INTEGER MULTIPLICATION

By: Tamal Chakraborty

Multiplication of 2 n bit numbers

- Let x and y be represented as n-digit strings in some base B. For any positive integer m less than n, one can split the two given numbers as follows

$$x = x_1 . B^m + x_0,\ x_0 < B^m$$
$$y = y_1 . B^m + y_0,\ y_0 < B^m$$

- The product xy, can be then represented as:

$$xy = (x_1 . B^m + x_0) . (y_1 . B^m + y_0)$$
$$= z_2 . B^{2m} + z_1 . B^m + z_0$$

- Where,

$$z_2 = x_1 y_1, z_1 = x_1 y_0 + x_0 y_1, z_0 = x_0 y_0$$

Performance Analysis

- Without loss of generality, let us assume that x_1, y_1, x_0, y_0 are n/2 digit numbers, for if the number of digit in any of them is less than n/2, then we can add zeros to the left-hand side to make them n/2 digit long.
- Now, clearly the multiplication of 2 n digit numbers, xy, involves 4 multiplications of n/2 digit numbers and three additions of n/2 digit numbers.
- Let T(n) be the time needed to multiply 2 n digit numbers. Then,
- $T(n) = 4T(n/2) + 3n$, since addition is $O(n)$ operation
- Solving, the above recurrence by Master Method, we get
- $T(n) = \Theta(n^2)$

Karatsuba Technique

- Karatsuba observed that xy can be computed in only 3 multiplications, such that:
  - $z_2 = x_1 y_1$
  - $z_0 = x_0 y_0$
  - $z_1 = (x_1 + x_2).(y_1 + y_2) - z_2 - z_0$
  - In this case, T(n) would be given by:
  - $T(n) = 3T(n/2) + O(n)$
  - Thus, $T(n) = \theta\left(n^{\log_2 3}\right) < \theta(n^2)$

Example

- To compute the product of 1234 and 5678, choose B = 10 and m = 2. Then
- $12\ 34 = 12 \times 10^2 + 34$
- $56\ 78 = 56 \times 10^2 + 78$
- $z_2 = 12 \times 56 = 672$
- $z_0 = 34 \times 78 = 2652$
- $z_1 = (12 + 34)(56 + 78) - z_2 - z_0$

$= 46 \times 134 - 672 - 2652 = 2840$

- result $= z_2 \times 10^{2\times2} + z_1 \times 10^2 + z_0$

$= 672 \times 10000 + 2840 \times 100 + 2652 =$ **7006652**

MULTIPLICATION OF TWO MATRICES

By: Tamal Chakraborty

Matrix Multiplication

- Suppose we wish to compute the matrix product C = AB, where A & B are n x n matrices.
- Without loss of generality we can assume that n is an exact power of 2, since if it is not an exact power of 2, we can add zeros in the rows and columns of A & B to make it a power of 2.
- Assuming n to be an exact power of 2, we divide A, B & C into 4 n/2 x n/2 matrices, then the equation C = AB can be rewritten as follows:

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & g \\ f & h \end{pmatrix}$$

Matrix Multiplications

- The matrix product equation corresponds to four equations:

  $r = ae + bf$

  $s = ag + bh$

  $t = ce + df$

  $u = cg + dh$

- Each of these equations gives rise to two multiplications of n/2 x n/2 matrices, in addition to 1 matrix addition (which is $\Theta(n^2)$ operation.)
- Thus the recurrence relation for matrix multiplication is: $T(n) = 8T(n/2) + \Theta(n^2)$
- Solving by master method, we get $T(n) = \Theta(n^3)$

Strassen's Algorithm

- In Strassen's matrix multiplication algorithm 7 (n/2 x n/2) matrix multiplications are used, instead of 8, to reduce the time.
- We need to compute r, s, t, u such that

  $r = ae + bf$

  $s = ag + bh$

  $t = ce + df$

  $u = cg + dh$
- Let $P1 = a(g - h) = ag - ah$
- $P2 = (a + b)h = ah + bh$
- Then $s = P1 + P2$

Strassen's Algorithm

- Let P3 = (c + d)e = ce + de
- P4 = d(f – e) = df – de
- Thus t = P3 + P4
- Now we have calculated s, t of the result matrix, we need to compute r = ae + bf & u = cg + dh
- Let us look at the terms ae & dh first
- Let P5 = (a + d)(e + h) = ae + ah + de + dh
- We are getting two terms ah & de in P5, which are not essential

Strassen's Algorithm

- To cancel ah & de, we can use P4 & P2, since P5 – P4 + P2 = ae + dh + df – bh. But two other inessential terms appear.
- Let P6 = (b – d)(f + h)
- Then r = P5 – P4 + P2 + P6
- Similarly, if we cancel the inessential terms in P5 by using P1 & P3, such that P5 + P1 – P3 = ae + ag – ce + dh, we get two other inessential terms.
- Let P7 = (a – c)(e + g)
- Then u = P5 + P1 – P3 – P7
- Since we are using 7 matrix products, Strassen's algorithm runs in $\theta\left(n^{\log_2 7}\right)$ time.

MULTIPLICATION OF A CHAIN OF MATRICES

By: Tamal Chakraborty

Matrix Chain Multiplication

- Let us suppose that we are given a chain of n matrices $A_1, A_2, \ldots, A_n$ and we have to calculate the matrix product P, such that $P = A_1A_2\ldots A_n$
- We know that multiplication of 2 (n x n) matrices takes $\Theta(n^3)$ time.
- Thus if two matrices A & B of dimensions (p x q) and (q x r) would require p.q.r scalar multiplications.
- Our intension is to multiply n matrices $A_1, A_2, \ldots, A_n$ to find their product P, such that the number of scalar multiplications are minimum.

Matrix Multiplication is Associative

- Suppose A, B & C are 3 matrices, then their product P can be calculated as $P = (A.B).C = A.(B.C)$
- That is, multiplying A with B first and then multiplying their product with C gives the same result as multiplying B with C first and then multiplying their product with A.
- But the way we parenthesize this operation has a big impact on the number of scalar multiplications required to compute the product.

Calculating the number of scalar multiplications

- Suppose the dimensions of A, B & C are given as (1 x 2), (2 x 5) & (5 x 1) respectively.
- If we find their product as (A.B).C then the number of scalar multiplications needed are 1.2.5 + 1.5.1 = 15
- If we find their product as A.(B.C) then the number of scalar multiplications needed are 2.5.1 + 1.2.1 = 12
- Thus we observe that parenthesizing the matrices as A.(B.C) evaluates their product with less scalar multiplications.

Problem Statement

- The matrix chain multiplication problem can be restated as follows:
- Given n matrices $A_1, A_2, \ldots, A_n$ of dimensions $(p_0 \times p_1), (p_1 \times p_2), \ldots, (p_{n-1} \times p_n)$ respectively, we have to fully parenthesize the product $P = A_1A_2\ldots A_n$ in such a way that it minimizes the number of scalar multiplications.

The structure of optimal solution

- Let $A_{i,j}$ denote the matrix product $A_i A_{i+1} \ldots A_j$
- Let us suppose that there is an integer k, such that if we parenthesize the product $A_{i,j}$ as $(A_i A_{i+1} \ldots A_k) . (A_{k+1} \ldots A_j)$ the number of scalar multiplications are minimum.
- Clearly to compute $A_{i,j}$ one has to compute the product matrices $A_{i,k} = (A_i A_{i+1} \ldots A_k)$ and $A_{k+1,j} = (A_{k+1} \ldots A_j)$ and then multiply them.
- Since the dimension of A_i is $(p_{i-1} \text{ x } p_i)$ the dimensions of $A_{i,k}$ & $A_{k+1,j}$ would be $(p_{i-1} \text{ x } p_k)$ and $(p_k \text{ x } p_j)$ respectively.
- Hence multiplication of $A_{i,k}$ & $A_{k+1,j}$ would require $p_{i-1} p_k p_j$ scalar multiplications.

A recursive solution

- Let $m(i, j)$ be the optimal cost for multiplying matrices $A_i A_{i+1} \ldots A_j$ i.e. computing $A_{i,j}$
- Since computation of $A_{i,j}$ requires computation of $A_{i,k}$ & $A_{k+1,j}$ and then $p_{i-1}p_kp_j$ scalar multiplications for evaluating their product, we have:
- $m(i, j) = m(i, k) + m(k + 1, j) + p_{i-1}p_kp_j$
- Clearly $m(i, i) = 0$ & $m(i, i + 1) = p_{i-1}p_ip_{i+1}$
- The above recurrence relation assumes that we know the value of k, which we do not, hence we need to check it for all possible values of k, where $i \leq k < j$

Example

- Find the least number of scalar multiplications to compute the product of the following chain of matrices: $A_{(1 \times 2)}$, $B_{(2 \times 5)}$, $C_{(5 \times 10)}$, $D_{(10 \times 1)}$
- Solution:

We need to compute m(1, 4), which is given by:

$$m(1,4) = min\begin{Bmatrix} m(1,1) + m(2,4) + 1.2.1 \\ m(1,2) + m(3,4) + 1.5.1 \\ m(1,3) + m(4,4) + 1.10.1 \end{Bmatrix}$$

m(1, 1) = m(4, 4) = 0

m(1, 2) = 1.2.5 = 10 & m(3, 4) = 5.10.1 = 50

Example

$$m(1,3) = min\begin{Bmatrix} m(1,1) + m(2,3) + 1.2.10 \\ m(1,2) + m(3,3) + 1.5.10 \end{Bmatrix}$$

$$= min\begin{Bmatrix} 0 + 2.5.10 + 1.2.10 \\ 1.2.5 + 0 + 1.5.10 \end{Bmatrix} = 60$$

$$m(2,4) = min\begin{Bmatrix} m(2,2) + m(3,4) + 2.5.1 \\ m(2,3) + m(3,4) + 2.10.1 \end{Bmatrix}$$

$$= min\begin{Bmatrix} 0 + 5.10.1 + 2.5.1 \\ 1.2.10 + 0 + 2.10.1 \end{Bmatrix} = 40$$

Thus $m(1,4) = min\begin{Bmatrix} 0 + 40 + 2 \\ 10 + 50 + 5 \\ 60 + 0 + 10 \end{Bmatrix} = 42$

Polynomial Multiplication Techniques

By: Tamal Chakraborty

Polynomials

- A *polynomial* *in the variable x over an algebraic field F is a representation of A(x)* as a formal sum:

$$A(x) = \sum_{j=0}^{n-1} a_j x^j$$

- *We call the values a_0, a_1, …, a_{n-1} the* ***coefficients*** *of the polynomial.*

Polynomials

- *A polynomial $A(x)$ is said to have* ***degree*** *k if its highest non-zero coefficient is a_k*
- *For example: degree of $2x^3 + 3x + 5$ is 3*
- *Any integer strictly greater than the degree of a polynomial is a* ***degree-bound*** *of that polynomial.*
- *For example: degree bound of $2x^3 + 3x + 5$ is 4*

Polynomial Operations: Addition

- *if A(x) and B(x) are polynomials of degree-bound n, we say that their* ***sum*** *is a polynomial C(x), also of degree-bound n, such that C(x) = A(x) + B(x) for all x in the underlying field. That is:*

$$\textit{if}\ \ A(x) = \sum_{j=0}^{n-1} a_j x^j \ \textit{and}\ B(x) = \sum_{j=0}^{n-1} b_j x^j \ \text{then}\ C(x) = \sum_{j=0}^{n-1} c_j x^j$$

- *where $c_j = a_j + b_j$ for $j = 0, 1, \ldots, n - 1$.*
- *For example, if we have the polynomials*
- *$A(x) = 6x^3 + 7x^2 - 10x + 9$ and*
- *$B(x) = -2x^3 + 4x - 5$, then*
- *$C(x) = 4x^3 + 7x^2 - 6x + 4$.*

Polynomial Operations: Multiplication

- *if $A(x)$ and $B(x)$ are polynomials of degree-bound n, we say that their product $C(x)$ is a polynomial of degree-bound $2n - 1$ such that $C(x) = A(x)\,B(x)$ for all x in the underlying field.*
- *We can multiply polynomials by multiplying each term in $A(x)$ by each term in $B(x)$ and combining terms with equal powers.*

Polynomial Operations: Multiplication

- *For example, we can multiply*
- *$A(x) = 6x^3 + 7x^2 - 10x + 9$ and*
- *$B(x) = -2x^3 + 4x - 5$*
- *as follows:*

$$\begin{array}{rrrrrrr}
 & & & 6x^3 & +\ 7x^2 & -\ 10x & +\ 9 \\
 & & & -\ 2x^3 & & +\ 4x & -\ 5 \\
\hline
 & & & -\ 30x^3 & -\ 35x^2 & +\ 50x & -\ 45 \\
 & & 24x^4 & +\ 28x^3 & -\ 40x^2 & +\ 36x & \\
-\ 12x^6 & -\ 14x^5 & +\ 20x^4 & -\ 18x^3 & & & \\
\hline
-\ 12x^6 & -\ 14x^5 & +\ 44x^4 & -\ 20x^3 & -\ 75x^2 & +\ 86x & -\ 45
\end{array}$$

We have, $C(x) = \sum_{j=0}^{2n-2} c_j x^j$ where $c_j = \sum_{k=0}^{j} a_k b_{j-k}$

Polynomial Representation

- Coefficient Representation
  - *A* ***coefficient representation*** *of a polynomial $A(x)$ of degree-bound n is a vector of coefficients $a = (a_0, a_1, \ldots, a_{n-1})$.*
  - *For example: if $A(x) = 2x^2 + 3x + 4$ then $(4, 3, 2)$ is a coefficient representation of $A(x)$*
  - *the operation of* ***evaluating*** *the polynomial $A(x)$ at a given point x_0 consists of computing the value of $A(x_0)$.* ***Evaluation*** *takes time $\Theta(n)$ using Horner's rule:*
  - $A(x_0) = a_0 + x_0(a_1 + x_0(a_2 + \ldots + x_0(a_{n-2} + x_0(a_{n-1}))\ldots))$
  - *For example: evaluation of $A(x)$ at $x = 5$ is given by*
  - $A(5) = 4 + 5(3 + 5(2)) = 69$

Polynomial Representation

- Point-value Representation
  - *A point-value representation of a polynomial $A(x)$ of degree-bound n 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 example: if $A(x) = 2x^2 + 3x + 4$ then $\{(0, 4), (1, 9), (2, 18)\}$ is a point-value representation of $A(x)$.*
  - *Given a coefficient form of a polynomial the point-value form can be computed using Horner's method, by evaluating $A(x_k)$ for $k = 0, 1, \ldots, n-1$. This n-point evaluation takes $\Theta(n^2)$ time*
  - *The process of determining the coefficient form of a polynomial, given the point-value representation is called* ***interpolation.***

Lagrange's Interpolation Formula

- *Given n points (x_k, y_k), the coefficients of the unique polynomial is obtained using Lagrange's formula:*

$$A(x) = \sum_{k=0}^{n-1} y_k \frac{\prod_{j \neq k} (x - x_j)}{\prod_{j \neq k} (x_k - x_j)}$$

- *For example: if {(0, 4), (1, 9), (2, 18)} is the input:*

$$A(x) = \frac{(x-1)(x-2)}{(0-1)(0-2)}4 + \frac{(x-0)(x-2)}{(1-0)(1-2)}9 + \frac{(x-0)(x-1)}{(2-0)(2-1)}18$$

$$or, \mathrm{A(x)} = 2(\mathrm{x}^2 - 3\mathrm{x} + 2) - 9(\mathrm{x}^2 - 2\mathrm{x}) + 9(\mathrm{x}^2 - \mathrm{x})$$

$$\mathrm{or,\ A(x)} = 2x^2 + 3x + 4$$

Lagrange's Interpolation Formula

- ***Exercise:***
- *Find the coefficient form of the polynomial given the point-value form {(0,5), (1, 10), (2, 21)}*

- ***Note:***
- *Given the point-value form we can get the coefficient form of the polynomial in* $\Theta(n^2)$ *time, where n is the degree bound of the polynomial.*

Polynomial Multiplication

- *Let A(x) and B(x) be two polynomials represented in coefficient form. Computing their product C(x) takes* $\Theta(n^2)$ *time, since the computation involves multiplication of each coefficient a of A with each coefficient b of B.*
- *Let* $A(x) = \{(x_0, y_0), (x_1, y_1), \ldots, (x_{2n-1}, y_{2n-1})\}$ *and* $B(x) = \{(x_0, ý_0), (x_1, ý_1), \ldots, (x_{2n-1}, ý_{2n-1})\}$ *be two polynomials represented in point-value form. Computing their product C(x) takes* $\Theta(n)$ *time, where* $C(x) = \{(x_0, y_0 ý_0), (x_1, y_1 ý_1), \ldots, (x_{2n-1}, y_{2n-1} ý_{2n-1})\}$

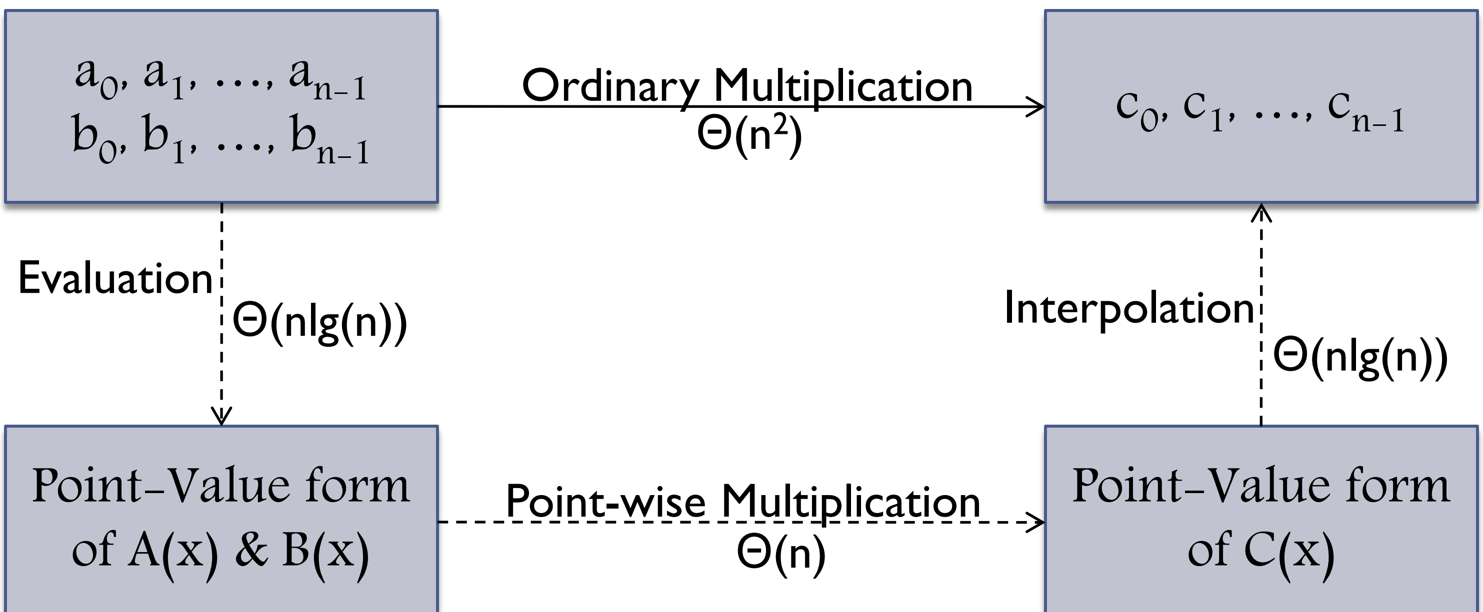
Polynomial Multiplication

Can we use the linear-time multiplication method for polynomials in point-value form to expedite polynomial multiplication in coefficient form?

The answer hinges on our ability to convert a polynomial quickly from coefficient form to point-value form (evaluate) and vice-versa (interpolate).

Polynomial Multiplication

- *While converting from coefficient to point-value form we can use any points we want as evaluation points, but by choosing the evaluation points* ***wisely****, we can convert between representations in only $\Theta(n\lg(n))$ time.*



Complex Roots of Unity

- *We will show that if we use the complex roots of unity, then we can evaluate and interpolate polynomials in $\Theta(n\lg(n))$ time.*
- *A complex nth root of unity is a complex number ω such that: $\omega^n = 1$*
- *There are exactly n complex nth roots of unity, given by $e^{2\pi ik/n}$ for $k = 0,1,\ldots,n-1$, where $i = \sqrt{-1}$*
- *$\omega_n = e^{2\pi i/n}$ is the principal nth root of unity.*
- *The other complex nth roots of unity are powers of ω_n, given by $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$*

Properties of Complex Roots of Unity

1. $(\omega_n^{k})^2 = \omega_{n/2}^{k}$

- $\omega_{n/2}^{k} = e^{2\pi ik/(n/2)} = e^{4\pi ik/n} = (\omega_n^{k})^2$

2. $(\omega_n^{k})^2 = (\omega_n^{k+n/2})^2$

- $(\omega_n^{k+n/2})^2 = \omega_n^{2k+n} = \omega_n^{2k}\omega_n^{n} = \omega_n^{2k} = (\omega_n^{k})^2$

3. **Summation Property:** $\sum_{j=0}^{n-1} (\omega_n^k)^j = 0$

We know $\sum_{k=0}^{n} x^k = 1 + x + x^2 + \ldots + x^n$

is a geometric series and has the value : $\sum_{k=0}^{n} x^k = \dfrac{x^{n+1} - 1}{x - 1}$

$$\text{Thus, } \sum_{j=0}^{n-1} (\omega_n^k)^j = \frac{(\omega_n^k)^n - 1}{\omega_n^k - 1} = \frac{(\omega_n^n)^k - 1}{\omega_n^k - 1} = \frac{(1)^k - 1}{\omega_n^k - 1} = 0$$

The DFT

- *We wish to evaluate a polynomial* $A(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$ *of degree-bound n at* $\omega_n^{\,0}, \omega_n^{\,1}, \ldots, \omega_n^{\,n-1}$
- *Without loss of generality we assume that n is a power of 2, since a given degree bound can always be raised by adding new high order zero coefficients as necessary.*
- *Let A be given in coefficient form* $a = (a_0, a_1, \ldots, a_{n-1})$.
- *Let* y_k *be the results at* $k = 0, 1, \ldots, n-1$, *i.e.*

$$y_k = A(\omega_n^k) = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$$

- *The vector* $y = (y_0, y_1, \ldots, y_{n-1})$ *is called the* ***Discrete Fourier Transform (DFT)*** *of the coefficient vector* $a = (a_0, a_1, \ldots, a_{n-1})$. *We write* $y = DFT_n(a)$

The FFT

- *By using a method called the* ***Fast Fourier Transform (FFT)****, which takes advantage of the special properties of the complex roots of unity, we can compute* $DFT_n(a)$ *in* $\Theta(n\lg(n))$ *time.*
- *The FFT employs a divide & conquer strategy, using the even-index and odd-index coefficients of* $A(x)$ *separately to define two new n/2 degree-bound polynomials* $A^0(x)$ *and* $A^1(x)$.
- $A^0(x) = a_0 + a_2x + a_4x^2 + \ldots + a_{n-2}x^{n/2-1}$
- $A^1(x) = a_1 + a_3x + a_5x^2 + \ldots + a_{n-1}x^{n/2-1}$
- *Thus* $A(x) = A^0(x^2) + x\,A^1(x^2)$

The FFT

- *So, the problem of evaluating* $A(x)$ *at* $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$ *now reduces to:*

1. *Evaluating the n/2 degree bound polynomials* $A^0(x)$ *and* $A^1(x)$ *at points* $(\omega_n^0)^2, (\omega_n^1)^2, \ldots, (\omega_n^{n-1})^2$
2. *Then combining the results according to the equation* $A(x) = A^0(x^2) + x\,A^1(x^2)$

- *Using the properties of complex roots of unity the list of values in step 1 consists not of n-distinct values but only of n/2 complex* $(n/2)^{th}$ *roots of unity.*

The FFT

- Let us illustrate with an example that we need to evaluate $A^0(x)$ and $A^1(x)$ only at n/2 points instead of n points.

1. Say, n = 4, So we need to compute the polynomials at points (ω40)2, (ω41)2, (ω42)2, (ω43)2
2. As per the rule: (ωnk)2 = ωn/2k, we have (ω40)2 = (ω20) and (ω41)2 = (ω21)
3. As per the rule: (ωnk)2 = (ωnk + n/2)2 , we have (ω42)2 = (ω40+2)2 = (ω40)2 (here k = 0) and (ω43)2 = (ω41+2)2 = (ω41)2 (here k = 1)
4. But as shown in step 2, (ω40)2 = (ω20) and (ω41)2 = (ω21)
5. Hence we need to evaluate A0(x) and A1(x) only at 2 points (ω20) and (ω21)

The FFT

- Let $T(n)$ be the total time taken by the algorithm for evaluating the degree-bound n polynomial $A(x)$ at n distinct points.
- Then as per our divide & conquer strategy we need to evaluate two degree-bound n/2 polynomials $A^0(x)$ and $A^1(x)$ only at n/2 points, each of which will take $T(n/2)$ time.
- Then we have to combine the result using equation $A(x) = A^0(x^2) + x\, A^1(x^2)$, which requires polynomial addition and hence takes $\Theta(n)$ time.
- So, we get a recurrence relation of the form:
- $\mathbf{T(n) = 2T(n/2) + \Theta(n)}$
- Solving this using the master method we see that the time taken to evaluate the polynomial is $\mathbf{\Theta(nlg(n))}$.

The FFT

- *So far, we have evaluated a polynomial using the n complex roots of unity as points of evaluation in $\Theta(n\lg(n))$ time.*
- *Now we need to interpolate the complex roots of unity by a polynomial.*
- *If a polynomial $A(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$ has the point value form $\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$ then we can represent it in the matrix equation:*

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_{n-1} & x_{n-1}^2 & \ldots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \ldots \\ \ldots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \ldots \\ \ldots \\ y_{n-1} \end{pmatrix}$$

The FFT

- *In this case we have evaluated the polynomial A(x) = $a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1}$ at points $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$ so the point value form is $\{(x_0, \omega_n^0), (x_1, \omega_n^1), \ldots, (x_{n-1}, \omega_n^{n-1})\}$, their relationship can be represented by the following matrix equation:*

$$\begin{pmatrix} 1 & \omega_n^0 & (\omega_n^0)^2 & \ldots & (\omega_n^0)^{n-1} \\ 1 & \omega_n^1 & (\omega_n^1)^2 & \ldots & (\omega_n^1)^{n-1} \\ 1 & \omega_n^2 & (\omega_n^2)^2 & \ldots & (\omega_n^2)^{n-1} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & \omega_n^{n-1} & (\omega_n^{n-1})^2 & \ldots & (\omega_n^{n-1})^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \ldots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \ldots \\ y_{n-1} \end{pmatrix}$$

- *i.e. $\boldsymbol{y} = \boldsymbol{V_n a}$, where V_n is called the Vandermonde matrix containing the appropriate powers of ω_n*
- ***The (k, j)th entry in V_n is ω_n^{kj}** for j, k = 0, 1, …, n–1*
- *The coefficient vector a can be obtained by $\boldsymbol{a} = \boldsymbol{V_n^{-1} y}$*

The FFT

- *Theorem:*
  - *for j, k = 0, 1, ..., n-1 the (j, k) entry of* V_n^{-1} *is* ω_n^{-kj} / n
- *Proof:*
  - *We will prove the theorem by showing that* $V_n^{-1} V_n = I_n$
  - *Where* I_n *is the n x n identity matrix*
  - *The (j, l) entry of* $V_n^{-1} V_n$ *is given by:*

$$[V_n^{-1} V_n]_{j,l} = \sum_{k=0}^{n-1} (\omega_n^{-kj} / n)(\omega_n^{kl}) = \sum_{k=0}^{n-1} \omega_n^{k(l-j)} / n$$

  - *The summation equals 1 if j = l and it is 0 otherwise by the summation property of* ω_n *(refer to slide 15 for proof)*
  - *Hence* $V_n^{-1} V_n = I_n$

The FFT

- *Given the inverse matrix V_n^{-1}, $a = DFT_n^{-1}(y)$ is given by:*

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

- *Comparing this with the equation $y = DFT_n(a)$, i.e.*

$$y_k = \sum_{j=0}^{n-1} a_j \omega_n^{kj}$$

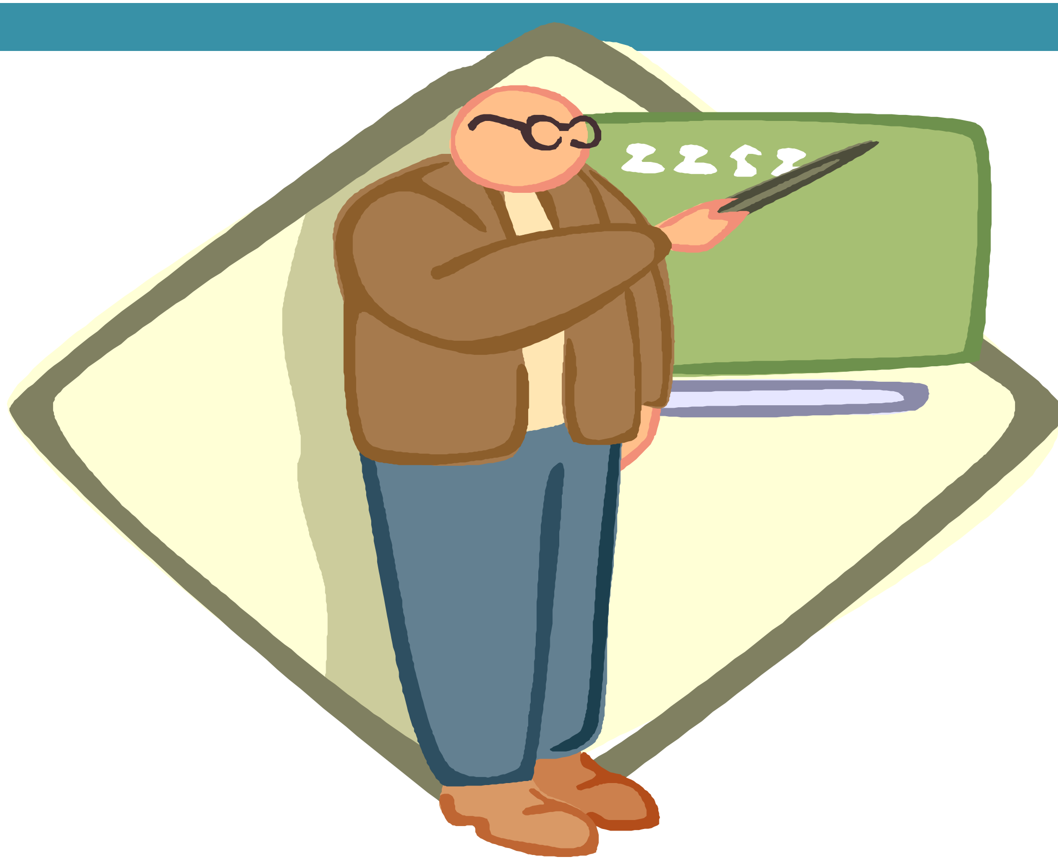
- *We see that by modifying the FFT algorithm to switch the roles of a and y, replacing ω_n by ω_n^{-1} and dividing each element of result by n, we can compute the inverse of DFT.*
- *Thus DFT_n^{-1} can be computed in* ***$\Theta(n\lg(n))$*** *time as well.*

The FFT

- *So, two polynomials A(x) and B(x) in coefficient form can be multiplied by:*

1. *Evaluating them at n distinct points* $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$ *in* $\Theta(n\lg(n))$ *time, where* ω_n *is the principal complex* n^{th} *root of untiy.*
2. *Multiplying the results of step 1 (in point-value form) in* $\Theta(n)$ *time, to get the product.*
3. *Interpolating the product polynomial of step 2 to its coefficient form to get the desired output* $C(x) = A(x).B(x)$ *in* $\Theta(n\lg(n))$ *time.*

- *Hence total time taken by the polynomial multiplication algorithm is* $\Theta(n\lg(n)) + \Theta(n) + \Theta(n\lg(n))$
- ***Which is*** $\boldsymbol{\Theta(n\lg(n))}$.



Thank You!!!