Project 2
Number Theoretic Transform

KG

January 23, 2023

1 Introduction

In this project, we will examine the *Number Theoretic Transform* and its use in speeding up arithmetic in some rings. We want to find a ring isomorphism $\phi : \mathbb{F}[X]/(X^{2^m}+1) \to \mathbb{F}^{2^m}$. We have a primitive 2^{m+1}th root of unity ω. This isomorphism can be computed by an iterative process, which we describe briefly.

First, Consider a ring of the form $R_0 = \mathbb{F}[X]/(X^{2u} - v^2)$, which by the Chinese remainder theorem is isomorphic to $R_1 \times R_2$, with $R_1 = \mathbb{F}[X]/(X^u - v)$ and $R_2 = \mathbb{F}[X]/(X^u + v)$. The CRT isomorphism maps the coset in R_0 represented by $f(X)$ to the cosets of R_1 and R_2 represented by the same polynomial $f(X)$. We divide by $X^u - v$ and $X^u + v$, respectively, to find representatives of minimal degree.

Given a polynomial $\sum_{i=0}^{2u-1} f_i X^i$, the remainder when dividing by $X^u - v$ is $\sum_{i=0}^{u-1}(f_i + v f_{i+u})X^i$, since $X^u = v$ in R_1. Likewise, the remainder when dividing by $X^u + v$ is $\sum_{i=0}^{u-1}(f_i - v f_{i+u})X^i$, since $X^u = -v$ in R_2.

This construction can be repeated many times. If we begin with $X^{2^m}+1$ and a 2^{m+1}th root of 1, we need m iterations to get to linear factors. Finally, $\mathbb{F}[X]/(X-v)$ is trivially isomorphic to $\mathbb{F}$.

Let ω be a primitive 2^{m+1}th root of 1 in $\mathbb{F}$. Then $\omega^{2^m} = -1$. Suppose the polynomial is $X^{2^j} - v^2$ with $v^2 = \omega^i$. Then we choose $v = \omega^{i/2}$ and $-v = \omega^{i/2+2^{m-1}}$. For the first split of $X^{2^m}+1$, $v = \omega^{2^{m-1}}$.

Computing the inverse of this isomorphism is also fast, up to a power of 2. If $g = f_i - v f_{i+u}$, and $h = f_i + v f_{i+u}$, then $2f_i = g + h$ and $2f_{i+u} = (h-g)v^{-1}$.

It may be useful to look at Section 7 of Bernstein [1].

Examples: For simplicity, we only list the representatives and skip the coset notation.

1. Multiply $1+2X$ and $3+4X$ in $R_0 = \mathbb{F}_p[X]/(X^2+1)$, $p = 257$. Let $m = 1$, $\omega = v = 2^4$, and note that $X^2+1 = (X-16)(X+16)$ in $\mathbb{F}_p[X]$.

   $1+2X \mapsto (33, -31)$.

   $3+4X \mapsto (67, -61)$.

   $(1+2X)(3+4X) = 3+6X+4X+8X^2 = -5+10X \mapsto (155, -165)$.

2. Multiply $1+2X+3X^2+4X^3$ and $5+6X+7X^2+8X^3$ in $R_0 = \mathbb{F}_p[X]/(X^4+1)$, $p = 257$.

   Let $m = 2$, $\omega = 2^2$.

   - $v = 2^4$, $X^4+1 = (X^2-16)(X^2+16)$.
     $1+2X+3X^2+4X^3 \mapsto (49+66X, -47-62X)$
     $5+6X+7X^2+8X^3 \mapsto (117+134X, -107-122X)$
   - $v = 2^2$, $X^2-16 = (X-4)(X+4)$.
     $49+66X \mapsto (56, 42)$
     $117+134X \mapsto (139, 95)$
   - $v^2 = -2^4 = 2^{10} \Rightarrow v = 2^5$, $X^2+16 = (X-64)(X+64)$
     $210+195X \mapsto (97, 66)$
     $150+135X \mapsto (52, 248)$

2 Tasks

You will write a report using LaTeX. The report should include an explanation of what you have done, a description of your implementation, the theoretical analysis, any experimental results with explanations and a code listing.

You do not need to report on the results of the first task.

1. Warm-up Do these multiplications:

- $R_0 = \mathbb{F}_p[X]/(X^8+1)$, $p = 257$, $m = 3$, $\omega = 3^{2^4} = 249$.
  $(\sum_{i=0}^{7}(i+1)X^i)(\sum_{i=0}^{7}(i+9)X^i)$
- $R_0 = \mathbb{F}_p[X]/(X^{2^7}+1$, $p = 257$, $m = 7$, $\omega = 3$.
  $(\sum_{i=0}^{127}(i+1)X^i)(\sum_{i=0}^{127}(i+9)X^i)$
- $R_0 = \mathbb{F}_p[X]/(X^{2^8}+1)$, $p = (15\cdot 2^9+1) = 7681$, $m = 8$, $\omega = 4055$.
  $(\sum_{i=0}^{255}(i+1)X^i)(\sum_{i=0}^{255}(i+9)X^i)$

The first could be done by hand. The next two probably needs computer help.

2. Theory

a. Correctness Show that the map ϕ sketched in the introduction is a ring isomorphism.

b. Optional: Discrete Fourier Transform (DFT) Explain what the DFT is. Explain how the isomorphism ϕ is related to the DFT. Also explain the connection between ϕ and the Fast Fourier Transform (FFT).

3. Arithmetic

a. Implement: Naive arithmetic Implement naive arithmetic in $\mathbb{F}_p[X]/(X^n+1)$, in particular naive (schoolbook) multiplication of elements. You may either do a straight-forward polynomial multiplication followed by polynomial remainder, or you can exploit the specific form of the polynomial X^n+1 to use a more efficient formula for multiplication. Also implement unit tests.

b. Implement: NTT Implement the isomorphism $\phi : \mathbb{F}_p[X]/(X^{2^m}+1) \to \mathbb{F}_p^{2^m}$ described in the introduction, as well as its inverse.

Implement multiplication of elements in $\mathbb{F}_p[X]/(X^{2^m}+1)$ by applying the isomorphism ϕ, multiplying element-wise in $\mathbb{F}_p^{2^m}$ and then applying the inverse isomorphism ϕ^{-1}.

Also implement unit tests.

c. Analyse Explain the theoretical cost of the two multiplication algorithms you have implemented, in terms of multiplications in $\mathbb{F}_p$.

d. Run Find a suitable sequence of primes p and use them to time the cost of multiplications in $\mathbb{F}_p[X]/(X^{2^m}+1)$ using both the naive multiplication implementation from Task 3a and Task 3b. If your programming environment has native polynomial arithmetic, also time native polynomial multiplication.

Compare the results with the theoretical analysis from Task 3c.

References

[1] Daniel J. Bernstein. Multidigit multiplication for mathematicians. `https://cr.yp.to/papers.html#m3`, 2001.