Solutions to Homework Assignment 4
CS 6104: Algorithmic Number Theory


Suppose

\[ f = \sum_{i \leq s} c_i X^i \]
\[ g = \sum_{i \leq t} a_i X^i \]

\( f = g^2 \) if and only if:

1. \( s = 2t \); and

2. \( \sum_{i \leq s} c_i X^i = (\sum_{i \leq t} a_i X^i)^2 = \sum_{i \leq 2t} (\sum_{j=i-t}^t a_j a_{i-j}) X^i \)

That is,

1) \( s = 2t \); and

2) \( c_i = \sum_{j=i-t}^t a_j a_{i-j} \quad (i \leq s) \).

If \( k \) is of characteristic 2, then the conditions are:

1. \( s = 2t \)

2. For the coefficients:

\[
c_i = \sum_{j=i-t}^t a_j a_{i-j}
\]

\[
= \begin{cases} 
  a_{i/2}^2 + 2 \sum_{j=i/2+1}^t a_j a_{i-j} & (i = 2t, 2t-2, \ldots) \\
  2 \sum_{j=(i+1)/2}^t a_j a_{i-j} & (i = 2t-1, 2t-3, \ldots)
\end{cases}
\]

\[
= \begin{cases} 
  a_{i/2}^2 & (i = 2t, 2t-2, \ldots) \\
  0 & (i = 2t-1, 2t-3, \ldots)
\end{cases}
\]

If \( k \) is not of characteristic 2, then the conditions are:

1) \( s = 2t \); and
2) For the coefficients:

\[ c_i = \sum_{j=t-1}^{t} a_j a_{i-j} = \begin{cases} a_i^2 & (i = s) \\ 2a_t a_{i-t} + \sum_{j=i-t+1}^{t-1} a_j a_{i-j} & (i < s) \end{cases} \]

That is,

\[ a_t = \sqrt{c_s} \]

\[ a_i = \frac{c_{i+t} - \sum_{j=i+1}^{t-1} a_j a_{i-j}}{2a_t} \quad i < t. \]

This requires that \( c_s \) be a square.

Hence, if \( k \) is of characteristic 2, \( f \) is a square in \( k((1/x)) \) if and only if \( \text{deg}(f) \) is even, and \( \alpha_{2i-1} = 0 \) \((i \leq s/2)\). If \( k \) is not of characteristic 2, \( f \) is a square in \( k((1/x)) \) if and only if \( \text{deg}(f) \) is even, and the first coefficient is a square in \( k \).

---

Problem 2. [Solution Courtesy of Craig Struble] This problem is inspired by problem 13 in Chapter 6. For \( m \geq 1 \), define

\[ \tau(m) = \frac{m}{\phi(m)}, \]

where \( \phi \) is the Euler phi function.

A. For what value of \( m \), where \( 1 \leq m \leq 10,000,000 \), is \( \tau(m) \) maximized?

B. More generally, for what values of \( m \) (as \( m \) goes from 1 to \( \infty \)), does \( \tau(m) \) reach new maxima? (A new maximum is an \( m \) such that \( \tau(m') < \tau(m) \), whenever \( m' < m \).)

C. Use methods from Chapter 2 to show Landau’s result that \( \tau(m) = O(\log \log m) \).

D. Fix a prime \( p \). Give an asymptotic lower bound on the probability that a randomly selected polynomial in \( \mathbb{F}_p[X] \) of degree \( n \) is primitive.

---

A. The value at which \( \tau(m) \) is maximized where \( 1 \leq m \leq 10,000,000 \) is

\[ m = 9,699,690 \]

\[ = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \]

Part B explains why this is the maximum value.
B. Suppose \( m = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k} \) is the unique prime factorization of \( m \). Equation (2.2) on page 23 of the text states

\[
\phi(m) = \prod_{1 \leq i \leq k} (p_i - 1)p_i^{e_i - 1}.
\]

Simplifying \( \tau(m) \),

\[
\tau(m) = \frac{m}{\phi(m)} = \frac{p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}}{(p_1 - 1)p_1^{e_1 - 1}(p_2 - 1)p_2^{e_2 - 1} \cdots (p_k - 1)p_k^{e_k - 1}} = \frac{p_1p_2 \cdots p_k}{(p_1 - 1)(p_2 - 1) \cdots (p_k - 1)}.
\]

The value of \( \tau(m) \) depends only on the prime factors of \( m \), regardless of their exponents. So, consider only values of \( m \) that are the product of unique primes. Each prime \( p \) contributes a factor of

\[
\frac{p}{p - 1} = \frac{1}{1 - \frac{1}{p}}
\]

to \( \tau(m) \). Clearly, if \( x < y \), then \( \frac{1}{1 - x} > \frac{1}{1 - y} \). So \( \tau(m) \) is maximized by multiplying primes that are as small as possible; that is, \( \tau(m) \) is maximized when \( m \) is the product of the first \( k \) primes, and the maximum changes when \( m \) is multiplied by the next prime. So, to find the value \( m \) that maximizes \( \tau(m) \) when \( 1 \leq m \leq n \), multiply consecutive primes \( p_i \) together until \( m = p_1p_2 \cdots p_k \leq n \leq p_1p_2 \cdots p_{k+1} \).

C. For this part, assume that \( m = p_1p_2 \cdots p_k \) is the product of the first \( k \) primes. We see from Part B that

\[
\tau(m) = \prod_{i=1}^{k} \frac{1}{1 - \frac{1}{p_i}}.
\]

To use the techniques in Chapter 2, we need to manipulate the product and find a sum that can be bounded. Consider writing

\[
\tau(m) = \prod_{i=1}^{k} e^{f_i}
\]

where \( e \) is the exponential and \( f_i \) is a polynomial such that

\[
e^{f_i} = \frac{1}{1 - \frac{1}{p_i}}.
\]

Hence,

\[
f_i = \ln \left( \frac{1}{1 - \frac{1}{p_i}} \right)
\]
Using the laws of logarithms and Maclaurin expansion,

\[
\ln \left( \frac{1}{1 - \frac{1}{p_i}} \right) = - \ln \left( 1 - \frac{1}{p_i} \right) = \frac{1}{p_i} + \frac{1}{2p_i^2} + \frac{1}{3p_i^3} + \cdots = \frac{1}{p_i} + O \left( \frac{1}{p_i^2} \right)
\]

So, \( f_i = \frac{1}{p_i} + O \left( \frac{1}{p_i^2} \right) \). Now \( \tau(m) \) can be written as

\[
\tau(m) = \prod_{i=1}^{k} e^{\frac{1}{p_i} + O \left( \frac{1}{p_i^2} \right)} \leq \prod_{i=1}^{k} e^{\frac{1}{p_i} + \frac{N}{p_i^2}}
\]

where \( N \) is a constant as defined for the \( O \) notation. Exponents add when multiplying powers together, so now we can apply techniques from Chapter 2. Begin by bounding the sum of the \( \frac{1}{p_i} \) terms.

\[
\sum_{p \leq p_k} \frac{1}{p} \sim \sum_{n \leq p_k} \frac{1}{n \log n} \approx \int_{2}^{p_k} \frac{1}{t \log t} dt = \log \log p_k - \log \log 2 = \log \log p_k
\]

To bound the sum \( \sum_{i=1}^{k} \frac{N}{p_i^2} \), note that \( \sum_{x=1}^{\infty} \frac{1}{x^2} \) converges. Thus the sum \( \sum_{i=1}^{k} \frac{N}{p_i^2} \) also converges to a constant, call it \( D \). Now, ignoring constant factors, we get

\[
\prod_{i=1}^{k} e^{\frac{1}{p_i} + \frac{N}{p_i^2}} \sim e^{\log \log p_k + D} \sim e^D \log p_k \sim \log p_k
\]

One final step is necessary to reach our goal. How is \( p_k \) related to \( m \)? Consider \( \log m \),

\[
\log m = \log(p_1 p_2 \cdots p_k) = \log p_1 + \log p_2 + \cdots + \log p_k = \sum_{p \leq p_k} \log p \sim \sum_{n=1}^{p_k} \log n \log n
\]
\[
\begin{align*}
\tau(m) &= \sum_{n=1}^{p_k} \frac{1}{p_k} \\
&= \Theta(\log \log m).
\end{align*}
\]

Hence, \( \tau(m) = O(\log \log m) \).

**D.** The number of monic primitive polynomials of degree \( n \) in \( \mathbb{F}_p[X] \) is

\[
\frac{\phi(p^n - 1)}{n}.
\]

The total number of monic polynomials of degree \( n \) in \( \mathbb{F}_p[X] \) is \( p^n \). So the probability of selecting a primitive polynomial of degree \( n \) in \( \mathbb{F}_p[X] \) is

\[
\frac{\phi(p^n - 1)}{np^n}.
\]

In Part C, we gave an upper bound for \( \tau(m) \). Use this to obtain a lower bound for \( \phi(m) \).

\[
\begin{align*}
\tau(m) &= \frac{m}{\phi(m)} \\
O(\log \log m) &= \frac{m}{\phi(m)} \\
O\left(\frac{\log \log m}{m}\right) &= \frac{1}{\phi(m)} \\
\phi(m) &= \Omega\left(\frac{m}{\log \log m}\right).
\end{align*}
\]

The probability is then bounded by

\[
\begin{align*}
\frac{\phi(p^n - 1)}{np^n} &= \Omega\left(\frac{p^n - 1}{np^n \log \log (p^n - 1)}\right) \\
&= \Omega\left(\frac{1}{n \log (p^n - 1)}\right) \\
&= \Omega\left(\frac{1}{n \log (n \log p)}\right) \\
&= \Omega\left(\frac{1}{n \log n}\right).
\end{align*}
\]

---

**Problem 3. [Solution Courtesy of Degong Song]** Chapter 7, problem 4. Flesh out the solution in the back of the book.

From \( p \equiv 3 \pmod{4} \) we get \( \left(\frac{-1}{p}\right) = -1 \). This means that the equation \( X^2 = -1 \) does not have solution in \( \mathbb{F}_p \), so \( i \not\in \mathbb{F}_p \).
From \( i^2 = -1 \) and definition of \( \mathbb{F}_p(i) \), we know

\[
\mathbb{F}_p(i) = \left\{ \frac{a + bi}{c + di} \mid a, b, c, d \in \mathbb{F}_p \right\},
\]

where \( c \) and \( d \) can not be 0 simultaneously.

First we show that \( c^2 + d^2 = 0 \) if and only if \( c = d = 0 \). Otherwise, assume \( c \neq 0 \), then \( c^{-1} \in \mathbb{F}_p \) and hence from \( (c^{-1})^2 (c^2 + d^2) = 0 \) we see that \( (c^{-1})d + 1 = 0 \) while \( c^{-1}d \in \mathbb{F}_p \). This is in contradiction with \( \left( \frac{-1}{p} \right) = -1 \).

Thus, from

\[
\frac{a + bi}{c + di} = \frac{(a + bi)(c - di)}{(c + di)(c - di)} = \frac{ac + bd}{c^2 + d^2} + \frac{bc - ad}{c^2 + d^2}i,
\]

and \( c^2 + d^2 \in \mathbb{F}_p^* \), it is not difficult to verify that

\[
\mathbb{F}_p(i) = \mathbb{F}_p[i] = \{ a + bi \mid a, b \in \mathbb{F}_p \}.
\]

\( \mathbb{F}_p(i) \) (or \( \mathbb{F}_p[i] \)) has \( p^2 \) elements and it is an extension field of \( \mathbb{F}_p \) with operation compatible to that of \( \mathbb{F}_p \). From book, any two finite fields with \( p^2 \) elements are isomorphic, and from above discussion, one Model of \( \mathbb{F}_p^* \) can be given as \( \mathbb{F}_p(i) \).

Since \( \left( \frac{1}{p} \right) = 1 \) and \( \left( \frac{p-1}{p} \right) = \left( \frac{-1}{p} \right) = -1 \), we can use binary search to find a \( x \) such that

\[ 1 \leq x < p - 1, \quad \left( \frac{x}{p} \right) = 1 \quad \text{and} \quad \left( \frac{x+1}{p} \right) = -1. \]

The algorithm for doing this is given below (in the algorithm, legendre[x,p] means \( \left( \frac{x}{p} \right) \));

```
procedure FindX(left,right)
{
    if(right-left=1)
        return left;
    mid=Floor[(left+right)/2];
    if(legendre[mid,p]=1)
        return FindX(mid,right);
    else
        return FindX(left,mid);
}
```

We use \( \text{FindX}(1,p-1) \) to call the program and get \( x \).

Use power algorithm and \( \left( \frac{x}{p} \right) = x^{(p-1)/2} \mod p \), the time complexity to get \( \left( \frac{x}{p} \right) \) is \( O((\log p)^3) \) bit operation. Due to binary search, there will be \( \log p \) such operations. After considering all the other operations, the complexity to find this \( x \) using above algorithm is \( (\log p)^4 \).
Using the above $x$, we can construct a non-square element in $\mathbb{F}_{p^2}$. The fact that \( \left( \frac{x}{p} \right) = 1 \) and

\[
\left( \frac{-x}{p} \right) = \left( \frac{x}{p} \right) \left( \frac{-1}{p} \right)
= (-1)(-1)
= 1
\]
tell us that both $x$ and $-(x + 1)$ have root in $\mathbb{F}_p$. Noting that $p = 3 \mod 4$, from Corollary 7.1.2, we have

\[
u \equiv \sqrt{x} \\
= x^{(p+1)/4}
\]

and

\[
v \equiv \sqrt{-(x + 1)} \\
= -(x + 1)^{(p+1)/4},
\]
and these $u$ and $v$ can be computed using $O((\lg p)^3)$ bit operation.

Now, the element $u + vi$ must be a non-square element in $\mathbb{F}_{p^2}$. Otherwise, there exists $a + bi$ with $a, b \in \mathbb{F}_p$ such that

\[
u + iv = (a + ib)^2 \\
= a^2 - b^2 + 2abi,
\]
which implies $u = a^2 - b^2$, $v = 2ab$, and hence

\[
u^2 + v^2 = (a^2 - b^2)^2 + (2ab)^2 \\
= (a^2 + b^2)^2.
\]

On the other hand, from $u^2 = x$, $v^2 = -(x + 1)$ we see that $u^2 + v^2 = -1$. This together with above equation implies $(a^2 + b^2)^2 = -1$. Since $a^2 + b^2 \in \mathbb{F}_p$, we get \( \left( \frac{-1}{p} \right) = 1 \). This is in contradiction with \( \left( \frac{-1}{p} \right) = -1 \).

Any square roots in $\mathbb{F}_{p^2}$ can be computed using Tonelli’s algorithm. Tonelli’s algorithm is nondeterministic only because it randomly chooses an element $g \in \mathbb{F}_{p^2}$ and hope it is not a square (and thus it will be a generator). Now that we have found a non-square element $u + vi$ in $\mathbb{F}_{p^2}$ using above procedure, we can use this $u + vi$ as $g$ in Tonelli’s algorithm. In this situation, Tonelli’s algorithm would become deterministic.

The time complexity for computing $u + vi$ is $O((\lg p)^4)$ bit operation. The running time for the modified Tonelli’s algorithm is also $O((\lg p)^4)$ bit operation (cf. Theorem 7.1.3). So, the total running time for computing square roots in $\mathbb{F}_{p^2}$ using this method is $O((\lg p)^4)$ bit operation. So, it is deterministic polynomial time.