Introduction to Algorithms

Mong-Jen Kao (高孟駿)

Tuesday 10:10 – 12:00

Thursday 15:30 – 16:20

Divide-and-Conquer

– More Examples

More on recursion for problem solving.

Example 5.

Fast Fourier Transform (FFT)

Fast Conversion

between *coefficient representation* and *point-value representation* of a polynomial.

Coefficient Representation of Polynomials

- Traditionally,
  we represent a polynomial by the ***coefficient*** *of its monomials*.
  - Ex. $f = (a_0, a_1, \ldots, a_n)$ for a degree n polynomial

$$f(x) = \sum_{0 \le i \le n} a_i \cdot x^i \ .$$

- In this way, for any two degree n polynomials $f(x)$ and $g(x)$,
  - $f(x) + g(x)$ can be done in $O(n)$ time.
  - $f(x) \cdot g(x)$ can be done in $O\left(n^2\right)$ time.

(Complex-) Root Representation

- It is well-known that, for a degree n polynomial $f(x)$,
  - if $r_1, r_2, \dots, r_k$ are all of its *(potentially be complex)* roots and
  - $q_1, q_2, \dots, q_k$ are the corresponding multiplicities of the roots,

  then $f(x)$ can be uniquely represented as $f(x) = \prod_{1 \le i \le k} (x - r_i)^{q_i}$.

- In this way, $f(x) \cdot g(x)$ can be done in $O(n)$ time.

- However, for $n \geq 5$, there is no general way for computing the roots of a degree-n polynomial.

By the well-known Galois Theorem.

Point-Value Representation

- The following theorem states that, in general, evaluations for n distinct input values also uniquely define a degree-n polynomial.

<u>Theorem 1. (Uniqueness of an Interpolating Polynomial)</u>

For any set of n point-value pairs $\{ (x_1, y_1), \dots, (x_n, y_n) \}$
such that $x_i \neq x_j$ for any $1 \leq i \neq j \leq n$,
there is a unique polynomial $A(x)$ of degree at most n such that
$y_k = A(x_k)$ for all $1 \leq k \leq n$.

Given $\boldsymbol{n}$ ***point-value pairs***, the degree-n polynomial is ***<u>uniquely determined</u>***.

Proof of Theorem 1

- The evaluation of the point-value pairs $(x_1, y_1), \dots, (x_n, y_n)$ is equivalent to the following matrix operation.

$$\underbrace{\boxed{\begin{pmatrix} 1 & x_1 & \cdots & x_1^n \\ 1 & x_2 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^n \end{pmatrix}}}_{V(x_1, \dots, x_n)} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

non-zero when $x_k \neq x_j$.

- $V(x_1, \dots, x_n)$ has determinant $\prod_{1 \le j < k \le n} (x_k - x_j)$.

To be proved in HW3.

$V(x_1, \dots, x_n)$ is invertible.

Point-Value Representation

- The following theorem states that, in general, evaluations for n distinct input values also uniquely defines a degree-n polynomial.

- In this way, for any two degree n polynomials $f(x)$ and $g(x)$
  - $f(x) + g(x)$ can be done in $O(n)$ time.
  - $f(x) \cdot g(x)$ can be done in $O(n)$ time.

- For the conversion between coefficient representation and point-value representation,
  - The naïve approach takes $\Theta(n^2)$.

Fast Fourier Transform (FFT)

- For the conversion between coefficient representation and point-value representation,
  - The naïve approach takes $\Theta(n^2)$.
  - If we *choose* $x_1, \dots, x_n$ *wisely*,
    the conversion can be done in $\boldsymbol{O(n \log n)}$ **time**!

- For computing $C(x) = A(x) \cdot B(x)$,
  where $A = (a_0, \dots, a_n)$, $B = (b_0, \dots, b_n)$, and $C = (c_0, c_1, \dots, c_{2n})$.
  - We will choose $x_j \coloneqq \omega_{2n}^{j}$ for all $0 \leq j < 2n$.

j-th *complex root of unity*.

- **For computing $\boldsymbol{C(x) = A(x) \cdot B(x)}$,**
  where $A = (a_0, \dots, a_n)$, $B = (b_0, \dots, b_n)$, and $C = (c_0, c_1, \dots, c_{2n})$.
  - Choose $x_j \coloneqq \omega_{2n}^{j}$ for all $0 \le j < 2n$.

$$A = (a_0, a_1, \dots, a_n)$$
$$B = (b_0, b_1, \dots, b_n)$$

Discrete Fourier Transform (DFT)
(Evaluation) in $\boldsymbol{\Theta(n \log n)}$ time.

$$A(\omega_{2n}^{0}), A(\omega_{2n}^{1}), \dots, A(\omega_{2n}^{2n-1})$$
$$B(\omega_{2n}^{0}), B(\omega_{2n}^{1}), \dots, B(\omega_{2n}^{2n-1})$$

Pointwise multiplication in $\boldsymbol{\Theta(n)}$ time.

$$C(\omega_{2n}^{0}), C(\omega_{2n}^{1}), \dots, C(\omega_{2n}^{2n-1})$$

Inverse DFT
(Interpolation) in $\boldsymbol{\Theta(n \log n)}$.

$$C = (c_0, c_1, \dots, c_{2n})$$

Complex Roots of Unity

- The Euler's formula states that

$$e^{i\cdot\theta} \;=\; \cos\theta + i\cdot\sin\theta$$

for any $\theta \in \mathbb{R}$, where i is the *imaginary unit* with $i^2 = -1$.

  – The formula can be proved by Taylor's expansion or by solving the differential equation

$$f(x) \;=\; -i\cdot\frac{\mathrm{d}f(x)}{\mathrm{d}x}$$

with $f(x) = \cos x + i\cdot\sin x$ and the boundary condition $f(0) = 1$.

Complex Roots of Unity

This formula is *what makes all the magic happen*.

- The Euler's formula states that, for any $\theta \in \mathbb{R}$,

$$e^{i\cdot\theta} = \cos\theta + i\cdot\sin\theta \ .$$

- By taking $\omega_n^j \coloneqq e^{2\pi j\cdot i/n}$,

  we know that $\omega_n^0, \omega_n^1, \dots, \omega_n^{n-1}$ are the n ***distinct roots*** for $x^n = 1$.

- By the above definitions, for any $j, k \in \mathbb{Z}_{\geq 0}$, we have

$$\omega_n^j \cdot \omega_n^k = \omega_n^{j+k}, \qquad \text{and} \left(\omega_n^j\right)^k = \omega_n^{j\cdot k} \ .$$

Complex Roots of Unity

This formula is *what makes all the magic happen*.

- The Euler's formula states that, for any $\theta \in \mathbb{R}$,

$$e^{i \cdot \theta} = \cos\theta + i \cdot \sin\theta \ .$$

- By taking $\omega_n^j := e^{2\pi j \cdot i/n}$,
  we know that $\omega_n^0, \omega_n^1, \dots, \omega_n^{n-1}$ are the n ***distinct roots*** for $x^n = 1$.

- In FFT, we will use $\left(\omega_n^j,\ f\left(\omega_n^j\right)\right)$ for all $0 \le j < n$ to be the *point-value representation* for any degree-n polynomial $f(x)$.

The Discrete Fourier Transform (DFT) Problem

Given $f \coloneqq (a_0, a_1, \ldots, a_{n-1})$, where n is a power of 2, compute

$$f\left(\omega_n^j\right) = \sum_{0 \le k < n} a_k \cdot \omega_n^{k \cdot j} \quad \text{for all } 0 \le j < n.$$

- Let $f^{[0]} \coloneqq (a_0, a_2, \ldots, a_{n-2})$ and $f^{[1]} \coloneqq (a_1, a_3, \ldots, a_{n-1})$ be two degree-$n/2$ polynomials formed by the even-indexed coefficients and the odd-indexed coefficients of f, respectively.
  - We have $f(x) = f^{[0]}\left(x^2\right) + x \cdot f^{[1]}\left(x^2\right)$.

Solve them recursively and then merge the result.

Two Properties

$$\omega_n^i := e^{\frac{2\pi}{n}i} = \cos\frac{2\pi}{n} + i \cdot \sin\frac{2\pi}{n}\ .$$

1. For any $n = 2^k \geq 2$ and any $0 \leq j < n/2$, we have

$$\left(\omega_n^j\right)^2 = \omega_n^{2j} = \omega_{n/2}^j\ .$$

   – Verifiable by the Euler's formula.

Used in the ***recursive DFT*** problem.

2. For any $n = 2^k \geq 2$ and any $0 \leq j < n/2$, we have

$$\omega_n^{j+n/2} = -\omega_n^j\ .$$

   – Verifiable by the fact that $\omega_n^{n/2} = -1$.

Used when ***merging*** the result.

- Recursive-FFT$(a_0, a_1, \ldots, a_{n-1})$ with $n = 2^k \geq 1$.

A. If $n = 1$, then return $\{a_0\}$.

B. Let $\omega \leftarrow 1$, $A^{[0]} = (a_0, a_2, \ldots, a_{n-2})$, and $A^{[1]} = (a_1, a_3, \ldots, a_{n-1})$.

C. $y^{[0]} \leftarrow$Recursive-FFT$\left(A^{[0]}\right)$.
   $y^{[1]} \leftarrow$Recursive-FFT$\left(A^{[1]}\right)$.

D. For $k \leftarrow 0$ to $n/2 - 1$, do the following.
   - $y_k = y_k^{[0]} + \omega \cdot y_k^{[1]}$.
   - $y_{k+n/2} = y_k^{[0]} - \omega \cdot y_k^{[1]}$.
   - $\omega \leftarrow \omega \cdot \omega_n^1$.

E. Return y.

The <u>*Fast Fourier Transform (FFT) algorithm*</u> for the DFT Problem.

Interpolation at the Complex Roots of Unity

- Given $f\left(\omega_n^i\right)$ for all $0 \le i < n$, compute $a_0, a_1, \dots, a_{n-1}$ such that

$$f(x) = \sum_{0 \le j < n} a_j \cdot x^j \ .$$

- Recall that

$$\underbrace{\boxed{\begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \omega_n^3 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \cdots & \omega_n^{2(n-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \cdots & \omega_n^{3(n-1)} \\ 1 & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{pmatrix}}}_{V_n} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}.$$

Theorem 2.

For any $0 \le j, k < n$, the (j, k) entry of V_n^{-1} is ω_n^{-kj}/n.

- The (j, j') entry of $V_n^{-1} \cdot V_n$ is

$$[V_n^{-1} \cdot V_n]_{j,j'} = \sum_{0 \le k < n} \left(\omega_n^{-kj} / n\right) \cdot \omega_n^{kj'}$$

$$= \sum_{0 \le k < n} \omega_n^{k(j'-j)} / n \ .$$

- The summation is 1 if $j = j'$ and 0 otherwise.

  Hence $V_n^{-1} \cdot V_n = I_n$.

Interpolation at the Complex Roots of Unity

- Given $f\left(\omega_n^i\right)$ for all $0 \leq i < n$, compute $a_0, a_1, \ldots, a_{n-1}$ such that

$$f(x) = \sum_{0 \leq j < n} a_j \cdot x^j \ .$$

- By Theorem 2, for any $0 \leq j < n$, we have

$$a_j = \frac{1}{n} \cdot \sum_{0 \leq k < n} y_k \cdot \omega_n^{-kj} \ .$$

- This is ***exactly the DFT problem*** if we switch the roles of $(a_0, \ldots, a_{n-1})$ and $(y_0, \ldots, y_{n-1})$, replace ω_n by ω_n^{-1}, and divide the result by n.

Fast Fourier Transform (FFT)

$$A = (a_0, a_1, \dots, a_n)$$
$$B = (b_0, b_1, \dots, b_n)$$

Discrete Fourier Transform (DFT)
(Evaluation) in $\boldsymbol{\Theta(n \log n)}$ time.

$$A(\omega_{2n}^{0}), A(\omega_{2n}^{1}), \dots, A(\omega_{2n}^{2n-1})$$
$$B(\omega_{2n}^{0}), B(\omega_{2n}^{1}), \dots, B(\omega_{2n}^{2n-1})$$

Pointwise
multiplication
in $\boldsymbol{\Theta(n)}$ time.

$$C(\omega_{2n}^{0}), C(\omega_{2n}^{1}), \dots, C(\omega_{2n}^{2n-1})$$

Inverse DFT
(Interpolation)
in $\boldsymbol{\Theta(n \log n)}$.

$$C = (c_0, c_1, \dots, c_{2n})$$