The Discrete Fourier Transform

Abhiram Ranade

January 13, 2016

Tutorial

1. Which of the following statements convey information? Which can be simplified while conveying the same information?
   1.1 Algorithm A takes time at least $O(n^2)$.
   1.2 Algorithm A times time at most $O(n^2)$
2. Suppose I have 27 coins, one of which is heavier than the rest, which have the same weight. You are to find the heavier coin using a two pan balance. With the balance, you can compare whether a given set of coins is heavier than another set, or is lighter, or both have equal weight. Show that you can determine the heavy coin using 3 comparisons.
3. Show that three comparisons are necessary. Hint: Argue that every algorithm must be a decision tree..
4. Harder: Suppose one coin is known to have a *different* weight than the rest – you do not know whether it is heavier or lighter. Determine the number of comparisons needed as a function of n. Give upper and lower bounds.

Convolution using Divide and Conquer

Convolution using Divide and Conquer

Convolution:

Convolution using Divide and Conquer

Convolution:

Input: Numbers $a_0, \ldots, a_{n-1}$, $b_0, \ldots, b_{m-1}$

Convolution using Divide and Conquer

Convolution:

Input: Numbers $a_0, \ldots, a_{n-1}$, $b_0, \ldots, b_{m-1}$

Output: $c_0, \ldots, c_{m+n-2}$ where $c_k = \sum_{i,j|i+j=k} a_i b_j$

Convolution using Divide and Conquer

Convolution:

Input: Numbers $a_0, \ldots, a_{n-1}$, $b_0, \ldots, b_{m-1}$

Output: $c_0, \ldots, c_{m+n-2}$ where $c_k = \sum_{i,j|i+j=k} a_i b_j$

Alternative definition:

Convolution using Divide and Conquer

Convolution:

Input: Numbers $a_0, \ldots, a_{n-1}$, $b_0, \ldots, b_{m-1}$

Output: $c_0, \ldots, c_{m+n-2}$ where $c_k = \sum_{i,j|i+j=k} a_i b_j$

Alternative definition:

Let $A(x) = \sum_{i=0}^{n-1} a_i x^i, B(x) = \sum_{j=0}^{m-1} b_j x^j$

Convolution using Divide and Conquer

Convolution:

Input: Numbers $a_0, \ldots, a_{n-1}$, $b_0, \ldots, b_{m-1}$

Output: $c_0, \ldots, c_{m+n-2}$ where $c_k = \sum_{i,j|i+j=k} a_i b_j$

Alternative definition:

Let $A(x) = \sum_{i=0}^{n-1} a_i x^i$, $B(x) = \sum_{j=0}^{m-1} b_j x^j$

Then c_k are coefficients of $C(x) = A(x)B(x)$

Convolution using Divide and Conquer

Convolution:
Input: Numbers $a_0, \ldots, a_{n-1}$, $b_0, \ldots, b_{m-1}$
Output: $c_0, \ldots, c_{m+n-2}$ where $c_k = \sum_{i,j|i+j=k} a_i b_j$

Alternative definition:
Let $A(x) = \sum_{i=0}^{n-1} a_i x^i$, $B(x) = \sum_{j=0}^{m-1} b_j x^j$
Then c_k are coefficients of $C(x) = A(x)B(x)$

Time to convolve using "direct" method: $O(mn)$

Convolution using Divide and Conquer

Convolution:
Input: Numbers $a_0, \ldots, a_{n-1}$, $b_0, \ldots, b_{m-1}$
Output: $c_0, \ldots, c_{m+n-2}$ where $c_k = \sum_{i,j|i+j=k} a_i b_j$

Alternative definition:
Let $A(x) = \sum_{i=0}^{n-1} a_i x^i$, $B(x) = \sum_{j=0}^{m-1} b_j x^j$
Then c_k are coefficients of $C(x) = A(x)B(x)$

Time to convolve using "direct" method: $O(mn)$

Today: Time $O(n \log n)$ Assume $n \geq m$

Convolution using Divide and Conquer

Convolution:
Input: Numbers $a_0, \ldots, a_{n-1}$, $b_0, \ldots, b_{m-1}$
Output: $c_0, \ldots, c_{m+n-2}$ where $c_k = \sum_{i,j|i+j=k} a_i b_j$

Alternative definition:
Let $A(x) = \sum_{i=0}^{n-1} a_i x^i, B(x) = \sum_{j=0}^{m-1} b_j x^j$
Then c_k are coefficients of $C(x) = A(x)B(x)$

Time to convolve using "direct" method: $O(mn)$

Today: Time $O(n \log n)$ Assume $n \geq m$

Polynomial multiplication is related to integer multiplication.
Fastest integer multiplication algorithm is based on polynomial multiplication algorithms.

Convolution using Divide and Conquer

Convolution:
Input: Numbers $a_0, \ldots, a_{n-1}$, $b_0, \ldots, b_{m-1}$
Output: $c_0, \ldots, c_{m+n-2}$ where $c_k = \sum_{i,j|i+j=k} a_i b_j$

Alternative definition:
Let $A(x) = \sum_{i=0}^{n-1} a_i x^i$, $B(x) = \sum_{j=0}^{m-1} b_j x^j$
Then c_k are coefficients of $C(x) = A(x)B(x)$

Time to convolve using "direct" method: $O(mn)$

Today: Time $O(n \log n)$ Assume $n \geq m$

Polynomial multiplication is related to integer multiplication. Fastest integer multiplication algorithm is based on polynomial multiplication algorithms.

Convolution is very important in Signal Processing.

The basic signal processing problem

Input Signal $\rightarrow$ $\boxed{\text{"Filter"}}$ $\rightarrow$ Output signal

The basic signal processing problem

Input Signal $\rightarrow \boxed{\text{"Filter"}} \rightarrow$ Output signal

Action of a "linear time invariant" filter:

The basic signal processing problem

Input Signal $\rightarrow \boxed{\text{"Filter"}} \rightarrow$ Output signal

Action of a "linear time invariant" filter:

Input: Unit height pulse at time 0

The basic signal processing problem

Input Signal $\rightarrow \boxed{\text{"Filter"}} \rightarrow$ Output signal

Action of a "linear time invariant" filter:

Input: Unit height pulse at time 0

Output: Pulses of heights b_j at time j, for $j = 0, \ldots, m-1$.

The basic signal processing problem

Input Signal $\rightarrow \boxed{\text{"Filter"}} \rightarrow$ Output signal

Action of a "linear time invariant" filter:

Input: Unit height pulse at time 0

Output: Pulses of heights b_j at time j, for $j = 0, \ldots, m-1$.

$b_0, \ldots, b_{m-1}$ = characteristic of the filter. "Impulse response"

The basic signal processing problem

Input Signal $\rightarrow \boxed{\text{"Filter"}} \rightarrow$ Output signal

Action of a "linear time invariant" filter:
Input: Unit height pulse at time 0
Output: Pulses of heights b_j at time j, for $j = 0, \ldots, m-1$.
$b_0, \ldots, b_{m-1}$ = characteristic of the filter. "Impulse response"

Input: Unit height pulse at time t.

The basic signal processing problem

Input Signal $\rightarrow$ $\boxed{\text{"Filter"}}$ $\rightarrow$ Output signal

Action of a "linear time invariant" filter:
Input: Unit height pulse at time 0
Output: Pulses of heights b_j at time j, for $j = 0, \ldots, m-1$.
$b_0, \ldots, b_{m-1}$ = characteristic of the filter. "Impulse response"

Input: Unit height pulse at time t.
Output: Pulses of height b_j at time $t + j$, for $j = 0, \ldots, m-1$.

The basic signal processing problem

Input Signal $\rightarrow$ $\boxed{\text{"Filter"}}$ $\rightarrow$ Output signal

Action of a "linear time invariant" filter:

Input: Unit height pulse at time 0

Output: Pulses of heights b_j at time j, for $j = 0, \ldots, m-1$.

$b_0, \ldots, b_{m-1}$ = characteristic of the filter. "Impulse response"

Input: Unit height pulse at time t.

Output: Pulses of height b_j at time $t + j$, for $j = 0, \ldots, m-1$.

Time invariance

The basic signal processing problem

Input Signal $\rightarrow$ $\boxed{\text{"Filter"}}$ $\rightarrow$ Output signal

Action of a "linear time invariant" filter:

Input: Unit height pulse at time 0

Output: Pulses of heights b_j at time j, for $j = 0, \ldots, m-1$.

$b_0, \ldots, b_{m-1}$ = characteristic of the filter. "Impulse response"

Input: Unit height pulse at time t.

Output: Pulses of height b_j at time $t + j$, for $j = 0, \ldots, m-1$.

Time invariance

Input: Pulses of height $a_0, \ldots, a_{n-1}$ at times $0, \ldots, n-1$

The basic signal processing problem

Input Signal $\rightarrow$ $\boxed{\text{"Filter"}}$ $\rightarrow$ Output signal

Action of a "linear time invariant" filter:

Input: Unit height pulse at time 0

Output: Pulses of heights b_j at time j, for $j = 0, \ldots, m-1$.

$b_0, \ldots, b_{m-1}$ = characteristic of the filter. "Impulse response"

Input: Unit height pulse at time t.

Output: Pulses of height b_j at time $t + j$, for $j = 0, \ldots, m-1$.

Time invariance

Input: Pulses of height $a_0, \ldots, a_{n-1}$ at times $0, \ldots, n-1$

Output: Pulses of height c_k, at times $k = 0, \ldots, m+n-2$, where

$c_k = \sum_{i,j|k=i+j} a_i b_j$

The basic signal processing problem

Input Signal $\rightarrow$ $\boxed{\text{"Filter"}}$ $\rightarrow$ Output signal

Action of a "linear time invariant" filter:
Input: Unit height pulse at time 0
Output: Pulses of heights b_j at time j, for $j = 0, \ldots, m-1$.
$b_0, \ldots, b_{m-1}$ = characteristic of the filter. "Impulse response"

Input: Unit height pulse at time t.
Output: Pulses of height b_j at time $t + j$, for $j = 0, \ldots, m-1$.
Time invariance

Input: Pulses of height $a_0, \ldots, a_{n-1}$ at times $0, \ldots, n-1$
Output: Pulses of height c_k, at times $k = 0, \ldots, m+n-2$, where
$c_k = \sum_{i,j|k=i+j} a_i b_j$
Linearity

The basic signal processing problem

Input Signal $\rightarrow \boxed{\text{"Filter"}} \rightarrow$ Output signal

Action of a "linear time invariant" filter:
Input: Unit height pulse at time 0
Output: Pulses of heights b_j at time j, for $j = 0, \ldots, m-1$.
$b_0, \ldots, b_{m-1}$ = characteristic of the filter. "Impulse response"

Input: Unit height pulse at time t.
Output: Pulses of height b_j at time $t + j$, for $j = 0, \ldots, m-1$.
Time invariance

Input: Pulses of height $a_0, \ldots, a_{n-1}$ at times $0, \ldots, n-1$
Output: Pulses of height c_k, at times $k = 0, \ldots, m+n-2$, where
$c_k = \sum_{i,j|k=i+j} a_i b_j$
Linearity

Output is a convolution of input and impulse response

Alternate representation of polynomials

Alternate representation of polynomials

$a_0, \ldots, a_{n-1}$ is "coefficient representation"

Alternate representation of polynomials

$a_0, \ldots, a_{n-1}$ is "coefficient representation"

Theorem: There is a unique degree d polynomial through $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ if x_i are distinct.

Alternate representation of polynomials

$a_0, \ldots, a_{n-1}$ is "coefficient representation"

Theorem: There is a unique degree d polynomial through $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ if x_i are distinct.

$(x_0, A(x_0)), (x_1, A(x_1)), \ldots, (x_{n-1}, A(x_{n-1}))$ is "point-value representation".

Alternate representation of polynomials

$a_0, \ldots, a_{n-1}$ is "coefficient representation"

Theorem: There is a unique degree d polynomial through $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ if x_i are distinct.

$(x_0, A(x_0)), (x_1, A(x_1)), \ldots, (x_{n-1}, A(x_{n-1}))$ is "point-value representation".

Coefficient $\rightarrow$ point-value : Evaluate!

Alternate representation of polynomials

$a_0, \ldots, a_{n-1}$ is "coefficient representation"

Theorem: There is a unique degree d polynomial through $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ if x_i are distinct.

$(x_0, A(x_0)), (x_1, A(x_1)), \ldots, (x_{n-1}, A(x_{n-1}))$ is "point-value representation".

Coefficient $\rightarrow$ point-value : Evaluate!
point-value $\rightarrow$ Coefficient : Lagrange Interpolation

Alternate representation of polynomials

$a_0, \ldots, a_{n-1}$ is "coefficient representation"

Theorem: There is a unique degree d polynomial through $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ if x_i are distinct.

$(x_0, A(x_0)), (x_1, A(x_1)), \ldots, (x_{n-1}, A(x_{n-1}))$ is "point-value representation".

Coefficient $\rightarrow$ point-value : Evaluate!
point-value $\rightarrow$ Coefficient : Lagrange Interpolation

Suppose $C(x) = A(x)B(x)$.

Alternate representation of polynomials

$a_0, \ldots, a_{n-1}$ is "coefficient representation"

Theorem: There is a unique degree d polynomial through $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ if x_i are distinct.

$(x_0, A(x_0)), (x_1, A(x_1)), \ldots, (x_{n-1}, A(x_{n-1}))$ is "point-value representation".

Coefficient $\rightarrow$ point-value : Evaluate!
point-value $\rightarrow$ Coefficient : Lagrange Interpolation

Suppose $C(x) = A(x)B(x)$.
If we have the A, B in point-value form, at same $2n$ points, then we get C in point-value form using $2n$ multiplications!

Alternate representation of polynomials

$a_0, \ldots, a_{n-1}$ is "coefficient representation"

Theorem: There is a unique degree d polynomial through $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ if x_i are distinct.

$(x_0, A(x_0)), (x_1, A(x_1)), \ldots, (x_{n-1}, A(x_{n-1}))$ is "point-value representation".

Coefficient $\rightarrow$ point-value : Evaluate!
point-value $\rightarrow$ Coefficient : Lagrange Interpolation

Suppose $C(x) = A(x)B(x)$.
If we have the A, B in point-value form, at same $2n$ points, then we get C in point-value form using $2n$ multiplications!
Representation of C is $(\ldots, (x_i, A(x_i)B(x_i)), \ldots)$

Alternate representation of polynomials

$a_0, \ldots, a_{n-1}$ is "coefficient representation"

Theorem: There is a unique degree d polynomial through $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ if x_i are distinct.

$(x_0, A(x_0)), (x_1, A(x_1)), \ldots, (x_{n-1}, A(x_{n-1}))$ is "point-value representation".

Coefficient $\rightarrow$ point-value : Evaluate!
point-value $\rightarrow$ Coefficient : Lagrange Interpolation

Suppose $C(x) = A(x)B(x)$.
If we have the A, B in point-value form, at same $2n$ points, then we get C in point-value form using $2n$ multiplications!
Representation of C is $(\ldots, (x_i, A(x_i)B(x_i)), \ldots)$

Alternate convolution algorithm: Convert A, B to point-value form ($2n$ points), pointwise multiply, Interpolate result.

Alternate representation of polynomials

$a_0, \ldots, a_{n-1}$ is "coefficient representation"

Theorem: There is a unique degree d polynomial through $(x_0, y_0), (x_1, y_1), \ldots, (x_d, y_d)$ if x_i are distinct.

$(x_0, A(x_0)), (x_1, A(x_1)), \ldots, (x_{n-1}, A(x_{n-1}))$ is "point-value representation".

Coefficient $\rightarrow$ point-value : Evaluate!
point-value $\rightarrow$ Coefficient : Lagrange Interpolation

Suppose $C(x) = A(x)B(x)$.
If we have the A, B in point-value form, at same $2n$ points, then we get C in point-value form using $2n$ multiplications!
Representation of C is $(\ldots, (x_i, A(x_i)B(x_i)), \ldots)$

Alternate convolution algorithm: Convert A, B to point-value form ($2n$ points), pointwise multiply, Interpolate result.

Need fast evaluation/interpolation.

Can we evaluate/interpolate quickly?

Can we evaluate/interpolate quickly?

Naive evaluation method: Evaluating degree $n-1$ polynomial at 1 point takes $O(n)$ time. At $2n$ points: $O(n^2)$ time.

Can we evaluate/interpolate quickly?

Naive evaluation method: Evaluating degree $n-1$ polynomial at 1 point takes $O(n)$ time. At $2n$ points: $O(n^2)$ time.

Main Result 1: It is possible to evaluate a degree $n-1$ polynomial at n carefully chosen points in time $O(n \log n)$.

Can we evaluate/interpolate quickly?

Naive evaluation method: Evaluating degree $n-1$ polynomial at 1 point takes $O(n)$ time. At $2n$ points: $O(n^2)$ time.

Main Result 1: It is possible to evaluate a degree $n-1$ polynomial at n carefully chosen points in time $O(n \log n)$.

Main Result 2: It is possible to construct the coefficients of a degree $n-1$ polynomial given the values at n carefully chosen points in time $O(n \log n)$.

Can we evaluate/interpolate quickly?

Naive evaluation method: Evaluating degree $n-1$ polynomial at 1 point takes $O(n)$ time. At $2n$ points: $O(n^2)$ time.

Main Result 1: It is possible to evaluate a degree $n-1$ polynomial at n carefully chosen points in time $O(n \log n)$.

Main Result 2: It is possible to construct the coefficients of a degree $n-1$ polynomial given the values at n carefully chosen points in time $O(n \log n)$.

For convolving, we need to evaluate the polynomial at $2n-1$ points. To fit with the above results, we simply consider the polynomials to be of degree $n' = 2n-2$, using 0s for higher coefficients. Then we use the above results substituting n' for n.

Can we evaluate/interpolate quickly?

Naive evaluation method: Evaluating degree $n-1$ polynomial at 1 point takes $O(n)$ time. At $2n$ points: $O(n^2)$ time.

Main Result 1: It is possible to evaluate a degree $n-1$ polynomial at n carefully chosen points in time $O(n \log n)$.

Main Result 2: It is possible to construct the coefficients of a degree $n-1$ polynomial given the values at n carefully chosen points in time $O(n \log n)$.

For convolving, we need to evaluate the polynomial at $2n-1$ points. To fit with the above results, we simply consider the polynomials to be of degree $n' = 2n-2$, using 0s for higher coefficients. Then we use the above results substituting n' for n.

Convolution time: Evaluate + multiply + Interpolate : $O(n \log n) + O(n) + O(n \log n) = O(n \log n)$

Choosing the points at which to evaluate

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $A(-x) = \sum_{i \ even} a_i x^i - \sum_{i \ odd} a_i x^i$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\qquad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\qquad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\qquad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\quad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\quad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\qquad$ $A(-x) = \sum_{i \ even} a_i x^i - \sum_{i \ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\qquad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\qquad$ $A(-x) = E(x^2) - xO(x^2)$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\qquad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\qquad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\qquad$ $A(-x) = E(x^2) - xO(x^2)$

Value of E, O at $x^2 \rightarrow$ value of A at $\pm x$ $\qquad$ in $O(1)$ time

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\qquad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\qquad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\qquad$ $A(-x) = E(x^2) - xO(x^2)$

Value of E, O at x^2 $\rightarrow$ value of A at $\pm x$ $\qquad$ in $O(1)$ time

Value of E, O at $x_0^2, x_1^2, \ldots, x_{n/2-1}^2$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\qquad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\qquad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\qquad$ $A(-x) = E(x^2) - xO(x^2)$

Value of E, O at x^2 $\rightarrow$ value of A at $\pm x$ $\qquad$ in $O(1)$ time

Value of E, O at $x_0^2, x_1^2, \ldots, x_{n/2-1}^2$
$\rightarrow$ value of A at $\pm x_0, \pm x_1, \pm x_{n/2-1}$ $\qquad$ in $O(n)$ time

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\qquad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\qquad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\qquad$ $A(-x) = E(x^2) - xO(x^2)$

Value of E, O at x^2 $\rightarrow$ value of A at $\pm x$ $\qquad$ in $O(1)$ time

Value of E, O at $x_0^2, x_1^2, \ldots, x_{n/2-1}^2$
$\rightarrow$ value of A at $\pm x_0, \pm x_1, \pm x_{n/2-1}$ $\qquad$ in $O(n)$ time

Alternatively: Value of E, O at $v_0, v_1, v_{n/2-1}$ $\qquad$ degree $n/2 - 1$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\quad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\quad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\quad$ $A(-x) = E(x^2) - xO(x^2)$

Value of E, O at x^2 $\rightarrow$ value of A at $\pm x$ $\quad$ in $O(1)$ time

Value of E, O at $x_0^2, x_1^2, \ldots, x_{n/2-1}^2$
$\rightarrow$ value of A at $\pm x_0, \pm x_1, \pm x_{n/2-1}$ $\quad$ in $O(n)$ time

Alternatively: Value of E, O at $v_0, v_1, v_{n/2-1}$ $\quad$ degree $n/2 - 1$
$\rightarrow$ Value of A at $\pm\sqrt{v_0}, \pm\sqrt{v_1}, \ldots, \pm\sqrt{v_{n/2-1}}$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\qquad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\qquad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\qquad$ $A(-x) = E(x^2) - xO(x^2)$

Value of E, O at x^2 $\rightarrow$ value of A at $\pm x$ $\qquad$ in $O(1)$ time

Value of E, O at $x_0^2, x_1^2, \ldots, x_{n/2-1}^2$
$\rightarrow$ value of A at $\pm x_0, \pm x_1, \pm x_{n/2-1}$ $\qquad$ in $O(n)$ time

Alternatively: Value of E, O at $v_0, v_1, v_{n/2-1}$ $\qquad$ degree $n/2 - 1$
$\rightarrow$ Value of A at $\pm\sqrt{v_0}, \pm\sqrt{v_1}, \ldots, \pm\sqrt{v_{n/2-1}}$

Value of EE, OE, EO, OO at $w_0, w_1, \ldots, w_{n/4-1}$ $\qquad$ degree $n/4 - 1$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\quad$ $A(-x) = \sum_{i \text{ even}} a_i x^i - \sum_{i \text{ odd}} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\quad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\quad$ $A(-x) = E(x^2) - xO(x^2)$

Value of E, O at x^2 $\rightarrow$ value of A at $\pm x$ $\quad$ in $O(1)$ time

Value of E, O at $x_0^2, x_1^2, \ldots, x_{n/2-1}^2$
$\rightarrow$ value of A at $\pm x_0, \pm x_1, \pm x_{n/2-1}$ $\quad$ in $O(n)$ time

Alternatively: Value of E, O at $v_0, v_1, v_{n/2-1}$ $\quad$ degree $n/2 - 1$
$\rightarrow$ Value of A at $\pm\sqrt{v_0}, \pm\sqrt{v_1}, \ldots, \pm\sqrt{v_{n/2-1}}$

Value of EE, OE, EO, OO at $w_0, w_1, \ldots, w_{n/4-1}$ $\quad$ degree $n/4 - 1$
$\rightarrow$ Value of E, O at $\pm\sqrt{w_0}, \pm\sqrt{w_1}, \ldots, \pm\sqrt{w_{n/4-1}}$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\quad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\quad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\quad$ $A(-x) = E(x^2) - xO(x^2)$

Value of E, O at $x^2 \to$ value of A at $\pm x$ $\quad$ in $O(1)$ time

Value of E, O at $x_0^2, x_1^2, \ldots, x_{n/2-1}^2$
$\to$ value of A at $\pm x_0, \pm x_1, \pm x_{n/2-1}$ $\quad$ in $O(n)$ time

Alternatively: Value of E, O at $v_0, v_1, v_{n/2-1}$ $\quad$ degree $n/2 - 1$
$\to$ Value of A at $\pm\sqrt{v_0}, \pm\sqrt{v_1}, \ldots, \pm\sqrt{v_{n/2-1}}$

Value of EE, OE, EO, OO at $w_0, w_1, \ldots, w_{n/4-1}$ $\quad$ degree $n/4 - 1$
$\to$ Value of E, O at $\pm\sqrt{w_0}, \pm\sqrt{w_1}, \ldots, \pm\sqrt{w_{n/4-1}}$
$\to$ Value of A at 4th roots of each of $w_0, w_1, \ldots, w_{n/4-1}$

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\quad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\quad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\quad$ $A(-x) = E(x^2) - xO(x^2)$

Value of E, O at x^2 $\rightarrow$ value of A at $\pm x$ $\quad$ in $O(1)$ time

Value of E, O at $x_0^2, x_1^2, \ldots, x_{n/2-1}^2$
$\rightarrow$ value of A at $\pm x_0, \pm x_1, \pm x_{n/2-1}$ $\quad$ in $O(n)$ time

Alternatively: Value of E, O at $v_0, v_1, v_{n/2-1}$ $\quad$ degree $n/2 - 1$
$\rightarrow$ Value of A at $\pm\sqrt{v_0}, \pm\sqrt{v_1}, \ldots, \pm\sqrt{v_{n/2-1}}$

Value of EE, OE, EO, OO at $w_0, w_1, \ldots, w_{n/4-1}$ $\quad$ degree $n/4 - 1$
$\rightarrow$ Value of E, O at $\pm\sqrt{w_0}, \pm\sqrt{w_1}, \ldots, \pm\sqrt{w_{n/4-1}}$
$\rightarrow$ Value of A at 4th roots of each of $w_0, w_1, \ldots, w_{n/4-1}$

Value of n degree-0-polynomials at single point u_0

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\quad A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\quad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\quad A(-x) = E(x^2) - xO(x^2)$

Value of E, O at $x^2 \rightarrow$ value of A at $\pm x$ $\quad$ in $O(1)$ time

Value of E, O at $x_0^2, x_1^2, \ldots, x_{n/2-1}^2$
$\rightarrow$ value of A at $\pm x_0, \pm x_1, \pm x_{n/2-1}$ $\quad$ in $O(n)$ time

Alternatively: Value of E, O at $v_0, v_1, v_{n/2-1}$ $\quad$ degree $n/2 - 1$
$\rightarrow$ Value of A at $\pm\sqrt{v_0}, \pm\sqrt{v_1}, \ldots, \pm\sqrt{v_{n/2-1}}$

Value of EE, OE, EO, OO at $w_0, w_1, \ldots, w_{n/4-1}$ $\quad$ degree $n/4 - 1$
$\rightarrow$ Value of E, O at $\pm\sqrt{w_0}, \pm\sqrt{w_1}, \ldots, \pm\sqrt{w_{n/4-1}}$
$\rightarrow$ Value of A at 4th roots of each of $w_0, w_1, \ldots, w_{n/4-1}$

Value of n degree-0-polynomials at single point u_0
$\rightarrow \ldots \rightarrow$ Value of A at n nth roots of u_0.

Choosing the points at which to evaluate

$A(x) = \sum_i a_i x^i$ $\qquad$ $A(-x) = \sum_{i\ even} a_i x^i - \sum_{i\ odd} a_i x^i$

"even(A)": $E(y) = \sum_i a_{2i} y^i$ $\qquad$ "odd(A)": $O(y) = \sum_i a_{2i+1} y^i$

$A(x) = E(x^2) + xO(x^2)$ $\qquad$ $A(-x) = E(x^2) - xO(x^2)$

Value of E, O at $x^2 \to$ value of A at $\pm x$ $\qquad$ in $O(1)$ time

Value of E, O at $x_0^2, x_1^2, \ldots, x_{n/2-1}^2$
$\to$ value of A at $\pm x_0, \pm x_1, \pm x_{n/2-1}$ $\qquad$ in $O(n)$ time

Alternatively: Value of E, O at $v_0, v_1, v_{n/2-1}$ $\qquad$ degree $n/2 - 1$
$\to$ Value of A at $\pm\sqrt{v_0}, \pm\sqrt{v_1}, \ldots, \pm\sqrt{v_{n/2-1}}$

Value of EE, OE, EO, OO at $w_0, w_1, \ldots, w_{n/4-1}$ $\qquad$ degree $n/4 - 1$
$\to$ Value of E, O at $\pm\sqrt{w_0}, \pm\sqrt{w_1}, \ldots, \pm\sqrt{w_{n/4-1}}$
$\to$ Value of A at 4th roots of each of $w_0, w_1, \ldots, w_{n/4-1}$

Value of n degree-0-polynomials at single point u_0
$\to \ldots \to$ Value of A at n nth roots of u_0. $\qquad$ Choose $u_0 = 1$

On complex roots of 1

On complex roots of 1

$\omega_n = e^{2\pi\sqrt{-1}/n}$ is a principal nth root of 1.

On complex roots of 1

$\omega_n = e^{2\pi\sqrt{-1}/n}$ is a principal nth root of 1.

Let $\omega = \omega_n$

On complex roots of 1

$\omega_n = e^{2\pi\sqrt{-1}/n}$ is a principal nth root of 1.

Let $\omega = \omega_n$

$\omega^0, \omega^1, \omega^2, \ldots, \omega^{n-1}$ are all nth roots of 1.

On complex roots of 1

$\omega_n = e^{2\pi\sqrt{-1}/n}$ is a principal nth root of 1.

Let $\omega = \omega_n$

$\omega^0, \omega^1, \omega^2, \ldots, \omega^{n-1}$ are all nth roots of 1.

A is evaluated at $\omega^0, \ldots, \omega^{n-1}$

On complex roots of 1

$\omega_n = e^{2\pi\sqrt{-1}/n}$ is a principal nth root of 1.

Let $\omega = \omega_n$

$\omega^0, \omega^1, \omega^2, \ldots, \omega^{n-1}$ are all nth roots of 1.

A is evaluated at $\omega^0, \ldots, \omega^{n-1}$

Note that second half of sequence = negative of first half

On complex roots of 1

$\omega_n = e^{2\pi\sqrt{-1}/n}$ is a principal nth root of 1.

Let $\omega = \omega_n$

$\omega^0, \omega^1, \omega^2, \ldots, \omega^{n-1}$ are all nth roots of 1.

A is evaluated at $\omega^0, \ldots, \omega^{n-1}$

Note that second half of sequence = negative of first half

E, O are evaluated at $(\omega^0)^2, \ldots, (\omega^i)^2, \ldots, (\omega^{n/2-1})^2$

On complex roots of 1

$\omega_n = e^{2\pi\sqrt{-1}/n}$ is a principal nth root of 1.

Let $\omega = \omega_n$

$\omega^0, \omega^1, \omega^2, \ldots, \omega^{n-1}$ are all nth roots of 1.

A is evaluated at $\omega^0, \ldots, \omega^{n-1}$

Note that second half of sequence = negative of first half

E, O are evaluated at $(\omega^0)^2, \ldots, (\omega^i)^2, \ldots, (\omega^{n/2-1})^2$

$(\omega^i)^2 = (e^{2\pi\sqrt{-1}/n})^{2i} = (e^{2\pi\sqrt{-1}/(n/2)})^i = \omega^i_{n/2}$

On complex roots of 1

$\omega_n = e^{2\pi\sqrt{-1}/n}$ is a principal nth root of 1.

Let $\omega = \omega_n$

$\omega^0, \omega^1, \omega^2, \ldots, \omega^{n-1}$ are all nth roots of 1.

A is evaluated at $\omega^0, \ldots, \omega^{n-1}$

Note that second half of sequence = negative of first half

E, O are evaluated at $(\omega^0)^2, \ldots, (\omega^i)^2, \ldots, (\omega^{n/2-1})^2$

$(\omega^i)^2 = (e^{2\pi\sqrt{-1}/n})^{2i} = (e^{2\pi\sqrt{-1}/(n/2)})^i = \omega_{n/2}^i$

E, O are evaluated at powers of $\omega_{n/2} = \omega^2$.

On complex roots of 1

$\omega_n = e^{2\pi\sqrt{-1}/n}$ is a principal nth root of 1.

Let $\omega = \omega_n$

$\omega^0, \omega^1, \omega^2, \ldots, \omega^{n-1}$ are all nth roots of 1.

A is evaluated at $\omega^0, \ldots, \omega^{n-1}$

Note that second half of sequence = negative of first half

E, O are evaluated at $(\omega^0)^2, \ldots, (\omega^i)^2, \ldots, (\omega^{n/2-1})^2$

$(\omega^i)^2 = (e^{2\pi\sqrt{-1}/n})^{2i} = (e^{2\pi\sqrt{-1}/(n/2)})^i = \omega^i_{n/2}$

E, O are evaluated at powers of $\omega_{n/2} = \omega^2$.

E, O can be evaluated recursively: $n/2 - 1$ degree polynomial to be evaluated at $n/2$ $n/2$th roots of 1.

Evaluate$(a_0, a_1, \ldots, a_{n-1}, \omega)\{$

Evaluate($a_0, a_1, \ldots, a_{n-1}, \omega$){

// Evaluate A at $\omega^0, \ldots, \omega^{n-1}$

Evaluate($a_0, a_1, \ldots, a_{n-1}, \omega$){

// Evaluate A at $\omega^0, \ldots, \omega^{n-1}$
// $\omega^{n/2+i} = -\omega^i$ for $i = 0, \ldots, n/2 - 1$.

Evaluate$(a_0, a_1, \ldots, a_{n-1}, \omega)\{$

// Evaluate A at $\omega^0, \ldots, \omega^{n-1}$
// $\omega^{n/2+i} = -\omega^i$ for $i = 0, \ldots, n/2 - 1$.

1. Base case: if $n = 1$ return a_0
2. $E =$ Evaluate$(a_0, a_2, \ldots, a_{n-2}, \omega^2)$
3. $O =$ Evaluate$(a_1, a_3, \ldots, a_{n-1}, \omega^2)$
4. for $i = 0$ to $n/2 - 1$
5. $\quad A[i] = E[i] + \omega^i O[i] \qquad A[n/2 + i] = E[i] - \omega^i O[i]$
6. end for
7. Return A // array

Evaluate($a_0, a_1, \ldots, a_{n-1}, \omega$){

// Evaluate A at $\omega^0, \ldots, \omega^{n-1}$
// $\omega^{n/2+i} = -\omega^i$ for $i = 0, \ldots, n/2 - 1$.

1. Base case: if $n = 1$ return a_0
2. E = Evaluate($a_0, a_2, \ldots, a_{n-2}, \omega^2$)
3. O = Evaluate($a_1, a_3, \ldots, a_{n-1}, \omega^2$)
4. for $i = 0$ to $n/2 - 1$
5. $\quad A[i] = E[i] + \omega^i O[i] \qquad A[n/2 + i] = E[i] - \omega^i O[i]$
6. end for
7. Return A // array

$$T(n) = 2T(n/2) + O(n)$$

Evaluate($a_0, a_1, \ldots, a_{n-1}, \omega$){

// Evaluate A at $\omega^0, \ldots, \omega^{n-1}$
// $\omega^{n/2+i} = -\omega^i$ for $i = 0, \ldots, n/2 - 1$.

1. Base case: if $n = 1$ return a_0
2. $E =$ Evaluate($a_0, a_2, \ldots, a_{n-2}, \omega^2$)
3. $O =$ Evaluate($a_1, a_3, \ldots, a_{n-1}, \omega^2$)
4. for $i = 0$ to $n/2 - 1$
5. $\quad A[i] = E[i] + \omega^i O[i] \qquad A[n/2 + i] = E[i] - \omega^i O[i]$
6. end for
7. Return A // array

$$T(n) = 2T(n/2) + O(n)$$

So $T(n) = O(n \log n)$

Polynomial Evaluation as matrix vector multiplication

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

Polynomial Evaluation as matrix vector multiplication

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

Polynomial Evaluation as matrix vector multiplication

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

$$\begin{bmatrix} \vdots \\ A(\omega^i) \\ \vdots \end{bmatrix} = \begin{bmatrix} & & & \vdots & & \\ \omega^0 & \omega^i & \dots & \omega^{ij} & \dots & \omega^{i(n-1)} \\ & & & \vdots & & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_j \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Polynomial Evaluation as matrix vector multiplication

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

$$\begin{bmatrix} \vdots \\ A(\omega^i) \\ \vdots \end{bmatrix} = \begin{bmatrix} & & & \vdots & & \\ \omega^0 & \omega^i & \dots & \omega^{ij} & \dots & \omega^{i(n-1)} \\ & & & \vdots & & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_j \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Interpolation = multiplying lhs by inverse of matrix, $M(\omega)$.

Polynomial Evaluation as matrix vector multiplication

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

$$\begin{bmatrix} \vdots \\ A(\omega^i) \\ \vdots \end{bmatrix} = \begin{bmatrix} & & & \vdots & & \\ \omega^0 & \omega^i & \dots & \omega^{ij} & \dots & \omega^{i(n-1)} \\ & & & \vdots & & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_j \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Interpolation = multiplying lhs by inverse of matrix, $M(\omega)$.
Matrix $M(\omega)$ has orthogonal columns!

Next

Polynomial Evaluation as matrix vector multiplication

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

$$\begin{bmatrix} \vdots \\ A(\omega^i) \\ \vdots \end{bmatrix} = \begin{bmatrix} & & & \vdots & & \\ \omega^0 & \omega^i & \dots & \omega^{ij} & \dots & \omega^{i(n-1)} \\ & & & \vdots & & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_j \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Interpolation = multiplying lhs by inverse of matrix, $M(\omega)$.

Matrix $M(\omega)$ has orthogonal columns! Next

Each column has norm $\sqrt{n}$ Next

Polynomial Evaluation as matrix vector multiplication

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

$$\begin{bmatrix} \vdots \\ A(\omega^i) \\ \vdots \end{bmatrix} = \begin{bmatrix} & & & \vdots & & \\ \omega^0 & \omega^i & \dots & \omega^{ij} & \dots & \omega^{i(n-1)} \\ & & & \vdots & & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_j \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Interpolation = multiplying lhs by inverse of matrix, $M(\omega)$.

Matrix $M(\omega)$ has orthogonal columns! Next

Each column has norm $\sqrt{n}$ Next

Matrix inverse $= \frac{1}{n} M(\omega)^* = \frac{1}{n} M(\omega^{-1})$ Next

Polynomial Evaluation as matrix vector multiplication

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

$$\begin{bmatrix} \vdots \\ A(\omega^i) \\ \vdots \end{bmatrix} = \begin{bmatrix} & & & \vdots & & \\ \omega^0 & \omega^i & \dots & \omega^{ij} & \dots & \omega^{i(n-1)} \\ & & & \vdots & & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_j \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Interpolation = multiplying lhs by inverse of matrix, $M(\omega)$.

Matrix $M(\omega)$ has orthogonal columns! Next

Each column has norm $\sqrt{n}$ Next

Matrix inverse $= \frac{1}{n}M(\omega)^* = \frac{1}{n}M(\omega^{-1})$ Next

Multiplication by inverse is similar to multiplication by original.

Polynomial Evaluation as matrix vector multiplication

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

$$\begin{bmatrix} \vdots \\ A(\omega^i) \\ \vdots \end{bmatrix} = \begin{bmatrix} & & & \vdots & & \\ \omega^0 & \omega^i & \dots & \omega^{ij} & \dots & \omega^{i(n-1)} \\ & & & \vdots & & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_j \\ \vdots \\ a_{n-1} \end{bmatrix}$$

Interpolation = multiplying lhs by inverse of matrix, $M(\omega)$.
Matrix $M(\omega)$ has orthogonal columns! Next
Each column has norm $\sqrt{n}$ Next

Matrix inverse $= \frac{1}{n}M(\omega)^* = \frac{1}{n}M(\omega^{-1})$ Next

Multiplication by inverse is similar to multiplication by original.
Interpolation time is also $O(n \log n)$.

Conjugation, orthogonality and norm

ith row of $M(\omega)$ is: $R_i = [\ldots \omega^{ik} \ldots]$

Conjugation, orthogonality and norm

ith row of $M(\omega)$ is: $R_i = [\ldots \omega^{ik} \ldots]$

To find its norm, we have to multiply by conjugate transpose

Conjugation, orthogonality and norm

ith row of $M(\omega)$ is: $R_i = [\dots\omega^{ik}\dots]$

To find its norm, we have to multiply by conjugate transpose
Conjugate of ω is ω^{-1} Argand diagram

Conjugation, orthogonality and norm

ith row of $M(\omega)$ is: $R_i = [\ldots \omega^{ik} \ldots]$

To find its norm, we have to multiply by conjugate transpose
Conjugate of ω is ω^{-1} Argand diagram
Conjugate of ω^{ik} is ω^{-ik}.

Conjugation, orthogonality and norm

ith row of $M(\omega)$ is: $R_i = [\ldots \omega^{ik} \ldots]$

To find its norm, we have to multiply by conjugate transpose
Conjugate of ω is ω^{-1} Argand diagram
Conjugate of ω^{ik} is ω^{-ik}.

$R_i R_i^* = \sum_k \omega^{ik}(\omega^{-ik}) = n$

Conjugation, orthogonality and norm

ith row of $M(\omega)$ is: $R_i = [\ldots \omega^{ik} \ldots]$

To find its norm, we have to multiply by conjugate transpose
Conjugate of ω is ω^{-1} Argand diagram
Conjugate of ω^{ik} is ω^{-ik}.

$R_i R_i^* = \sum_k \omega^{ik}(\omega^{-ik}) = n$

Hence norm $= \sqrt{n}$.

Conjugation, orthogonality and norm

ith row of $M(\omega)$ is: $R_i = [\ldots \omega^{ik} \ldots]$

To find its norm, we have to multiply by conjugate transpose
Conjugate of ω is ω^{-1} Argand diagram
Conjugate of ω^{ik} is ω^{-ik}.

$R_i R_i^* = \sum_k \omega^{ik}(\omega^{-ik}) = n$

Hence norm $= \sqrt{n}$.

For $i \neq j$: $R_i R_j^* = \sum_k \omega^{ik}(\omega^{-jk}) = \sum_{k=0}^{n-1} \omega^{(i-j)k}$

Conjugation, orthogonality and norm

ith row of $M(\omega)$ is: $R_i = [\ldots\omega^{ik}\ldots]$

To find its norm, we have to multiply by conjugate transpose
Conjugate of ω is ω^{-1} Argand diagram
Conjugate of ω^{ik} is ω^{-ik}.

$R_i R_i^* = \sum_k \omega^{ik}(\omega^{-ik}) = n$

Hence norm $= \sqrt{n}$.

For $i \neq j$: $R_i R_j^* = \sum_k \omega^{ik}(\omega^{-jk}) = \sum_{k=0}^{n-1} \omega^{(i-j)k}$
$= \frac{(\omega^{i-j})^n - 1}{\omega^{i-j} - 1}$ Sum of geometric series

Conjugation, orthogonality and norm

ith row of $M(\omega)$ is: $R_i = [\ldots \omega^{ik} \ldots]$

To find its norm, we have to multiply by conjugate transpose
Conjugate of ω is ω^{-1} Argand diagram
Conjugate of ω^{ik} is ω^{-ik}.

$R_i R_i^* = \sum_k \omega^{ik}(\omega^{-ik}) = n$

Hence norm $= \sqrt{n}$.

For $i \neq j$: $R_i R_j^* = \sum_k \omega^{ik}(\omega^{-jk}) = \sum_{k=0}^{n-1} \omega^{(i-j)k}$
$= \frac{(\omega^{i-j})^n - 1}{\omega^{i-j} - 1}$ Sum of geometric series
$= 0$ $\omega^{in} = \omega^{jn} = 1$

Conjugation, orthogonality and norm

ith row of $M(\omega)$ is: $R_i = [\ldots \omega^{ik} \ldots]$

To find its norm, we have to multiply by conjugate transpose
Conjugate of ω is ω^{-1} Argand diagram
Conjugate of ω^{ik} is ω^{-ik}.

$R_i R_i^* = \sum_k \omega^{ik}(\omega^{-ik}) = n$

Hence norm $= \sqrt{n}$.

For $i \neq j$: $R_i R_j^* = \sum_k \omega^{ik}(\omega^{-jk}) = \sum_{k=0}^{n-1} \omega^{(i-j)k}$
$= \frac{(\omega^{i-j})^n - 1}{\omega^{i-j} - 1}$ Sum of geometric series
$= 0$ $\omega^{in} = \omega^{jn} = 1$

Hence orthogonal.

Remarks

Remarks

- Values at roots of unity = Discrete Fourier Transform of coefficients.

Remarks

- Values at roots of unity = Discrete Fourier Transform of coefficients.
- Algorithm is called FFT : Fast Fourier Transform. Cooley-Tukey 1965.

Remarks

- Values at roots of unity = Discrete Fourier Transform of coefficients.
- Algorithm is called FFT : Fast Fourier Transform.
  Cooley-Tukey 1965.
  Like many great ideas, it was apparently known to Gauss!

Remarks

- Values at roots of unity = Discrete Fourier Transform of coefficients.
- Algorithm is called FFT : Fast Fourier Transform.
  Cooley-Tukey 1965.
  Like many great ideas, it was apparently known to Gauss!
- Many ideas. Multiple representations. Exercising your freedom carefully (where to evaluate). Divide and conquer.

Remarks

- Values at roots of unity $=$ Discrete Fourier Transform of coefficients.
- Algorithm is called FFT : Fast Fourier Transform.
  Cooley-Tukey 1965.
  Like many great ideas, it was apparently known to Gauss!
- Many ideas. Multiple representations. Exercising your freedom carefully (where to evaluate). Divide and conquer.
- Version exists for ring of integers modulo a number. Useful for doing arithmetic.

Remarks

- Values at roots of unity = Discrete Fourier Transform of coefficients.
- Algorithm is called FFT : Fast Fourier Transform.
  Cooley-Tukey 1965.
  Like many great ideas, it was apparently known to Gauss!
- Many ideas. Multiple representations. Exercising your freedom carefully (where to evaluate). Divide and conquer.
- Version exists for ring of integers modulo a number. Useful for doing arithmetic.
- Reading: [Dasgupta, Papadimitriou, Vazirani].