CS70 Polynomials

Kelvin Lee

kelvinlee@berkeley.edu

September 30, 2020

Overview

1. Polynomials
2. Lagrange Interpolation
3. Finite Fields
4. Secret Sharing

Polynomials

Polynomials

Properies of polynomials:

Polynomials

Properies of polynomials:

- **Property 1:**

Polynomials

Properies of polynomials:

- **Property 1:** A non-zero polynomial of degree d has at most d roots.

Polynomials

Properies of polynomials:

- **Property 1:** A non-zero polynomial of degree d has at most d roots.
- **Property 2:**

Properies of polynomials:

- **Property 1:** A non-zero polynomial of degree d has at most d roots.
- **Property 2:** A polynomial of degree d is **uniquely** determined by $d + 1$ distinct points.

Polynomial Interpolation

Polynomial Interpolation

Given $d + 1$ distinct points, how do we determine the polynomial?

Polynomial Interpolation

Given $d+1$ distinct points, how do we determine the polynomial?

- We use a method called **Lagrange Interpolation**, which works similarly to the **Chinese Remainder Theorem**.

Polynomial Interpolation

Given $d+1$ distinct points, how do we determine the polynomial?

- We use a method called **Lagrange Interpolation**, which works similarly to the **Chinese Remainder Theorem**.
- Suppose the given points are $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$. We want to find a polynomial $p(x)$ such that $p(x_i) = y_i$ for $i = 1, \ldots, d+1$.

Polynomial Interpolation

Given $d+1$ distinct points, how do we determine the polynomial?

- We use a method called **Lagrange Interpolation**, which works similarly to the **Chinese Remainder Theorem**.
- Suppose the given points are $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$. We want to find a polynomial $p(x)$ such that $p(x_i) = y_i$ for $i = 1, \ldots, d+1$.
- In other words, we want to find polynomials $p_1(x), \ldots, p_{d+1}(x)$ such that

Polynomial Interpolation

Given $d+1$ distinct points, how do we determine the polynomial?

- We use a method called **Lagrange Interpolation**, which works similarly to the **Chinese Remainder Theorem**.
- Suppose the given points are $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$. We want to find a polynomial $p(x)$ such that $p(x_i) = y_i$ for $i = 1, \ldots, d+1$.
- In other words, we want to find polynomials $p_1(x), \ldots, p_{d+1}(x)$ such that

$$p_1(x) = 1 \text{ at } x_1 \text{ and } p_1(x) = 0 \text{ at } x_2, \ldots, x_{d+1};$$

Polynomial Interpolation

Given $d+1$ distinct points, how do we determine the polynomial?

- We use a method called **Lagrange Interpolation**, which works similarly to the **Chinese Remainder Theorem**.
- Suppose the given points are $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$. We want to find a polynomial $p(x)$ such that $p(x_i) = y_i$ for $i = 1, \ldots, d+1$.
- In other words, we want to find polynomials $p_1(x), \ldots, p_{d+1}(x)$ such that

$$p_1(x) = 1 \text{ at } x_1 \text{ and } p_1(x) = 0 \text{ at } x_2, \ldots, x_{d+1};$$

$$p_2(x) = 1 \text{ at } x_2 \text{ and } p_2(x) = 0 \text{ at } x_1, x_3 \ldots, x_{d+1};$$

Polynomial Interpolation

Given $d+1$ distinct points, how do we determine the polynomial?

- We use a method called **Lagrange Interpolation**, which works similarly to the **Chinese Remainder Theorem**.
- Suppose the given points are $(x_1, y_1), \ldots, (x_{d+1}, y_{d+1})$. We want to find a polynomial $p(x)$ such that $p(x_i) = y_i$ for $i = 1, \ldots, d+1$.
- In other words, we want to find polynomials $p_1(x), \ldots, p_{d+1}(x)$ such that

$$p_1(x) = 1 \text{ at } x_1 \text{ and } p_1(x) = 0 \text{ at } x_2, \ldots, x_{d+1};$$

$$p_2(x) = 1 \text{ at } x_2 \text{ and } p_2(x) = 0 \text{ at } x_1, x_3 \ldots, x_{d+1};$$

$$p_3(x) = 1 \text{ at } x_3 \text{ and } p_3(x) = 0 \text{ at } x_1, x_2, x_4, \ldots, x_{d+1} \text{ and so on...}$$

Lagrange Interpolation

Lagrange Interpolation

Continued:

Lagrange Interpolation

Continued:

- Let's start by finding $p_1(x)$.

Lagrange Interpolation

Continued:

- Let's start by finding $p_1(x)$.
- Since $p_1(x) = 0$ at $x_2, \ldots, x_{d+1}$, $p_1(x)$ must be a multiple of

$$q_1(x) = (x - x_2)(x - x_3)\ldots(x - x_{d+1}).$$

Lagrange Interpolation

Continued:

- Let's start by finding $p_1(x)$.
- Since $p_1(x) = 0$ at $x_2, \ldots, x_{d+1}$, $p_1(x)$ must be a multiple of

$$q_1(x) = (x - x_2)(x - x_3) \ldots (x - x_{d+1}).$$

- We also need $p_1(x) = 1$ at x_1. Notice that

$$q_1(x_1) = (x_1 - x_2)(x_1 - x_3) \ldots (x - x_{d+1}).$$

Lagrange Interpolation

Continued:

- Let's start by finding $p_1(x)$.
- Since $p_1(x) = 0$ at $x_2, \ldots, x_{d+1}$, $p_1(x)$ must be a multiple of

$$q_1(x) = (x - x_2)(x - x_3) \ldots (x - x_{d+1}).$$

- We also need $p_1(x) = 1$ at x_1. Notice that

$$q_1(x_1) = (x_1 - x_2)(x_1 - x_3) \ldots (x - x_{d+1}).$$

- Then $p_1(x) = \frac{q(x)}{q_1(x_1)}$ is the polynomial we are looking for.

Continued:

- Let's start by finding $p_1(x)$.
- Since $p_1(x) = 0$ at $x_2, \ldots, x_{d+1}$, $p_1(x)$ must be a multiple of

$$q_1(x) = (x - x_2)(x - x_3) \ldots (x - x_{d+1}).$$

- We also need $p_1(x) = 1$ at x_1. Notice that

$$q_1(x_1) = (x_1 - x_2)(x_1 - x_3) \ldots (x - x_{d+1}).$$

- Then $p_1(x) = \frac{q(x)}{q_1(x_1)}$ is the polynomial we are looking for.
- Similarly for $p_i(x)$, we have $p_i(x) = \frac{q(x)}{q_i(x_i)}$.

Lagrange Interpolation

Lagrange Interpolation

- After finding $p_1(x), \ldots, p_{d+1}(x)$, we can construct $p(x)$ by scaling up each bit by corresponding y_i:

Lagrange Interpolation

- After finding $p_1(x), \ldots, p_{d+1}(x)$, we can construct $p(x)$ by scaling up each bit by corresponding y_i:

$$p(x) = \sum_{i=1}^{d+1} y_i p_i(x)$$

Lagrange Interpolation

- After finding $p_1(x), \ldots, p_{d+1}(x)$, we can construct $p(x)$ by scaling up each bit by corresponding y_i:
$$p(x) = \sum_{i=1}^{d+1} y_i p_i(x)$$

  Does this remind you of CRT?

Lagrange Interpolation

- After finding $p_1(x), \ldots, p_{d+1}(x)$, we can construct $p(x)$ by scaling up each bit by corresponding y_i:

$$p(x) = \sum_{i=1}^{d+1} y_i p_i(x)$$

  Does this remind you of CRT?
- Now let us define $\Delta_i(x)$ in the following way (think of them as a basis):

Lagrange Interpolation

- After finding $p_1(x), \ldots, p_{d+1}(x)$, we can construct $p(x)$ by scaling up each bit by corresponding y_i:
$$p(x) = \sum_{i=1}^{d+1} y_i p_i(x)$$
Does this remind you of CRT?
- Now let us define $\Delta_i(x)$ in the following way (think of them as a basis):
$$\Delta_i(x) = \frac{\prod_{i \neq j}(x - x_j)}{\prod_{i \neq j}(x_i - x_j)}.$$

Lagrange Interpolation

- After finding $p_1(x), \dots, p_{d+1}(x)$, we can construct $p(x)$ by scaling up each bit by corresponding y_i:

$$p(x) = \sum_{i=1}^{d+1} y_i p_i(x)$$

  Does this remind you of CRT?
- Now let us define $\Delta_i(x)$ in the following way (think of them as a basis):

$$\Delta_i(x) = \frac{\prod_{i \neq j}(x - x_j)}{\prod_{i \neq j}(x_i - x_j)}.$$

- Then we have an **unique** polynomial

Lagrange Interpolation

- After finding $p_1(x), \ldots, p_{d+1}(x)$, we can construct $p(x)$ by scaling up each bit by corresponding y_i:
$$p(x) = \sum_{i=1}^{d+1} y_i p_i(x)$$

  Does this remind you of CRT?
- Now let us define $\Delta_i(x)$ in the following way (think of them as a basis):
$$\Delta_i(x) = \frac{\prod_{i \neq j}(x - x_j)}{\prod_{i \neq j}(x_i - x_j)}.$$
- Then we have an **unique** polynomial
$$\boxed{p(x) = \sum_{i=1}^{d+1} y_i \Delta_i(x).}$$

Finite Fields

Finite Fields

- The properties of a polynomial would not hold if the values are restricted to being natural numbers or integers because dividing two integers does not generally result in an integer.

Finite Fields

- The properties of a polynomial would not hold if the values are restricted to being natural numbers or integers because dividing two integers does not generally result in an integer.
- However, if we work with numbers modulo m where m is a prime number, then we can add, subtract, multiply and divide.

Finite Fields

- The properties of a polynomial would not hold if the values are restricted to being natural numbers or integers because dividing two integers does not generally result in an integer.
- However, if we work with numbers modulo m where m is a prime number, then we can add, subtract, multiply and divide.
- Then **Property 1** and **Property 2** hold if the coefficients and the variable x are restricted to take on values modulo m. When we work with numbers modulo m, we are working over a **finite field**, denoted by $GF(m)$ (**Galois Field**).

Secret Sharing

Secret Sharing

Basic Setting:

Secret Sharing

Basic Setting:

- Suppose there are n people. Let s be the secret number and q be a prime number greater than n and s. We will work over $GF(q)$.

Secret Sharing

Basic Setting:

- Suppose there are n people. Let s be the secret number and q be a prime number greater than n and s. We will work over $GF(q)$.
- Pick a random polynomial $P(x)$ of degree $k-1$ such that $P(0) = s$.

Secret Sharing

Basic Setting:

- Suppose there are n people. Let s be the secret number and q be a prime number greater than n and s. We will work over $GF(q)$.
- Pick a random polynomial $P(x)$ of degree $k-1$ such that $P(0) = s$.
- Distribute $P(1), \ldots P(n)$ to each person so that each one receives one value.

Secret Sharing

Basic Setting:

- Suppose there are n people. Let s be the secret number and q be a prime number greater than n and s. We will work over $GF(q)$.
- Pick a random polynomial $P(x)$ of degree $k-1$ such that $P(0)=s$.
- Distribute $P(1),\ldots P(n)$ to each person so that each one receives one value.
- Then in order to know what s is, at least k of the n people must work together so that they can perform **Lagrange interpolation** and find P.

Secret Sharing

Basic Setting:

- Suppose there are n people. Let s be the secret number and q be a prime number greater than n and s. We will work over $GF(q)$.
- Pick a random polynomial $P(x)$ of degree $k-1$ such that $P(0) = s$.
- Distribute $P(1), \ldots P(n)$ to each person so that each one receives one value.
- Then in order to know what s is, at least k of the n people must work together so that they can perform **Lagrange interpolation** and find P.
- If there are less than k people, they will learn nothing about s!

Problem Time!