A Faster Randomized Algorithm for Root Counting in Prime Power Rings

Leann Kopp and Natalie Randall

July 17, 2018

Abstract

Let $p$ be a prime and $f \in \mathbb{Z}[x]$ a polynomial of degree $d$ such that $f$ is not identically zero mod $p$. We introduce a Las Vegas randomized algorithm to count the number of roots of $f$ in $\mathbb{Z}/(p^k)$ for $k \in \mathbb{N}$ with $k \geq 2$ which runs in time $d^{1.5+o(1)}(\log p)^{2+o(1)}1.12^k$. We compare the randomized algorithm to simple brute force to see when we have practical time gains. In addition, we present an upper bound on the number of roots of $f$ (as a function of $p$, $k$, and the degree of $f$) that is optimal for $k = 2$.

1 Introduction

A deterministic algorithm for counting roots in $\mathbb{Z}/(p^k)$ in time $(d\log(p)+2^k)^O(1)$ is given in [2]. Here we propose a Las Vegas randomized algorithm which runs in time $d^{1.5+o(1)}(\log p)^{2+o(1)}1.12^k$. By “Las Vegas randomized,” we mean that our algorithm undercounts roots with a fixed error probability but otherwise returns a correct root count and always correctly announces failure. For instance, if we take our fixed error probability to be $\frac{1}{3}$, we can get an overall failure probability of less than $\frac{1}{3^{100}}$ by running the algorithm 100 times. Las Vegas randomized algorithms are common across algorithmic number theory; there are fast, widely accepted Las Vegas randomized algorithms for checking primality and for factoring polynomials over finite fields [1, 3, 4]. In our algorithm, we introduce randomization by using fast factorization (see [3]) to find roots of $f$ in $\mathbb{Z}/(p)$.

Prior to the deterministic algorithm in [2] there was little information on counting the roots of a polynomial over prime power rings. We can easily count the number of roots of a polynomial $f$ in $\mathbb{Z}/(p)$ by taking the degree of $\gcd(x^p-x,f)$, but this method relies on $\mathbb{Z}/(p)$ being a unique factorization domain, and $\mathbb{Z}/(p^k)$ is not a unique factorization domain for $k > 1$. To overcome this issue, we consider the Taylor expansion of our polynomial $f$ about a root $\zeta$ of the mod $p$ reduction of $f$ with a perturbation of $p\varepsilon$, where $\varepsilon \in \{0,\ldots,p^k-1\}$. From this expansion, we can divide by certain powers of $p$ in order to recursively isolate the roots of $f$ in the ring $\mathbb{Z}/(p^k)$. From a similar expansion, we also get an upper bound for the number of roots of $f$ in $\mathbb{Z}/(p^k)$ given by $\min\{d,p\}p^{k-1}$ and a sharp upper bound for $k = 2$ given by $\min\{\lfloor\frac{d}{2}\rfloor, p\}p^{k-1} + (d \mod 2)$.

2 Background and Randomized Algorithm

Lemma 2.1 (Hensel’s Lemma). If $f \in \mathbb{Z}[x]$ is a polynomial with integer coefficients, $p$ is prime, and $\zeta_j \in \{0,\ldots,p^{j-1}-1\}$ is a root of $f$ (mod $p^j$) and $f'(\zeta_j) \neq 0$ (mod $p$), then there is a unique $\zeta \in \{0,\ldots,p^{j+1}-1\}$ with $f(\zeta) = 0$ (mod $p^{j+1}$) and $\zeta = \zeta_j$ (mod $p^j$).
We will see below that we can use Hensel’s Lemma to determine the number of lifts of a root \( \zeta \) with \( s(i, \zeta) = 1 \).

Consider the expansion of \( f \) given by

\[
f(\zeta + p\varepsilon) = f(\zeta) + f'(\zeta)p\varepsilon + \cdots + \frac{f^{(\min(d,k-1))(\zeta)}}{p^{\min(d,k-1)!}} p^{\min(d,k-1)} \varepsilon^{\min(d,k-1)} \text{ mod } p^k,
\]

where \( \zeta \) is a root of the mod \( p \) reduction of \( f \). Let \( s \in \{1, \ldots, k\} \) be the maximal integer such that \( p^s \) divides each of \( f(\zeta), f'(\zeta)p, \ldots, \frac{f^{(\min(d,k-1))(\zeta)}}{p^{\min(d,k-1)!}} p^{\min(d,k-1)} \). More precisely, \( s = \min\{\text{ord}_p(f(\zeta)), \text{ord}_p(f'(\zeta)p), \ldots, \text{ord}_p(f^{(\min(d,k-1))(\zeta)}p^{\min(d,k-1)})\} \), where \( \text{ord}_p(x) \) refers to the p-adic valuation of \( x \).

If \( f(\zeta + p\varepsilon) = 0 \text{ (mod } p^k) \), then we can write

\[
p^s \left( \frac{f(\zeta)}{p^s} + \frac{f'(\zeta)}{p^{s-1}} \varepsilon + \cdots + \frac{f^{(\min(d,k-1))(\zeta)}}{p^{\min(d,k-1)!}} \varepsilon^{\min(d,k-1)} \right) = 0 \text{ mod } p^k,
\]

which is true if and only if

\[
\frac{f(\zeta)}{p^s} + \frac{f'(\zeta)}{p^{s-1}} \varepsilon + \cdots + \frac{f^{(\min(d,k-1))(\zeta)}}{p^{\min(d,k-1)!}} \varepsilon^{\min(d,k-1)} = 0 \text{ mod } p^k - s.
\]

In the case where \( s = 1 \), if \( f'(\zeta) \equiv 0 \text{ mod } p \) then we must have that \( p^2 \mid f(\zeta) \) and so \( \zeta \) has no lifts mod \( p^k \); however, if \( f'(\zeta) \not\equiv 0 \text{ mod } p \), then \( \zeta \) lifts to one unique root by Hensel’s Lemma. In the case where \( s = k \), the entire expression \( p^s \left( \frac{f(\zeta)}{p^s} + \frac{f'(\zeta)}{p^{s-1}} \varepsilon + \cdots + \frac{f^{(\min(d,k-1))(\zeta)}}{p^{\min(d,k-1)!}} \varepsilon^{\min(d,k-1)} \right) \) vanishes identically mod \( p^k \), so any \( \varepsilon \in \{0, \ldots, p^{k-1}\} \) is a zero of \( f(\zeta + p\varepsilon) \) and therefore we have that \( \zeta \) has \( p^k - 1 \) lifts.

The key idea of the randomized algorithm is that counting the number of roots when \( s = 1 \) and \( s = k \) is simple, as described above, and we can reduce all the computations to these two cases using recursion. If \( s \in \{2, \ldots, k-1\} \), we can reapply the algorithm to an instance of counting roots for the polynomial \( \frac{f(\zeta)}{p^s} + \frac{f'(\zeta)}{p^{s-1}} \varepsilon + \cdots + \frac{f^{(\min(d,k-1))(\zeta)}}{p^{\min(d,k-1)!}} \varepsilon^{\min(d,k-1)} \) in \( \mathbb{Z}/(p^{k-s}) \). Eventually this will reduce to the case where either \( s = 1 \) or \( s = k \) and the recursion will terminate, giving us that the root \( \zeta \) of \( f \mod p \) has a total number of \( p^{s-1} \) (the number of roots of \( \frac{f(\zeta)}{p^s} + \frac{f'(\zeta)}{p^{s-1}} \varepsilon + \cdots + \frac{f^{(\min(d,k-1))(\zeta)}}{p^{\min(d,k-1)!}} \varepsilon^{\min(d,k-1)} \text{ mod } p^{-s}) \) lifts to roots in \( \mathbb{Z}/(p^k) \).

**Algorithm 1 Randomized Prime Power Root Counting**

1: function COUNT\( (f \in \mathbb{Z}[x] \) has degree \( d \) and is not identically 0 mod \( p \), prime \( p \), \( k \in \mathbb{N} \) such that \( k \geq 2) \)
2: Factor \( f \) as in [3]
3: count := number of distinct linear factors of multiplicity 1 ▷ These roots in \( \mathbb{Z}/(p) \) can be lifted uniquely to roots in \( \mathbb{Z}/(p^k) \).
4: Push \( \{\zeta_0 \in \{0, \ldots, p-1\} | f(\zeta_0) = f'(\zeta_0) = 0 \text{ mod } p \text{ and } f(\zeta_0) = 0 \text{ mod } p^2\} \) onto a stack \( S \)
5: while \( S \neq 0 \) do
6: Pop a root \( \zeta_0 \) from the stack and define \( s(0, \zeta_0) := \) maximal integer such that \( p^{s(0, \zeta_0)} \) divides each of \( f(\zeta_0), f'(\zeta_0)p\varepsilon, \ldots, \frac{f^{(\min(d,k-1))(\zeta_0)}}{p^{\min(d,k-1)!}} p^{\min(d,k-1)} \)
7: if \( s(0, \zeta_0) = k \) then
8: count ← count + \( k^k \)
9: else
10: Define \( f_{\zeta_0}(x) := \frac{1}{p^{s(0, \zeta_0)}} f(\zeta_0 + px) \)
11: count ← count + \( p^{s(0, \zeta_0)} \cdot \text{COUNT}(f_{\zeta_0}(x), p, k - s(0, \zeta_0)) \)
12: end if
13: end while
14: return count
15: end function
Since the non-degenerate roots of the mod $p$ reduction of $f$ have a unique lift by Hensel’s Lemma, we only need to keep track of the degenerate roots. Our recurrence takes a degenerate root $\zeta_0$ as a point in a cluster of roots of $f$ in $\mathbb{Z}/(p^k)$ and recovers the other points in this cluster by expanding $\zeta_0$ to more digits base-$p$. In this way, we count the number of roots of $f$ in $\mathbb{Z}/(p^k)$ by counting the number of lifts from each root $\zeta_i$ of $f$ in $\mathbb{Z}/(p)$.

3 Discussion of Complexity Bound and Experimental Data

![Diagram of complexity tree](image)

Figure 1: Diagram of complexity tree

In the Figure 1, we see the basic tree structure of the algorithm. We need only keep track of the degenerate roots of $f$; the degenerate roots of $f$, denoted by $\zeta_i$, become the children
nodes from which more branching occurs. The depth and branching of our recurrence tree is strongly limited by the value of \( k \) and the degree of \( f \). For the initial parent node, the total number of degenerate roots is less than or equal to \( \frac{s(i,\zeta_i) - d_{i,\zeta_i}}{2} \). Non-degenerate roots have a unique lift by Hensel’s Lemma, so non-degenerate roots require no additional computations and are therefore shown on the left of the tree. We also see that we have a maximum of \( s(1,\zeta_1) \cdots s(l,\zeta_l) \) nodes at the bottom level of the tree.

We use Kedlaya-Umans fast \( \mathbb{Z}/(p)[x] \) factoring algorithm found in [3], which takes time \( d^{1.5+o(1)}(\log p)^{1+o(1)} + d^{1+o(1)}(\log p)^{2+o(1)} \) for a degree \( d \) polynomial, in order to factor the polynomials at each node in \( \mathbb{Z}/(p) \). In simplest terms, we can consider our total complexity as being less than or equal to (the number of nodes in the recursion tree) \( \times \) (the complexity of factoring over \( \mathbb{Z}/(p)[x] \)). Optimizing parameters, the worst case occurs when \( d \approx e \approx 2.71828 \) and the depth of the tree is \( \frac{k}{e} \). The final complexity of the randomized algorithm is given by

\[
(d^{1.5} \log p)^{1+o(1)} + (d \log^2 p)^{1+o(1)} + [(\min\{d, k-1\})^{1.5} \log p]^{1+o(1)} + (\min\{d, k-1\} \log^2 p)^{1+o(1)} \bigg(\frac{e/2}{k/e}\bigg)^{\lfloor k/e \rfloor},
\]

where \((e/2)^{\lfloor k/e \rfloor} \approx 1.12^k\).

Based on this complexity bound, we expect to see time improvements even for \( p \) as small as 2 when compared to brute-force counting since brute-force counting takes time approximately \( p^k \), giving us that brute-force takes time approximately \( 2^k \) for \( p = 2 \), while the randomized algorithm takes time approximately \( 1.12^k \). More details regarding computational time with \( p = 2 \) are given in Tables 1 and 2.

We now present computational data which illustrates the advantages to using the randomized algorithm over the brute force method. The brute force method takes a polynomial \( f \), a prime \( p \), and a power \( k \), and evaluates \( f \) at each value \( i \) from 0 to \( p^k - 1 \). If \( f(i) \) is identically equal to 0 (mod \( p^k \)), then that contributes to the total number of roots of \( f \in \mathbb{Z}/(p^k) \). We start by comparing the run times of the brute force algorithm and the randomized algorithm for \( p = 2 \).

Table 1 displays the average difference in computation time for the number of roots of 100 random polynomials of degree less than or equal to 100 in \( \mathbb{Z}/(2^k) \) for the given \( k \), between brute force and the randomized algorithm (negative implies brute force was faster). The times are shown in seconds. In general, a single computation took less than a second, so differences in the milliseconds are not insignificant. From the table, we see a switch from brute-force being more efficient to the randomized algorithm being more efficient at \( k = 10 \), and the difference becomes more pronounced as \( k \) increases.

<table>
<thead>
<tr>
<th>( k )</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg Diff (in seconds)</td>
<td>-0.0011</td>
<td>-0.00029</td>
<td>0.0028</td>
<td>0.01701</td>
<td>0.32499</td>
</tr>
</tbody>
</table>

Table 1: Average Difference in Run Times for 100 random polynomials with \( p = 2 \), taken as (time of brute-force) - (time of randomized algorithm)
polynomial in $x$ when brute force method almost 4 hours to count the number of roots of a polynomial. The difference between brute force and the randomized algorithm for large primes; it took the algorithm for prime numbers with at least four digits. We begin to see a very significant computations involving large prime numbers. Table 4 displays the run times for the randomized larger primes.

Table 2 shows the difference in computational time with specific examples, giving an idea of the overall time it takes for both the randomized algorithm and brute-force to run when $p = 2$. The difference in computational run time becomes more noticeable when we introduce larger primes.

<table>
<thead>
<tr>
<th>$f$</th>
<th>$p$</th>
<th>$k$</th>
<th>Brute Force</th>
<th>Randomized Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-71x^4 + 21x^3 - 84x^2 + 47x + 63$</td>
<td>2</td>
<td>5</td>
<td>0 ns</td>
<td>0 ns</td>
</tr>
<tr>
<td>$21x^5 - 66x^4 - 24x^3 - 88x^2 + 17x - 32$</td>
<td>2</td>
<td>6</td>
<td>0 ns</td>
<td>1000.00 μs</td>
</tr>
<tr>
<td>$-75x^6 + 82x^5 - 93x^4 - 19x^3 + 3x + 65$</td>
<td>2</td>
<td>7</td>
<td>1000.00 μs</td>
<td>1000.00 μs</td>
</tr>
<tr>
<td>$x^7 + x^6 + 62x^5 - 23x^4 - 58x - 66$</td>
<td>2</td>
<td>8</td>
<td>1000.00 μs</td>
<td>1000.00 μs</td>
</tr>
<tr>
<td>$48x^8 - 23x^7 + 90x^6 - 19x^5 + 31x + 7$</td>
<td>2</td>
<td>9</td>
<td>3.00 ms</td>
<td>1000.00 μs</td>
</tr>
<tr>
<td>$80x^8 - 37x^7 - 89x^6 + 58x^5 + 32x^4 - 61$</td>
<td>2</td>
<td>10</td>
<td>5.00 ms</td>
<td>0 ns</td>
</tr>
<tr>
<td>$-52x^9 + 51x^8 - 75x^7 - 23x^6 - 27x^5 - 38x$</td>
<td>2</td>
<td>11</td>
<td>11.00 ms</td>
<td>3.00 ms</td>
</tr>
<tr>
<td>$61x^{10} - 80x^9 - 17x^8 + 90x^7 + 13x^6 + 68$</td>
<td>2</td>
<td>12</td>
<td>51.00 ms</td>
<td>2.00 ms</td>
</tr>
<tr>
<td>$18x^{11} + 51x^8 + 49x^7 + 34x^6 - 64x^5 + 70$</td>
<td>2</td>
<td>13</td>
<td>35.00 ms</td>
<td>2.00 ms</td>
</tr>
<tr>
<td>$89x^{12} - 56x^9 + 73x^8 - x^4 + 80x^3 + 69x^2$</td>
<td>2</td>
<td>14</td>
<td>75.00 ms</td>
<td>6.00 ms</td>
</tr>
<tr>
<td>$-93x^{13} - 36x^8 + 53x^7 - 78x^6 - 67x^5 + 88$</td>
<td>2</td>
<td>15</td>
<td>212.00 ms</td>
<td>2.00 ms</td>
</tr>
</tbody>
</table>

Table 3: Run times for $p$ with at least 3 digits

Table 3 illustrates the advantages of using the randomized algorithm over brute-force for computations involving large prime numbers. Table 4 displays the run times for the randomized algorithm for prime numbers with at least four digits. We begin to see a very significant difference between brute force and the randomized algorithm for large primes; it took the brute force method almost 4 hours to count the number of roots of a polynomial in $\mathbb{Z}/(p^k)$ when $p$ was a 4 digit prime number, while the randomized algorithm counted the roots of a polynomial in $\mathbb{Z}/(p^k)$ when $p$ was a 9 digit prime in approximately a minute and a half.

<table>
<thead>
<tr>
<th>$f$</th>
<th>$p$</th>
<th>$k$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-56x^{14} + 73x^{13} - x^{12} + 80x^{10} + 69x^9 + 76x$</td>
<td>8713</td>
<td>3</td>
<td>2.00ms</td>
</tr>
<tr>
<td>$53x^{14} - 78x^{13} + 67x^{12} + 88x^{10} - 5x^9 - 36x^8$</td>
<td>13177</td>
<td>3</td>
<td>4.00ms</td>
</tr>
<tr>
<td>$55x^{14} + 69x^{13} + 86x^{10} - 23x^{12} + 17x^{11} + 31x^2$</td>
<td>95213</td>
<td>3</td>
<td>27.00ms</td>
</tr>
<tr>
<td>$35x^{14} + 34x^{13} - 14x^{12} - 92x^{11} + 90x^{12} - 32x^4$</td>
<td>104729</td>
<td>3</td>
<td>29.00ms</td>
</tr>
<tr>
<td>$62x^{14} + 31x^{13} + 51x^{12} + 98x^{10} - 80x^9 - 51x^5$</td>
<td>15485863</td>
<td>3</td>
<td>5.08s</td>
</tr>
<tr>
<td>$-40x^{14} - 10x^{13} + 67x^{10} - 40x^{11} - 82x^{10} - 82x^8$</td>
<td>104395301</td>
<td>3</td>
<td>41.49s</td>
</tr>
<tr>
<td>$-80x^{14} - 72x^{13} + 36x^{10} + 71x^{12} + 54x^8 + 84x^{12}$</td>
<td>179424673</td>
<td>3</td>
<td>92.51s</td>
</tr>
</tbody>
</table>

Table 4: Run times for the randomized algorithm when $p$ has $\geq 4$ digits
We expect the randomized algorithm to take the longest when a polynomial has many degenerate roots because a polynomial of this type will require many recursive calls. Polynomials with many degenerate roots do take longer than a random polynomial, but overall the randomized algorithm still outperforms other methods. For instance, counting roots of the 55 degree polynomial \((x - 1)(x - 2)^2 \cdots (x - 10)^{10}\) in \(\mathbb{Z}/(31^{10})\) took 6.4 seconds using the randomized algorithm, while counting roots in the same ring with a random polynomial of the same degree took only 1 millisecond. Despite this slowdown for polynomials with very degenerate roots, the randomized algorithm still outperforms other methods; counting roots of a polynomial in just \(\mathbb{Z}/(31^6)\) using brute force took 2.7 hours.

4 Bound on Number of Roots

**Lemma 4.1.** If a root \(\zeta\) of the mod \(p\) reduction of \(f\) has multiplicity \(j\), then \(s_\zeta \leq j\), where \(s_\zeta\) is the greatest integer such that \(p^s_\zeta\) divides each of \(f(\zeta), \ldots, f^{(k-1)}(\zeta)\).

*Proof.* If \(\zeta\) has multiplicity \(j\), then \(f(\zeta) = \cdots = f^{j-1}(\zeta) = 0 \pmod{p}\), but \(f^{(j)}(\zeta) \neq 0 \pmod{p}\). So \(\frac{f^{(i)}(\zeta)}{i!} p^i\) is divisible by \(p^j\) but not \(p^{j+1}\) and therefore \(s_\zeta \leq j\). \(\square\)

**Theorem 4.2.** Let \(p\) be a prime, \(f \in \mathbb{Z}[x]\) a polynomial of degree \(d\), and \(k \in \mathbb{N}\) such that \(d \geq k \geq 2\). Then \(N_f(p, d, k) \leq \min\{d, p\} p^{k-1}\), where \(N_f(p, d, k)\) denotes the number of roots of \(f\) in \(\mathbb{Z}/(p^k)\).

*Proof.* Let \(\zeta_i \in \{0, \ldots, p - 1\}\) be any root of the mod \(p\) reduction of \(f\), and let \(s(i, \zeta_i)\) be the greatest integer such that \(p^{s(i, \zeta_i)}\) divides each of \(f(\zeta_i), \ldots, f^{\min(d,k-1)}(\zeta_i)\). Let \(f_\zeta(x) = \frac{1}{p^{s(i, \zeta_i)}} f(\zeta + px)\). Clearly, we have that \(N_f(p, d, 1) \leq \min\{d, p\}\). We know from Lemma 4.1 that if \(\zeta_i \in \mathbb{Z}/(p)\) is a root of multiplicity \(J\), then \(J \leq s\). Let \(\delta_1\) denote the number of non-degenerate roots of \(f\) mod \(p\). From this, we see that

\[
N_f(p, d, k) \leq \delta_1 + \sum_{J=2}^{\min\{d,k-1\}} \sum_{\zeta_i, s(i, \zeta_i) = J} p^{s(i, \zeta_i) - 1} \cdot N_{f_\zeta}(p, k - 1, k - s(i, \zeta_i)) + \sum_{\zeta_i, s(i, \zeta_i) = k} p^{k-1}.
\]

Considering that \(N_{f_\zeta}(p, k - 1, k - s(i, \zeta_i)) \leq p^{k-s(i, \zeta_i)}\), we get

\[
N_f(p, d, k) \leq \sum_{J=1}^{\min\{d,k-1\}} \sum_{\zeta_i, s(i, \zeta_i) = J} p^{s(i, \zeta_i) - 1} \cdot p^{k-s(i, \zeta_i)} + \sum_{\zeta_i, s(i, \zeta_i) = k} p^{k-1};
\]

\[
N_f(p, d, k) \leq \sum_{J=1}^{\min\{d,k-1\}} \sum_{\zeta_i, s(i, \zeta_i) = J} p^{k-1} + \sum_{\zeta_i, s(i, \zeta_i) = k} p^{k-1};
\]

\[
N_f(p, d, k) \leq \sum_{J=1}^{k} \sum_{\zeta_i, s(i, \zeta_i) = J} p^{k-1}.
\]

Since the number of distinct roots of the mod \(p\) reduction of \(f\) is less than \(\min\{d, p\}\), we get that \(N_f(p, d, k) \leq \min\{d, p\} p^{k-1}\), as desired. \(\square\)

Examples of polynomials with more than \(\lfloor \frac{d}{k} \rfloor p^{k-1}\) roots are given below. These examples show that our bound is within a factor of \(k\) of optimality when \(d \leq p\).

**Example 4.3.** \((x - 2)^7(x - 1)^3\) with \(p = 17, k = 7\) has 24,221,090 roots, which is greater than \(\lfloor \frac{d}{k} \rfloor p^{k-1} = 24,137,569\).

**Example 4.4.** \((x - 1)^k x\) has \(p^{k-1} + 1\) roots when \(d = k + 1 \leq p\).

The following examples show that we can have \(p^k\) roots when \(d \geq p\).

**Example 4.5.** \((x^p - x)^k\) is a polynomial of degree \(pk\) with \(p^k\) roots in \(\mathbb{Z}/(p^k)\).

**Example 4.6.** \((x^{p^k-p^{k-1}} - 1)x^k\) has degree \(p^k - p^{k-1} + k\) and also vanishes on all of \(\mathbb{Z}/(p^k)\).
Theorem 4.7. Let $p$ be a prime and $f \in \mathbb{Z}[x]$ a polynomial of degree $d$ such that $d \geq 2$. Then the number of roots of $f$ in $\mathbb{Z}/(p^2)$ is less than or equal to $\min\{\lfloor \frac{d}{2} \rfloor, p\}p + (d \mod k)$, and this bound is sharp.

Proof. Let $\zeta_i \in \{0, \ldots, p-1\}$ be any root of the mod $p$ reduction of $f$, and let $s(i, \zeta_i)$ be the greatest integer such that $p^{s(i, \zeta_i)}$ divides each of $f(\zeta_i), \ldots, \frac{f^{\min(d,k-1)}(\zeta_i)}{\min(d,k-1)!}p^{k-1}$. Let $\delta_1$ denote the number of roots of $f$ in $\mathbb{Z}/(p)$ with $s(i, \zeta_i) = 1$, and let $\delta_2$ denote the number of roots of $f$ in $\mathbb{Z}/(p)$ with $s(i, \zeta_i) = 2$. We know that $\delta_1 + 2\delta_2 \leq d$ and that $\delta_2 \leq \lfloor \frac{d}{2} \rfloor$ by Lemma 4.1. Using this,

- $N_f(p, d, 2) \leq \delta_1 + p\delta_2$,
- $N_f(p, d, 2) \leq (d - 2\delta_2) + p\delta_2$,
- $N_f(p, d, 2) \leq (d - 2\lfloor \frac{d}{2} \rfloor) + \lfloor \frac{d}{2} \rfloor p$,
- $N_f(p, d, 2) \leq \lfloor \frac{d}{2} \rfloor p + (d \mod 2)$.

To show that this bound is sharp, we give several examples below for which this bound equals the number of roots of $f$ in $\mathbb{Z}/(p^2)$.

Example 4.8. With $p = 5$, the degree 3 polynomial $(x - 1)^2x$ has $\lfloor \frac{3}{2} \rfloor \cdot 5 + (3 \mod 2) = 6$ roots in $\mathbb{Z}/(p^2)$.

Example 4.9. In general, for $i, j \in \mathbb{Z}/(p)$ such that $i \neq j$, the polynomial $(x - i)^2(x - j)$ has $\lfloor \frac{d}{2} \rfloor p + (d \mod 2)$ roots in $\mathbb{Z}/(p^2)$ when $d \geq 2$ and $\lfloor \frac{d}{2} \rfloor \leq p$.

5 Acknowledgements

We would like to thank our advisor, Dr. J. Maurice Rojas, for his assistance and guidance; we would also like to thank Yuyu Zhu for all the suggestions and advice she gave us. We also thank Texas A&M University for hosting and the National Science Foundation (NSF) for funding this program.

References


