Z}/\ell^{e_1}\mathbb{Z} \oplus \mathbb{Z}/\ell^{e_2}\mathbb{Z}$, given $N = \#E(\mathbb{F}_p)$ and $\ell^e = \#G_\ell$, and analyze its expected time complexity in terms of ℓ and $n = \log p$.

6. Analyze the total expected time complexity of the algorithm to compute the structure of $E(\mathbb{F}_p)$ in the form $\mathbb{Z}/N_1\mathbb{Z} \oplus \mathbb{Z}/N_2\mathbb{Z}$, given $N = \#E(\mathbb{F}_p)$ and its prime factorization, as described above (hint: first determine the worst case scenario).

Problem 5. 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.

[4] There are much better ways to do this (a binary search, for example), but using them won't improve the worst-case complexity of the overall algorithm.

	Interest	Difficulty	Time Spent
Problem 1			
Problem 2			
Problem 3			
Problem 4			

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"), the pace (1="way too slow", 10="way too fast", 5="just right") and the novelty of the material (1="old hat", 10="all new").

Date	Lecture Topic	Material	Presentation	Pace	Novelty
3/7	Schoof's Algorithm				
3/12	Discrete Logarithm Problem (part 1)				

Please feel free to record any additional comments you have on the problem sets or lectures, in particular, ways in which they might be improved.

18.783 Elliptic Curves
Spring 2013