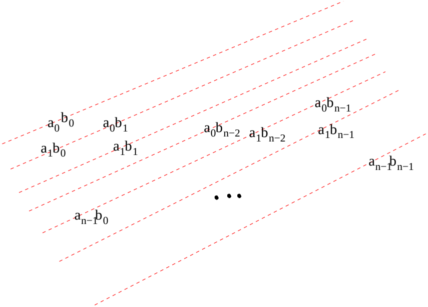
Paradigm: Divide-and-Conquer

Polynomial Multiplication: FFT

R. Inkulu
http://www.iitg.ac.in/rinkulu/

Description

Given two polynomials $A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$, and $B(x) = b_0 + b_1 x + b_2 x^2 + \ldots + b_{n-1} x^{n-1}$, computing $C(x) = \sum_{j=0}^{2n-2} c_j x^j$, where $c_j = \sum_{k=0}^{j} a_k b_{j-k}$.



summing along the diagonals yield the required c_js $\rightarrow$ takes $O(n^2)$ time.

Polynomial representations

- The *coefficient representation* of a polynomial $A(x) = \sum_{j=0}^{n-1} a_j x^j$ is a vector of coefficents $(a_0, a_1, \ldots, a_{n-1})$.
- A *set of points representation* of a polynomial $A(x)$ of degree $n-1$ is a set of n point-value pairs $\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$ such that all the x_k are distinct and $y_k = A(x_k)$ for $k = 0, 1, \ldots, n-1$.

Equivalence between polynomial representations

- coefficient $\rightarrow$ set of points

  Horner's rule: $A(x') = a_0 + x'(a_1 + x'(a_2 + \ldots + (x'(a_{n-2} + x'(a_{n-1})))))$.

- set of points $\rightarrow$ coefficient

  interpolating the polynomial with the help of matrix algebra

Naive Algorithm

(1) Since $C(x)$ is of degree $2n-2$, we require at least $2n-1$ points to recover $C(x)$. Choose $x_1, x_2, \ldots, x_{2n}$ and evaluate $A(x)$ and $B(x)$ at these points.

$O(n^2)$ time using Horner's rule

(2) Compute $C(x_j) = A(x_j)B(x_j)$ for $j = 1, 2, \ldots, 2n-1$.

$O(n)$ time

(3) Reconstructing $C(x)$ of degree $2n-2$ from $C(x_j)$ for $j = 1, 2, \ldots, 2n-1$.

$O(n^3)$ time using techniques from Linear Algebra

Handling (1): Divide-Conquer-Combine

- $A(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1}$

  $= (a_0 + a_2 x^2 + \ldots + a_{n-2} x^{n-2}) + x(a_1 + a_3 x^2 + \ldots + a_{n-1} x^{n-2})$

  $= (a_0 + a_2(x^2) + \ldots + a_{n-2}(x^2)^{(n-2)/2})$
  $+ x(a_1 + a_3(x^2) + \ldots + a_{n-1}(x^2)^{(n-2)/2})$

  $= A_{even}(x^2) + xA_{odd}(x^2)$

- Let $T(n)$ be the number of operations required to evaluate $A(x)$ of degree $n-1$ at $2n$ points.

  Then $T(n/2)$ must denote the number of operations required to evaluate a polynomial of degree $\frac{n}{2} - 1$ at n points.

  However, if A_{even} (resp. A_{odd}) is evaluated at only n points how to represent $A(x)$ with $2n$ points in the combine step?

Handling (1): intro to twiddle factors ($\omega_{j,2n}$)

- Choose the $2n$ points whose x in $A(x)$ equals to each of distinct complex numbers: $\omega_{0,2n}, \omega_{1,2n}, \ldots, \omega_{2n-1,2n}$, where $\omega_{j,2n} = e^{\frac{2\pi j}{2n} i} = \cos(\frac{2\pi j}{2n}) + i\sin(\frac{2\pi j}{2n})$.

  representing a polynomial P with the set of points corresponding to twiddle factors as x-coordinates is known as the *discrete Fourier transform of P*

- Given that $\omega^2_{j,2n} = \omega_{j,n}$ and $\omega^2_{j+n,2n} = \omega^2_{j,2n}$,
  $A(\omega_{j,2n}) = A_{even}(\omega^2_{j,2n}) + w_{j,2n}A_{odd}(\omega^2_{j,2n}) = A_{even}(\omega_{j,n}) + w_{j,2n}A_{odd}(\omega_{j,n})$

- Obtaining $A(\omega_{j,2n})$ from $A_{even}(\omega_{j,n})$ and $A_{odd}(\omega_{j,n})$ for all $j = 0, 1, \ldots, 2n-1$ together takes $O(n)$ time

- $T(n) = 2T(n/2) + O(n)$ i.e., $T(n)$ is $O(n \lg n)$

Handling (3): algebra to express c_js in terms of y_ks

- From (2), we have set of points representation for $C(x)$ as:
  $(\omega_{0,2n}, y_0), (\omega_{1,2n}, y_1), \ldots, (\omega_{2n-2,2n}, y_{2n-2})$
  and, intend to find coefficients $c_0, c_1, \ldots, c_{2n-2}$ in
  $C(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{2n-2} x^{2n-2}$.

- Substituting these points in $C(x)$ yields: $VC = Y$, where

  C is a column vector of order $(2n-1) \times 1$ comprising the coefficients of $C(x)$ $\rightarrow$ vector C of this form is termed as the *inverse discrete Fourier transform of Y*

  Y is a column vector of order $(2n-1) \times 1$ comprising second coordi of points.

  V is a Vandermonde matrix of order $(2n-1) \times (2n-1)$ whose $(j,k)^{th}$ entry is $\omega^k_{j,2n}$, which is equal to $\omega_{jk,2n}$.

- Note that (j,k)th entry of V^{-1} is $\frac{\omega_{-jk,2n}}{2n}$
  and, $c_j = \frac{1}{2n} \sum_{k=0}^{2n-2} y_k \omega_{-kj,2n}$, for $j = 0, 1, \ldots, 2n-2$

Handling (3): *reducing* inverse DFT computation to DFT computation

- $c_j = \frac{1}{2n} \sum_{k=0}^{2n-2} y_k \omega_{-kj,2n}$, for $j = 0, 1, \ldots, 2n-2$.

  contrast this with the FFT of $(a_0, a_1, \ldots, a_{n-1})$ that we have computed as part of (1): $A(\omega_{j,2n}) = \sum_{k=0}^{n-1} a_k \omega_{j,2n}^k = \sum_{k=0}^{n-1} a_k \omega_{kj,2n}$

- Therefore, compute the FFT of $(y_0, y_1, \ldots, y_{2n-2})$ and divide each element of the result by $2n$ to obtain coefficient vector corresp. to $C(x)$.