CS 70 Discussion 5A

October 2, 2024

Polynomials

Coefficient Representation: n values $c_0, c_1, ..., c_{n-1} \in \mathbb{R}$ define a *unique* degree $\leq n-1$ polynomial:

$$f(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1}$$

Point Representation: n *distinct* points $(a_0, b_0), (a_1, b_1), \ldots, (a_{n-1}, b_{n-1}) \in \mathbb{R}^2$ define a *unique* degree $\leq n-1$ polynomial.

Lagrange Interpolation

Goal: Given n distinct points $(a_0, b_0), (a_1, b_1), ..., (a_{n-1}, b_{n-1}) \in \mathbb{R}^2$, how do we generate our unique degree $\leq n-1$ polynomial $f(x)$?
Solution:

$$f(x) = \sum_{i=0}^{n-1} b_i \Delta_i(x)$$
$$\Delta_i(x) = \frac{\prod_{j \neq i}(x - a_j)}{\prod_{j \neq i}(a_i - a_j)}$$

$\Delta_i(x)$ is often referred-to as a *delta polynomial*. Also, note our construction for a polynomial is similar to the construction of the unique solution x for CRT.

Secret Sharing

Goal: We have a secret. We want a group of n people to only be able to learn the secret if at least m people agree to unlock the secret.

Solution:

1. Generate a random polynomial of degree $m - 1$ where $f(0)$ is our secret.
2. Pick n distinct points on our polynomial where each x-coordinate is > 0.
3. Give each person a unique one of the n points.
4. Two cases:
   - $\geq m$ people agree: m points uniquely define a degree $\leq m - 1$ polynomial, so construct $f(x)$ with Lagrange interpolation and get the secret at $f(0)$.
   - $< m$ people agree: We need at least m points to retrieve our polynomial, or else we still can't narrow-down to exactly 1 unique polynomial (i.e. we still have infinite possibilities for $f(0)$).

Galois Field

Problem: We have numerical instability with all our arithmetic operations during Lagrange interpolation (floating-point operations are imprecise on computers).
Solution: We use modular arithmetic. We define our polynomial within GF(p), where p is some large prime. Simply, this means that every arithmetic operation in this space is done $\bmod p$:

- $a + b = c \rightarrow a + b \equiv d \pmod{p}$
- $a - b = c \rightarrow a - b \equiv d \pmod{p}$
- $ab = c \rightarrow ab \equiv d \pmod{p}$
- $\frac{a}{b} = c \rightarrow ab^{-1} \equiv d \pmod{p}$
    - b^{-1} is modular inverse of $b \pmod{p}$

Now, we don't need to worry about floating-point numbers. Also, Lagrange interpolation works fine in GF(p), as we only perform additions, subtractions, multiplications, and divisions to construct our polynomial.