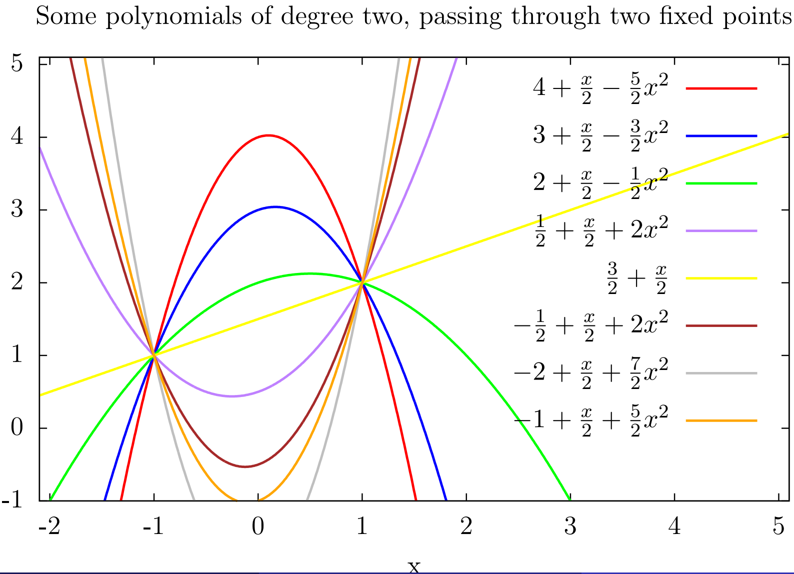
Fast Fourier Transform

Lecture 19

October 30, 2014

Polynomials and point value pairs



Multiplying polynomials quickly

Definition

polynomial $p(x)$ of degree n:a function
$p(x) = \sum_{j=0}^{n} a_j x^j = a_0 + x(a_1 + x(a_2 + \ldots + xa_n))$.

x_0: $p(x_0)$ can be computed in $O(n)$ time.
"dual" (and equivalent) representation...

Theorem

For any set $\left\{ (x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1}) \right\}$ *of* n ***point-value pairs*** *such that all the* x_k *values are distinct, there is a unique polynomial* $p(x)$ *of degree* $n - 1$*, such that* $y_k = p(x_k)$*, for* $k = 0, \ldots, n - 1$.

Polynomial via point-value

$\left\{(x_0, y_0), (x_1, y_1), (x_2, y_2)\right\}$: polynomial through points:

$$
\begin{aligned}
p(x) = {} & y_0 \frac{\cancel{(x - x_0)}(x - x_1)(x - x_2)}{\cancel{(x_0 - x_0)}(x_0 - x_1)(x_0 - x_2)} \\
& + y_1 \frac{(x - x_0)\cancel{(x - x_1)}(x - x_2)}{(x_1 - x_0)\cancel{(x_1 - x_1)}(x_1 - x_2)} \\
& + y_2 \frac{(x - x_0)(x - x_1)\cancel{(x - x_2)}}{(x_2 - x_0)(x_2 - x_1)\cancel{(x_2 - x_2)}}
\end{aligned}
$$

Polynomial via point-value

$\left\{(x_0, y_0), (x_1, y_1), (x_2, y_2)\right\}$: polynomial through points:

$$
\begin{aligned}
p(x) = {} & y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} \\
& + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} \\
& + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}
\end{aligned}
$$

Polynomial via point-value

$\Big\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\Big\}$: polynomial through points:

$$p(x) = \sum_{i=0}^{n-1} y_i \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)}.$$

ith is zero for $x = x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n-1}$, and is equal to y_i for $x = x_i$.

Polynomials: regular vs. point-value pair representation

Just because.

1. Given n point-value pairs. Can compute $p(x)$ in $O(n^2)$ time.
2. Point-value pairs representation: Multiply polynomials quickly!
3. p, q polynomial of degree $n-1$, both represented by $2n$ point-value pairs

$$\left\{(x_0, y_0), (x_1, y_1), \ldots, (x_{2n-1}, y_{2n-1})\right\} \quad \text{for} \quad p(x),$$

$$\text{and} \; \left\{(x_0, y'_0), (x_1, y'_1), \ldots, (x_{2n-1}, y'_{2n-1})\right\} \quad \text{for} \quad q(x).$$

4. $r(x) = p(x)q(x)$: product.

Polynomials: regular vs. point-value pair representation

Just because.

1. Given n point-value pairs. Can compute $p(x)$ in $O(n^2)$ time.
2. Point-value pairs representation: Multiply polynomials quickly!
3. p, q polynomial of degree $n-1$, both represented by $2n$ point-value pairs

$$\left\{(x_0, y_0), (x_1, y_1), \ldots, (x_{2n-1}, y_{2n-1})\right\} \text{ for } p(x),$$

$$\text{and } \left\{(x_0, y'_0), (x_1, y'_1), \ldots, (x_{2n-1}, y'_{2n-1})\right\} \text{ for } q(x).$$

4. $r(x) = p(x)q(x)$: product.

Polynomials: regular vs. point-value pair representation

Just because.

1. Given n point-value pairs. Can compute $p(x)$ in $O(n^2)$ time.
2. Point-value pairs representation: Multiply polynomials quickly!
3. p, q polynomial of degree $n-1$, both represented by $2n$ point-value pairs

$$\left\{ (x_0, y_0), (x_1, y_1), \ldots, (x_{2n-1}, y_{2n-1}) \right\} \quad \text{for} \quad p(x),$$

$$\text{and} \quad \left\{ (x_0, y'_0), (x_1, y'_1), \ldots, (x_{2n-1}, y'_{2n-1}) \right\} \quad \text{for} \quad q(x).$$

4. $r(x) = p(x)q(x)$: product.

Polynomials: regular vs. point-value pair representation

Just because.

1. Given n point-value pairs. Can compute $p(x)$ in $O(n^2)$ time.
2. Point-value pairs representation: Multiply polynomials quickly!
3. p, q polynomial of degree $n-1$, both represented by $2n$ point-value pairs

$$\left\{(x_0, y_0), (x_1, y_1), \ldots, (x_{2n-1}, y_{2n-1})\right\} \text{ for } p(x),$$

$$\text{and } \left\{(x_0, y'_0), (x_1, y'_1), \ldots, (x_{2n-1}, y'_{2n-1})\right\} \text{ for } q(x).$$

4. $r(x) = p(x)q(x)$: product.

Polynomials: regular vs. point-value pair representation

Just because.

1. In point-value representation representation of $r(x)$ is

$$\left\{ (x_0, r(x_0)), \ldots, (x_{2n-1}, r(x_{2n-1})) \right\}$$

$$= \left\{ \Big(x_0, p(x_0) q(x_0) \Big), \ldots, \Big(x_{2n-1}, p(x_{2n-1}) q(x_{2n-1}) \Big) \right\}$$

$$= \left\{ (x_0, y_0 y'_0), \ldots, (x_{2n-1}, y_{2n-1} y'_{2n-1}) \right\}.$$

Which implies...

1. $p(x)$ and $q(x)$: point-value pairs $\implies$ compute $r(x) = p(x)q(x)$ in linear time!
2. ...but $r(x)$ is in point-value representation. Bummer.
3. ...but we can compute $r(x)$ from this representation.
4. Purpose: Translate quickly (i.e., $O(n \log n)$ time) from the standard r to point-value pairs representation of polynomials.
5. ...and back!
6. $\implies$ computing product of two polynomials in $O(n \log n)$ time.
7. ***Fast Fourier Transform*** is a way to do this.
8. choosing the x_i values carefully, and using divide and conquer.

Which implies...

1. $p(x)$ and $q(x)$: point-value pairs $\implies$ compute $r(x) = p(x)q(x)$ in linear time!
2. ...but $r(x)$ is in point-value representation. Bummer.
3. ...but we can compute $r(x)$ from this representation.
4. Purpose: Translate quickly (i.e., $O(n \log n)$ time) from the standard r to point-value pairs representation of polynomials.
5. ...and back!
6. $\implies$ computing product of two polynomials in $O(n \log n)$ time.
7. ***Fast Fourier Transform*** is a way to do this.
8. choosing the x_i values carefully, and using divide and conquer.

Which implies...

1. $p(x)$ and $q(x)$: point-value pairs $\implies$ compute $r(x) = p(x)q(x)$ in linear time!
2. ...but $r(x)$ is in point-value representation. Bummer.
3. ...but we can compute $r(x)$ from this representation.
4. Purpose: Translate quickly (i.e., $O(n \log n)$ time) from the standard r to point-value pairs representation of polynomials.
5. ...and back!
6. $\implies$ computing product of two polynomials in $O(n \log n)$ time.
7. ***Fast Fourier Transform*** is a way to do this.
8. choosing the x_i values carefully, and using divide and conquer.

Which implies...

1. $p(x)$ and $q(x)$: point-value pairs $\implies$ compute $r(x) = p(x)q(x)$ in linear time!
2. ...but $r(x)$ is in point-value representation. Bummer.
3. ...but we can compute $r(x)$ from this representation.
4. Purpose: Translate quickly (i.e., $O(n \log n)$ time) from the standard r to point-value pairs representation of polynomials.
5. ...and back!
6. $\implies$ computing product of two polynomials in $O(n \log n)$ time.
7. ***Fast Fourier Transform*** is a way to do this.
8. choosing the x_i values carefully, and using divide and conquer.

Which implies...

1. $p(x)$ and $q(x)$: point-value pairs $\implies$ compute $r(x) = p(x)q(x)$ in linear time!
2. ...but $r(x)$ is in point-value representation. Bummer.
3. ...but we can compute $r(x)$ from this representation.
4. Purpose: Translate quickly (i.e., $O(n \log n)$ time) from the standard r to point-value pairs representation of polynomials.
5. ...and back!
6. $\implies$ computing product of two polynomials in $O(n \log n)$ time.
7. ***Fast Fourier Transform*** is a way to do this.
8. choosing the x_i values carefully, and using divide and conquer.

Which implies...

1. $p(x)$ and $q(x)$: point-value pairs $\implies$ compute $r(x) = p(x)q(x)$ in linear time!
2. ...but $r(x)$ is in point-value representation. Bummer.
3. ...but we can compute $r(x)$ from this representation.
4. Purpose: Translate quickly (i.e., $O(n \log n)$ time) from the standard r to point-value pairs representation of polynomials.
5. ...and back!
6. $\implies$ computing product of two polynomials in $O(n \log n)$ time.
7. ***Fast Fourier Transform*** is a way to do this.
8. choosing the x_i values carefully, and using divide and conquer.

Which implies...

1. $p(x)$ and $q(x)$: point-value pairs $\implies$ compute $r(x) = p(x)q(x)$ in linear time!
2. ...but $r(x)$ is in point-value representation. Bummer.
3. ...but we can compute $r(x)$ from this representation.
4. Purpose: Translate quickly (i.e., $O(n \log n)$ time) from the standard r to point-value pairs representation of polynomials.
5. ...and back!
6. $\implies$ computing product of two polynomials in $O(n \log n)$ time.
7. ***Fast Fourier Transform*** is a way to do this.
8. choosing the x_i values carefully, and using divide and conquer.

Which implies...

1. $p(x)$ and $q(x)$: point-value pairs $\implies$ compute $r(x) = p(x)q(x)$ in linear time!
2. ...but $r(x)$ is in point-value representation. Bummer.
3. ...but we can compute $r(x)$ from this representation.
4. Purpose: Translate quickly (i.e., $O(n \log n)$ time) from the standard r to point-value pairs representation of polynomials.
5. ...and back!
6. $\implies$ computing product of two polynomials in $O(n \log n)$ time.
7. ***Fast Fourier Transform*** is a way to do this.
8. choosing the x_i values carefully, and using divide and conquer.

Part I

Computing a polynomial quickly on n values

Computing a polynomial quickly on n values

Lets just use some magic.

1. Assume: polynomials have degree $n-1$, where $n = 2^k$.
2. .. pad polynomials with terms having zero coefficients.
3. ***Magic set*** of numbers: $\Psi = \{x_1, \ldots, x_n\}$.
   Property: $|\mathbf{SQ}(\Psi)| = n/2$, where $\mathbf{SQ}(\Psi) = \left\{ x^2 \;\middle|\; x \in \Psi \right\}$.
4. $|\mathrm{square}(\Psi)| = |\Psi|/2$.
5. Easy to find such set...
6. Magic: Have this property repeatedly...
   $\mathbf{SQ}(\mathbf{SQ}(\Psi))$ has $n/4$ distinct values.
7. $\mathbf{SQ}(\mathbf{SQ}(\mathbf{SQ}(\Psi)))$ has $n/8$ values.
8. $\mathbf{SQ}^i(\Psi)$ has $n/2^i$ distinct values.
9. Oops: No such set of real numbers.
10. NO SUCH SET.

Computing a polynomial quickly on n values

Lets just use some magic.

1. Assume: polynomials have degree $n-1$, where $n=2^k$.
2. .. pad polynomials with terms having zero coefficients.
3. ***Magic set*** of numbers: $\Psi = \{x_1, \ldots, x_n\}$.
   Property: $|\mathbf{SQ}(\Psi)| = n/2$, where $\mathbf{SQ}(\Psi) = \left\{ x^2 \;\middle|\; x \in \Psi \right\}$.
4. $|\mathrm{square}(\Psi)| = |\Psi|/2$.
5. Easy to find such set...
6. Magic: Have this property repeatedly...
   $\mathbf{SQ}(\mathbf{SQ}(\Psi))$ has $n/4$ distinct values.
7. $\mathbf{SQ}(\mathbf{SQ}(\mathbf{SQ}(\Psi)))$ has $n/8$ values.
8. $\mathbf{SQ}^i(\Psi)$ has $n/2^i$ distinct values.
9. Oops: No such set of real numbers.
10. NO SUCH SET.

Computing a polynomial quickly on n values

Lets just use some magic.

1. Assume: polynomials have degree $n-1$, where $n = 2^k$.
2. .. pad polynomials with terms having zero coefficients.
3. ***Magic set*** of numbers: $\Psi = \{x_1, \ldots, x_n\}$.
   Property: $|\mathbf{SQ}(\Psi)| = n/2$, where $\mathbf{SQ}(\Psi) = \left\{ x^2 \,\middle|\, x \in \Psi \right\}$.
4. $|\mathrm{square}(\Psi)| = |\Psi|/2$.
5. Easy to find such set...
6. Magic: Have this property repeatedly...
   $\mathbf{SQ}(\mathbf{SQ}(\Psi))$ has $n/4$ distinct values.
7. $\mathbf{SQ}(\mathbf{SQ}(\mathbf{SQ}(\Psi)))$ has $n/8$ values.
8. $\mathbf{SQ}^i(\Psi)$ has $n/2^i$ distinct values.
9. Oops: No such set of real numbers.
10. NO SUCH SET.

Computing a polynomial quickly on n values

Lets just use some magic.

1. Assume: polynomials have degree $n - 1$, where $n = 2^k$.
2. .. pad polynomials with terms having zero coefficients.
3. ***Magic set*** of numbers: $\Psi = \{x_1, \ldots, x_n\}$.
   Property: $|\mathbf{SQ}(\Psi)| = n/2$, where $\mathbf{SQ}(\Psi) = \left\{ x^2 \,\middle|\, x \in \Psi \right\}$.
4. $|\mathrm{square}(\Psi)| = |\Psi|/2$.
5. Easy to find such set...
6. Magic: Have this property repeatedly...
   $\mathbf{SQ}(\mathbf{SQ}(\Psi))$ has $n/4$ distinct values.
7. $\mathbf{SQ}(\mathbf{SQ}(\mathbf{SQ}(\Psi)))$ has $n/8$ values.
8. $\mathbf{SQ}^i(\Psi)$ has $n/2^i$ distinct values.
9. Oops: No such set of real numbers.
10. NO SUCH SET.

Computing a polynomial quickly on n values

Lets just use some magic.

1. Assume: polynomials have degree $n-1$, where $n = 2^k$.
2. .. pad polynomials with terms having zero coefficients.
3. ***Magic set*** of numbers: $\Psi = \{x_1, \ldots, x_n\}$.
   Property: $|\mathbf{SQ}(\Psi)| = n/2$, where $\mathbf{SQ}(\Psi) = \left\{ x^2 \;\middle|\; x \in \Psi \right\}$.
4. $|\mathrm{square}(\Psi)| = |\Psi|/2$.
5. Easy to find such set...
6. Magic: Have this property repeatedly...
   $\mathbf{SQ}(\mathbf{SQ}(\Psi))$ has $n/4$ distinct values.
7. $\mathbf{SQ}(\mathbf{SQ}(\mathbf{SQ}(\Psi)))$ has $n/8$ values.
8. $\mathbf{SQ}^i(\Psi)$ has $n/2^i$ distinct values.
9. Oops: No such set of real numbers.
10. NO SUCH SET.

Computing a polynomial quickly on n values

Lets just use some magic.

1. Assume: polynomials have degree $n-1$, where $n = 2^k$.
2. .. pad polynomials with terms having zero coefficients.
3. ***Magic set*** of numbers: $\Psi = \{x_1, \ldots, x_n\}$.
   Property: $|\mathsf{SQ}(\Psi)| = n/2$, where $\mathsf{SQ}(\Psi) = \left\{ x^2 \;\middle|\; x \in \Psi \right\}$.
4. $|\mathrm{square}(\Psi)| = |\Psi|/2$.
5. Easy to find such set...
6. Magic: Have this property repeatedly...
   $\mathsf{SQ}(\mathsf{SQ}(\Psi))$ has $n/4$ distinct values.
7. $\mathsf{SQ}(\mathsf{SQ}(\mathsf{SQ}(\Psi)))$ has $n/8$ values.
8. $\mathsf{SQ}^i(\Psi)$ has $n/2^i$ distinct values.
9. Oops: No such set of real numbers.
10. NO SUCH SET.

Computing a polynomial quickly on n values

Lets just use some magic.

1. Assume: polynomials have degree $n - 1$, where $n = 2^k$.
2. .. pad polynomials with terms having zero coefficients.
3. ***Magic set*** of numbers: $\Psi = \{x_1, \ldots, x_n\}$.
   Property: $|\mathbf{SQ}(\Psi)| = n/2$, where $\mathbf{SQ}(\Psi) = \left\{ x^2 \,\middle|\, x \in \Psi \right\}$.
4. $|\mathrm{square}(\Psi)| = |\Psi|/2$.
5. Easy to find such set...
6. Magic: Have this property repeatedly...
   $\mathbf{SQ}(\mathbf{SQ}(\Psi))$ has $n/4$ distinct values.
7. $\mathbf{SQ}(\mathbf{SQ}(\mathbf{SQ}(\Psi)))$ has $n/8$ values.
8. $\mathbf{SQ}^i(\Psi)$ has $n/2^i$ distinct values.
9. Oops: No such set of real numbers.
10. NO SUCH SET.

Computing a polynomial quickly on n values

Lets just use some magic.

1. Assume: polynomials have degree $n - 1$, where $n = 2^k$.
2. .. pad polynomials with terms having zero coefficients.
3. ***Magic set*** of numbers: $\Psi = \{x_1, \ldots, x_n\}$.
   Property: $|\mathsf{SQ}(\Psi)| = n/2$, where $\mathsf{SQ}(\Psi) = \left\{ x^2 \,\middle|\, x \in \Psi \right\}$.
4. $|\mathrm{square}(\Psi)| = |\Psi| / 2$.
5. Easy to find such set...
6. Magic: Have this property repeatedly...
   $\mathsf{SQ}(\mathsf{SQ}(\Psi))$ has $n/4$ distinct values.
7. $\mathsf{SQ}(\mathsf{SQ}(\mathsf{SQ}(\Psi)))$ has $n/8$ values.
8. $\mathsf{SQ}^i(\Psi)$ has $n/2^i$ distinct values.
9. Oops: No such set of real numbers.
10. NO SUCH SET.

Computing a polynomial quickly on n values

Lets just use some magic.

1. Assume: polynomials have degree $n-1$, where $n = 2^k$.
2. .. pad polynomials with terms having zero coefficients.
3. ***Magic set*** of numbers: $\Psi = \{x_1, \ldots, x_n\}$.
   Property: $|\mathbf{SQ}(\Psi)| = n/2$, where $\mathbf{SQ}(\Psi) = \left\{ x^2 \;\middle|\; x \in \Psi \right\}$.
4. $|\mathrm{square}(\Psi)| = |\Psi|/2$.
5. Easy to find such set...
6. Magic: Have this property repeatedly...
   $\mathbf{SQ}(\mathbf{SQ}(\Psi))$ has $n/4$ distinct values.
7. $\mathbf{SQ}(\mathbf{SQ}(\mathbf{SQ}(\Psi)))$ has $n/8$ values.
8. $\mathbf{SQ}^i(\Psi)$ has $n/2^i$ distinct values.
9. Oops: No such set of real numbers.
10. NO SUCH SET.

Computing a polynomial quickly on n values

Lets just use some magic.

1. Assume: polynomials have degree $n-1$, where $n = 2^k$.
2. .. pad polynomials with terms having zero coefficients.
3. ***Magic set*** of numbers: $\Psi = \{x_1, \ldots, x_n\}$.
   Property: $|\mathbf{SQ}(\Psi)| = n/2$, where $\mathbf{SQ}(\Psi) = \left\{ x^2 \;\middle|\; x \in \Psi \right\}$.
4. $|\mathrm{square}(\Psi)| = |\Psi|/2$.
5. Easy to find such set...
6. Magic: Have this property repeatedly...
   $\mathbf{SQ}(\mathbf{SQ}(\Psi))$ has $n/4$ distinct values.
7. $\mathbf{SQ}(\mathbf{SQ}(\mathbf{SQ}(\Psi)))$ has $n/8$ values.
8. $\mathbf{SQ}^i(\Psi)$ has $n/2^i$ distinct values.
9. Oops: No such set of real numbers.
10. NO SUCH SET.

Computing a polynomial quickly on n values

Lets just use some magic.

1. Assume: polynomials have degree $n-1$, where $n = 2^k$.
2. .. pad polynomials with terms having zero coefficients.
3. ***Magic set*** of numbers: $\Psi = \{x_1, \ldots, x_n\}$.
   Property: $|\mathbf{SQ}(\Psi)| = n/2$, where $\mathbf{SQ}(\Psi) = \left\{ x^2 \;\middle|\; x \in \Psi \right\}$.
4. $|\mathrm{square}(\Psi)| = |\Psi|/2$.
5. Easy to find such set...
6. Magic: Have this property repeatedly...
   $\mathbf{SQ}(\mathbf{SQ}(\Psi))$ has $n/4$ distinct values.
7. $\mathbf{SQ}(\mathbf{SQ}(\mathbf{SQ}(\Psi)))$ has $n/8$ values.
8. $\mathbf{SQ}^i(\Psi)$ has $n/2^i$ distinct values.
9. Oops: No such set of real numbers.
10. NO SUCH SET.

Collapsible sets

Assume magic...

Let us for the time being ignore this technicality, and fly, for a moment, into the land of fantasy, and assume that we do have such a set of numbers, so that $|\mathbf{SQ}^i(\Psi)| = n/2^i$ numbers, for $i = 0, \ldots, k$. Let us call such a set of numbers ***collapsible***.

Breaking the input polynomial into...

... two polynomials of half the degree

1. For a set $\mathcal{X} = \{x_0, \ldots, x_n\}$ and polynomial $p(x)$, let

$$p\Big(\mathcal{X}\Big) = \left\langle \Big(x_0, p(x_0)\Big), \ldots, \Big(x_n, p(x_n)\Big) \right\rangle.$$

2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$, where

$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$

3. all even degree terms in $u(\cdot)$, all odd degree terms in $v(\cdot)$.
4. maximum degree of $u(y)$, $v(y)$ is $n/2$.

Breaking the input polynomial into...

... two polynomials of half the degree

1. For a set $\mathcal{X} = \{x_0, \ldots, x_n\}$ and polynomial $p(x)$, let
$$p\left(\mathcal{X}\right) = \left\langle \left(x_0, p(x_0)\right), \ldots, \left(x_n, p(x_n)\right) \right\rangle.$$
2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$, where
$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$
3. all even degree terms in $u(\cdot)$, all odd degree terms in $v(\cdot)$.
4. maximum degree of $u(y)$, $v(y)$ is $n/2$.

Breaking the input polynomial into...

... two polynomials of half the degree

1. For a set $\mathcal{X} = \{x_0, \ldots, x_n\}$ and polynomial $p(x)$, let
$$p\Big(\mathcal{X}\Big) = \left\langle \Big(x_0, p(x_0)\Big), \ldots, \Big(x_n, p(x_n)\Big) \right\rangle.$$
2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$, where
$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$
3. all even degree terms in $u(\cdot)$, all odd degree terms in $v(\cdot)$.
4. maximum degree of $u(y)$, $v(y)$ is $n/2$.

Breaking the input polynomial into...

... two polynomials of half the degree

1. For a set $\mathcal{X} = \{x_0, \ldots, x_n\}$ and polynomial $p(x)$, let
$$p\Big(\mathcal{X}\Big) = \left\langle \Big(x_0, p(x_0)\Big), \ldots, \Big(x_n, p(x_n)\Big) \right\rangle.$$
2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$, where
$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$
3. all even degree terms in $u(\cdot)$, all odd degree terms in $v(\cdot)$.
4. maximum degree of $u(y)$, $v(y)$ is $n/2$.

FFT: The dividing stage

1. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
2. Ψ: collapsible set of size n.
3. $p(\Psi)$: compute polynomial of degree $n-1$ on n values.
4. Decompose:

$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$

5. Need to compute $u(x^2)$, for all $x \in \Psi$.
6. Need to compute $v(x^2)$, for all $x \in \Psi$.
7. $\mathbf{SQ}(\Psi) = \left\{ x^2 \;\middle|\; x \in \Psi \right\}$.
8. $\implies$ Need to compute $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$.
9. $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$: comp. poly. degree $\frac{n}{2} - 1$ on $\frac{n}{2}$ values.

FFT: The dividing stage

1. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
2. Ψ: collapsible set of size n.
3. $p(\Psi)$: compute polynomial of degree $n-1$ on n values.
4. Decompose:

$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$

5. Need to compute $u(x^2)$, for all $x \in \Psi$.
6. Need to compute $v(x^2)$, for all $x \in \Psi$.
7. $\mathbf{SQ}(\Psi) = \left\{ x^2 \,\middle|\, x \in \Psi \right\}$.
8. $\implies$ Need to compute $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$.
9. $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$: comp. poly. degree $\frac{n}{2} - 1$ on $\frac{n}{2}$ values.

FFT: The dividing stage

1. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
2. Ψ: collapsible set of size n.
3. $p(\Psi)$: compute polynomial of degree $n-1$ on n values.
4. Decompose:

$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$

5. Need to compute $u(x^2)$, for all $x \in \Psi$.
6. Need to compute $v(x^2)$, for all $x \in \Psi$.
7. $\mathbf{SQ}(\Psi) = \left\{ x^2 \;\middle|\; x \in \Psi \right\}$.
8. $\implies$ Need to compute $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$.
9. $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$: comp. poly. degree $\frac{n}{2} - 1$ on $\frac{n}{2}$ values.

FFT: The dividing stage

1. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
2. Ψ: collapsible set of size n.
3. $p(\Psi)$: compute polynomial of degree $n-1$ on n values.
4. Decompose:

$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$

5. Need to compute $u(x^2)$, for all $x \in \Psi$.
6. Need to compute $v(x^2)$, for all $x \in \Psi$.
7. $\mathbf{SQ}(\Psi) = \left\{ x^2 \,\middle|\, x \in \Psi \right\}$.
8. $\implies$ Need to compute $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$.
9. $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$: comp. poly. degree $\frac{n}{2} - 1$ on $\frac{n}{2}$ values.

FFT: The dividing stage

1. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
2. Ψ: collapsible set of size n.
3. $p(\Psi)$: compute polynomial of degree $n-1$ on n values.
4. Decompose:

$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$

5. Need to compute $u(x^2)$, for all $x \in \Psi$.
6. Need to compute $v(x^2)$, for all $x \in \Psi$.
7. $\mathbf{SQ}(\Psi) = \left\{ x^2 \,\middle|\, x \in \Psi \right\}$.
8. $\implies$ Need to compute $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$.
9. $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$: comp. poly. degree $\frac{n}{2} - 1$ on $\frac{n}{2}$ values.

FFT: The dividing stage

1. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
2. Ψ: collapsible set of size n.
3. $p(\Psi)$: compute polynomial of degree $n-1$ on n values.
4. Decompose:

$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$

5. Need to compute $u(x^2)$, for all $x \in \Psi$.
6. Need to compute $v(x^2)$, for all $x \in \Psi$.
7. $\mathbf{SQ}(\Psi) = \left\{ x^2 \;\middle|\; x \in \Psi \right\}$.
8. $\implies$ Need to compute $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$.
9. $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$: comp. poly. degree $\frac{n}{2} - 1$ on $\frac{n}{2}$ values.

FFT: The dividing stage

1. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
2. Ψ: collapsible set of size n.
3. $p(\Psi)$: compute polynomial of degree $n-1$ on n values.
4. Decompose:

$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$

5. Need to compute $u(x^2)$, for all $x \in \Psi$.
6. Need to compute $v(x^2)$, for all $x \in \Psi$.
7. $\mathbf{SQ}(\Psi) = \left\{ x^2 \;\middle|\; x \in \Psi \right\}$.
8. $\implies$ Need to compute $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$.
9. $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$: comp. poly. degree $\frac{n}{2} - 1$ on $\frac{n}{2}$ values.

FFT: The dividing stage

1. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
2. Ψ: collapsible set of size n.
3. $p(\Psi)$: compute polynomial of degree $n-1$ on n values.
4. Decompose:

$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$

5. Need to compute $u(x^2)$, for all $x \in \Psi$.
6. Need to compute $v(x^2)$, for all $x \in \Psi$.
7. $\mathbf{SQ}(\Psi) = \left\{ x^2 \,\middle|\, x \in \Psi \right\}$.
8. $\implies$ Need to compute $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$.
9. $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$: comp. poly. degree $\frac{n}{2} - 1$ on $\frac{n}{2}$ values.

FFT: The dividing stage

1. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
2. Ψ: collapsible set of size n.
3. $p(\Psi)$: compute polynomial of degree $n - 1$ on n values.
4. Decompose:

$$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i \quad \text{and} \quad v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i.$$

5. Need to compute $u(x^2)$, for all $x \in \Psi$.
6. Need to compute $v(x^2)$, for all $x \in \Psi$.
7. $\mathsf{SQ}(\Psi) = \left\{ x^2 \,\middle|\, x \in \Psi \right\}$.
8. $\implies$ Need to compute $u(\mathsf{SQ}(\Psi)), v(\mathsf{SQ}(\Psi))$.
9. $u(\mathsf{SQ}(\Psi)), v(\mathsf{SQ}(\Psi))$: comp. poly. degree $\frac{n}{2} - 1$ on $\frac{n}{2}$ values.

FFT: The conquering stage

1. Ψ: Collapsible set of size n.
2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
3. $u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i$ and $v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i$.
4. $u(\mathsf{SQ}(\Psi)), v(\mathsf{SQ}(\Psi))$: Computed recursively.
5. Need to compute $p(\Psi)$.
6. For $x \in \Psi$: Compute $p(x) = u(x^2) + x \cdot v(x^2)$.
7. Takes constant time per single element $x \in \Psi$.
8. Takes $O(n)$ time overall.

FFT: The conquering stage

1. Ψ: Collapsible set of size n.
2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
3. $u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i$ and $v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i$.
4. $u(\mathsf{SQ}(\Psi)), v(\mathsf{SQ}(\Psi))$: Computed recursively.
5. Need to compute $p(\Psi)$.
6. For $x \in \Psi$: Compute $p(x) = u(x^2) + x \cdot v(x^2)$.
7. Takes constant time per single element $x \in \Psi$.
8. Takes $O(n)$ time overall.

FFT: The conquering stage

1. Ψ: Collapsible set of size n.
2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
3. $u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i$ and $v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i$.
4. $u(\mathsf{SQ}(\Psi)), v(\mathsf{SQ}(\Psi))$: Computed recursively.
5. Need to compute $p(\Psi)$.
6. For $x \in \Psi$: Compute $p(x) = u(x^2) + x \cdot v(x^2)$.
7. Takes constant time per single element $x \in \Psi$.
8. Takes $O(n)$ time overall.

FFT: The conquering stage

1. Ψ: Collapsible set of size n.
2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
3. $u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i$ and $v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i$.
4. $u(\mathsf{SQ}(\Psi)), v(\mathsf{SQ}(\Psi))$: Computed recursively.
5. Need to compute $p(\Psi)$.
6. For $x \in \Psi$: Compute $p(x) = u(x^2) + x \cdot v(x^2)$.
7. Takes constant time per single element $x \in \Psi$.
8. Takes $O(n)$ time overall.

FFT: The conquering stage

1. Ψ: Collapsible set of size n.
2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
3. $u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i$ and $v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i$.
4. $u(\mathsf{SQ}(\Psi)), v(\mathsf{SQ}(\Psi))$: Computed recursively.
5. Need to compute $p(\Psi)$.
6. For $x \in \Psi$: Compute $p(x) = u(x^2) + x \cdot v(x^2)$.
7. Takes constant time per single element $x \in \Psi$.
8. Takes $O(n)$ time overall.

FFT: The conquering stage

1. Ψ: Collapsible set of size n.
2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
3. $u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i$ and $v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i$.
4. $u(\mathbf{SQ}(\Psi)), v(\mathbf{SQ}(\Psi))$: Computed recursively.
5. Need to compute $p(\Psi)$.
6. For $x \in \Psi$: Compute $p(x) = u(x^2) + x \cdot v(x^2)$.
7. Takes constant time per single element $x \in \Psi$.
8. Takes $O(n)$ time overall.

FFT: The conquering stage

1. Ψ: Collapsible set of size n.
2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
3. $u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i$ and $v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i$.
4. $u(\mathsf{SQ}(\Psi)), v(\mathsf{SQ}(\Psi))$: Computed recursively.
5. Need to compute $p(\Psi)$.
6. For $x \in \Psi$: Compute $p(x) = u(x^2) + x \cdot v(x^2)$.
7. Takes constant time per single element $x \in \Psi$.
8. Takes $O(n)$ time overall.

FFT: The conquering stage

1. Ψ: Collapsible set of size n.
2. $p(x) = \sum_{i=0}^{n-1} a_i x^i$ as $p(x) = u(x^2) + x \cdot v(x^2)$.
3. $u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i$ and $v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i$.
4. $u(\mathsf{SQ}(\Psi)), v(\mathsf{SQ}(\Psi))$: Computed recursively.
5. Need to compute $p(\Psi)$.
6. For $x \in \Psi$: Compute $p(x) = u(x^2) + x \cdot v(x^2)$.
7. Takes constant time per single element $x \in \Psi$.
8. Takes $O(n)$ time overall.

FFT algorithm

FFTAlg(p, X)

input: $p(x)$: A polynomial of degree n: $p(x) = \sum_{i=0}^{n-1} a_i x^i$

X: A collapsible set of n elements.

output: $p(X)$

$u(y) = \sum_{i=0}^{n/2-1} a_{2i} y^i$

$v(y) = \sum_{i=0}^{n/2-1} a_{1+2i} y^i$.

$Y = \mathbf{SQ}(X) = \left\{ x^2 \;\middle|\; x \in X \right\}$.

$U = \textbf{FFTAlg}(u, Y)$ /* $U = u(Y)$ */

$V = \textbf{FFTAlg}(v, Y)$ /* $V = v(Y)$ */

$Out \leftarrow \emptyset$

for $x \in X$ **do**

/* $p(x) = u(x^2) + x * v(x^2)$, $U[x^2]$ is the value $u(x^2)$ */

$(x, p(x)) \leftarrow \left(x, U[x^2] + x \cdot V[x^2]\right)$

$Out \leftarrow Out \cup \{(x, p(x))\}$

return Out

Running time analysis...

...an old foe emerges once again to serve

1. $T(m, n)$: Time of computing a polynomial of degree m on n values.
2. We have that:

$$T(n-1, n) = 2T(n/2 - 1, n/2) + O(n).$$

3. The solution to this recurrence is $O(n \log n)$.

Running time analysis...

...an old foe emerges once again to serve

1. $T(m, n)$: Time of computing a polynomial of degree m on n values.
2. We have that:

$$T(n-1, n) = 2T(n/2-1, n/2) + O(n).$$

3. The solution to this recurrence is $O(n \log n)$.

Running time analysis...

...an old foe emerges once again to serve

1. $T(m, n)$: Time of computing a polynomial of degree m on n values.
2. We have that:

$$T(n-1, n) = 2T(n/2 - 1, n/2) + O(n).$$

3. The solution to this recurrence is $O(n \log n)$.

Generating Collapsible Sets

1. How to generate collapsible sets?
2.

Generating Collapsible Sets

1. How to generate collapsible sets?
2. Trick: Use complex numbers!

Complex numbers – a quick reminder

1. Complex number: pair (α, β) of real numbers. Written as $\tau = \alpha + \mathrm{i}\beta$.
2. α: ***real*** part, β: ***imaginary*** part.
3. i is the root of -1.
4. Geometrically: a point in the complex plane:
    1. ***polar form***: $\tau = r\cos\phi + \mathrm{i}r\sin\phi = r(\cos\phi + \mathrm{i}\sin\phi)$
    2. $r = \sqrt{\alpha^2 + \beta^2}$ and $\phi = \arcsin(\beta/\alpha)$.

Complex numbers – a quick reminder

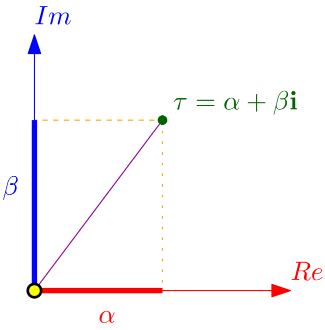
1. Complex number: pair (α, β) of real numbers. Written as $\tau = \alpha + \mathrm{i}\beta$.
2. α: ***real*** part, β: ***imaginary*** part.
3. i is the root of -1.
4. Geometrically: a point in the complex plane:
    1. ***polar form***: $\tau = r\cos\phi + \mathrm{i}r\sin\phi = r(\cos\phi + \mathrm{i}\sin\phi)$
    2. $r = \sqrt{\alpha^2 + \beta^2}$ and $\phi = \arcsin(\beta/\alpha)$.

Complex numbers – a quick reminder

1. Complex number: pair (α, β) of real numbers. Written as $\tau = \alpha + \mathrm{i}\beta$.
2. α: ***real*** part, β: ***imaginary*** part.
3. i is the root of -1.
4. Geometrically: a point in the complex plane:
   1. ***polar form***: $\tau = r\cos\phi + \mathrm{i}r\sin\phi = r(\cos\phi + \mathrm{i}\sin\phi)$
   2. $r = \sqrt{\alpha^2 + \beta^2}$ and $\phi = \arcsin(\beta/\alpha)$.

Complex numbers – a quick reminder

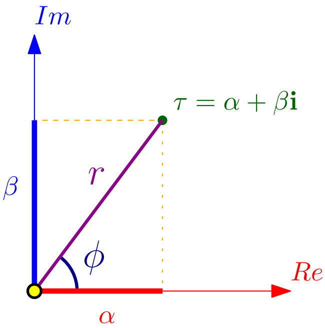
1. Complex number: pair (α, β) of real numbers. Written as $\tau = \alpha + \mathrm{i}\beta$.
2. α: ***real*** part,
   β: ***imaginary*** part.
3. i is the root of -1.
4. Geometrically: a point in the complex plane:



1. ***polar form***: $\tau = r\cos\phi + \mathrm{i}r\sin\phi = r(\cos\phi + \mathrm{i}\sin\phi)$
2. $r = \sqrt{\alpha^2 + \beta^2}$ and $\phi = \arcsin(\beta/\alpha)$.

Complex numbers – a quick reminder

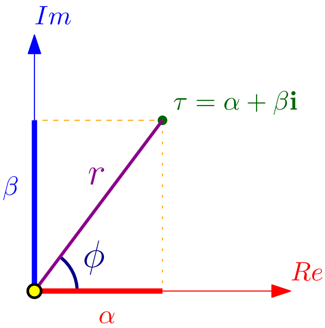
1. Complex number: pair (α, β) of real numbers. Written as $\tau = \alpha + \mathrm{i}\beta$.
2. α: ***real*** part, β: ***imaginary*** part.
3. i is the root of -1.
4. Geometrically: a point in the complex plane:



1. ***polar form***: $\tau = r\cos\phi + \mathrm{i}r\sin\phi = r(\cos\phi + \mathrm{i}\sin\phi)$
2. $r = \sqrt{\alpha^2 + \beta^2}$ and $\phi = \arcsin(\beta/\alpha)$.

Complex numbers – a quick reminder

1. Complex number: pair (α, β) of real numbers. Written as $\tau = \alpha + \mathbf{i}\beta$.
2. α: ***real*** part, β: ***imaginary*** part.
3. $\mathbf{i}$ is the root of -1.
4. Geometrically: a point in the complex plane:



1. ***polar form***: $\tau = r\cos\phi + \mathbf{i}r\sin\phi = r(\cos\phi + \mathbf{i}\sin\phi)$
2. $r = \sqrt{\alpha^2 + \beta^2}$ and $\phi = \arcsin(\beta/\alpha)$.

A useful formula: $\cos\phi + \mathrm{i}\sin\phi = \mathrm{e}^{\mathrm{i}\phi}$

1. By Taylor's expansion:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots,$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots,$$

and

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots.$$

2. Since $\mathrm{i}^2 = -1$:

$$e^{\mathrm{i}x} = 1 + \mathrm{i}\frac{x}{1!} - \frac{x^2}{2!} - \mathrm{i}\frac{x^3}{3!} + \frac{x^4}{4!} + \mathrm{i}\frac{x^5}{5!} - \frac{x^6}{6!} \cdots$$
$$= \cos x + \mathrm{i}\sin x.$$

A useful formula: $\cos\phi + \mathrm{i}\sin\phi = e^{\mathrm{i}\phi}$

1. By Taylor's expansion:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots,$$
$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots,$$

and $$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots.$$

2. Since $\mathrm{i}^2 = -1$:

$$e^{\mathrm{i}x} = 1 + \mathrm{i}\frac{x}{1!} - \frac{x^2}{2!} - \mathrm{i}\frac{x^3}{3!} + \frac{x^4}{4!} + \mathrm{i}\frac{x^5}{5!} - \frac{x^6}{6!}\cdots$$
$$= \cos x + \mathrm{i}\sin x.$$

Back to polar form

1. ***polar form***:
   $\tau = r\cos\phi + \mathrm{i} r\sin\phi = r(\cos\phi + \mathrm{i}\sin\phi) = re^{i\phi}$,
2. $\tau = re^{\mathrm{i}\phi}$, $\quad \tau' = r'e^{\mathrm{i}\phi'}$: complex numbers.
3. $\tau \cdot \tau' = re^{\mathrm{i}\phi} \cdot r'e^{\mathrm{i}\phi'} = rr'e^{\mathrm{i}(\phi+\phi')}$.
4. $e^{\mathrm{i}\phi}$ is 2π periodic (i.e., $e^{\mathrm{i}\phi} = e^{\mathrm{i}(\phi+2\pi)}$), and $1 = e^{\mathrm{i}0}$.
5. nth root of 1: complex number τ – raise it to power n get 1.
6. $\tau = re^{\mathrm{i}\phi}$, such that $\tau^n = r^n e^{\mathrm{i}n\phi} = e^{\mathrm{i}0}$.
7. $\implies r = 1$, and there must be an integer j, such that

$$n\phi = 0 + 2\pi j \implies \phi = j(2\pi/n).$$

Back to polar form

1. ***polar form***:
   $\tau = r\cos\phi + \mathrm{i}r\sin\phi = r(\cos\phi + \mathrm{i}\sin\phi) = re^{i\phi}$,
2. $\tau = re^{\mathrm{i}\phi}$, $\quad \tau' = r'e^{\mathrm{i}\phi'}$: complex numbers.
3. $\tau \cdot \tau' = re^{\mathrm{i}\phi} \cdot r'e^{\mathrm{i}\phi'} = rr'e^{\mathrm{i}(\phi+\phi')}$.
4. $e^{\mathrm{i}\phi}$ is 2π periodic (i.e., $e^{\mathrm{i}\phi} = e^{\mathrm{i}(\phi+2\pi)}$), and $1 = e^{\mathrm{i}0}$.
5. nth root of 1: complex number τ – raise it to power n get 1.
6. $\tau = re^{\mathrm{i}\phi}$, such that $\tau^n = r^n e^{\mathrm{i}n\phi} = e^{\mathrm{i}0}$.
7. $\implies r = 1$, and there must be an integer j, such that

$$n\phi = 0 + 2\pi j \implies \phi = j(2\pi/n).$$

Back to polar form

1. ***polar form***:
   $\tau = r\cos\phi + \mathrm{i} r \sin\phi = r(\cos\phi + \mathrm{i}\sin\phi) = re^{i\phi}$,
2. $\tau = re^{\mathrm{i}\phi}$, $\tau' = r' e^{\mathrm{i}\phi'}$: complex numbers.
3. $\tau \cdot \tau' = re^{\mathrm{i}\phi} \cdot r' e^{\mathrm{i}\phi'} = rr' e^{\mathrm{i}(\phi + \phi')}$.
4. $e^{\mathrm{i}\phi}$ is 2π periodic (i.e., $e^{\mathrm{i}\phi} = e^{\mathrm{i}(\phi + 2\pi)}$), and $1 = e^{\mathrm{i}0}$.
5. nth root of 1: complex number τ – raise it to power n get 1.
6. $\tau = re^{\mathrm{i}\phi}$, such that $\tau^n = r^n e^{\mathrm{i}n\phi} = e^{\mathrm{i}0}$.
7. $\implies$ $r = 1$, and there must be an integer j, such that

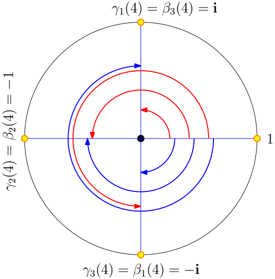
$$n\phi = 0 + 2\pi j \implies \phi = j(2\pi/n).$$

Back to polar form

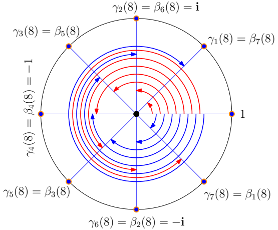
1. ***polar form***:
   $\tau = r\cos\phi + \mathrm{i}r\sin\phi = r(\cos\phi + \mathrm{i}\sin\phi) = re^{i\phi}$,
2. $\tau = re^{\mathrm{i}\phi}$, $\quad \tau' = r'e^{\mathrm{i}\phi'}$: complex numbers.
3. $\tau \cdot \tau' = re^{\mathrm{i}\phi} \cdot r'e^{\mathrm{i}\phi'} = rr'e^{\mathrm{i}(\phi+\phi')}$.
4. $e^{\mathrm{i}\phi}$ is 2π periodic (i.e., $e^{\mathrm{i}\phi} = e^{\mathrm{i}(\phi+2\pi)}$), and $1 = e^{\mathrm{i}0}$.
5. nth root of 1: complex number τ – raise it to power n get 1.
6. $\tau = re^{\mathrm{i}\phi}$, such that $\tau^n = r^n e^{\mathrm{i}n\phi} = e^{\mathrm{i}0}$.
7. $\implies r = 1$, and there must be an integer j, such that

$$n\phi = 0 + 2\pi j \implies \phi = j(2\pi/n).$$

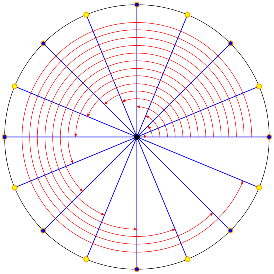
For $j = 0, \ldots, n-1$, we get the n distinct ***roots of unity***.



$(n = 4)$



$(n = 8)$



$(n = 16)$

Back to collapsible sets

1. Can do all basic calculations on complex numbers in $O(1)$ time.
2. Idea: Work over the complex numbers.
3. Use roots of unity!
4. γ: nth root of unity. There are n such roots, and let $\gamma_j(n)$ denote the jth root.

$$\gamma_j(n) = \cos((2\pi j)/n) + \mathrm{i}\sin((2\pi j)/n) = \gamma^j.$$

Let $\mathcal{A}(n) = \{\gamma_0(n), \ldots, \gamma_{n-1}(n)\}$.

5. $|\mathbf{SQ}(\mathcal{A}(n))|$ has $n/2$ entries.
6. $\mathbf{SQ}(\mathcal{A}(n)) = \mathcal{A}(n/2)$
7. n to be a power of 2, then $\mathcal{A}(n)$ is the *required* collapsible set.

Back to collapsible sets

1. Can do all basic calculations on complex numbers in $O(1)$ time.
2. Idea: Work over the complex numbers.
3. Use roots of unity!
4. γ: nth root of unity. There are n such roots, and let $\gamma_j(n)$ denote the jth root.

$$\gamma_j(n) = \cos((2\pi j)/n) + \mathrm{i}\sin((2\pi j)/n) = \gamma^j.$$

   Let $\mathcal{A}(n) = \{\gamma_0(n), \ldots, \gamma_{n-1}(n)\}$.
5. $|\mathbf{SQ}(\mathcal{A}(n))|$ has $n/2$ entries.
6. $\mathbf{SQ}(\mathcal{A}(n)) = \mathcal{A}(n/2)$
7. n to be a power of 2, then $\mathcal{A}(n)$ is the *required* collapsible set.

Back to collapsible sets

1. Can do all basic calculations on complex numbers in $O(1)$ time.
2. Idea: Work over the complex numbers.
3. Use roots of unity!
4. γ: nth root of unity. There are n such roots, and let $\gamma_j(n)$ denote the jth root.

$$\gamma_j(n) = \cos((2\pi j)/n) + \mathrm{i}\sin((2\pi j)/n) = \gamma^j.$$

Let $\mathcal{A}(n) = \{\gamma_0(n), \ldots, \gamma_{n-1}(n)\}$.

5. $|\mathbf{SQ}(\mathcal{A}(n))|$ has $n/2$ entries.
6. $\mathbf{SQ}(\mathcal{A}(n)) = \mathcal{A}(n/2)$
7. n to be a power of 2, then $\mathcal{A}(n)$ is the *required* collapsible set.

Back to collapsible sets

1. Can do all basic calculations on complex numbers in $O(1)$ time.
2. Idea: Work over the complex numbers.
3. Use roots of unity!
4. γ: nth root of unity. There are n such roots, and let $\gamma_j(n)$ denote the jth root.

$$\gamma_j(n) = \cos((2\pi j)/n) + \mathrm{i}\sin((2\pi j)/n) = \gamma^j.$$

Let $\mathcal{A}(n) = \{\gamma_0(n), \ldots, \gamma_{n-1}(n)\}$.

5. $|\mathsf{SQ}(\mathcal{A}(n))|$ has $n/2$ entries.
6. $\mathsf{SQ}(\mathcal{A}(n)) = \mathcal{A}(n/2)$
7. n to be a power of 2, then $\mathcal{A}(n)$ is the *required* collapsible set.

Back to collapsible sets

1. Can do all basic calculations on complex numbers in $O(1)$ time.
2. Idea: Work over the complex numbers.
3. Use roots of unity!
4. γ: nth root of unity. There are n such roots, and let $\gamma_j(n)$ denote the jth root.

$$\gamma_j(n) = \cos((2\pi j)/n) + \mathrm{i}\sin((2\pi j)/n) = \gamma^j.$$

Let $\mathcal{A}(n) = \{\gamma_0(n), \ldots, \gamma_{n-1}(n)\}$.

5. $|\mathbf{SQ}(\mathcal{A}(n))|$ has $n/2$ entries.
6. $\mathbf{SQ}(\mathcal{A}(n)) = \mathcal{A}(n/2)$
7. n to be a power of 2, then $\mathcal{A}(n)$ is the *required* collapsible set.

Back to collapsible sets

1. Can do all basic calculations on complex numbers in $O(1)$ time.
2. Idea: Work over the complex numbers.
3. Use roots of unity!
4. γ: nth root of unity. There are n such roots, and let $\gamma_j(n)$ denote the jth root.

$$\gamma_j(n) = \cos((2\pi j)/n) + \mathrm{i}\sin((2\pi j)/n) = \gamma^j.$$

   Let $\mathcal{A}(n) = \{\gamma_0(n), \ldots, \gamma_{n-1}(n)\}$.
5. $|\mathbf{SQ}(\mathcal{A}(n))|$ has $n/2$ entries.
6. $\mathbf{SQ}(\mathcal{A}(n)) = \mathcal{A}(n/2)$
7. n to be a power of 2, then $\mathcal{A}(n)$ is the *required* collapsible set.

Back to collapsible sets

1. Can do all basic calculations on complex numbers in $O(1)$ time.
2. Idea: Work over the complex numbers.
3. Use roots of unity!
4. γ: nth root of unity. There are n such roots, and let $\gamma_j(n)$ denote the jth root.

$$\gamma_j(n) = \cos((2\pi j)/n) + \mathrm{i}\sin((2\pi j)/n) = \gamma^j.$$

   Let $\mathcal{A}(n) = \{\gamma_0(n), \ldots, \gamma_{n-1}(n)\}$.
5. $|\mathbf{SQ}(\mathcal{A}(n))|$ has $n/2$ entries.
6. $\mathbf{SQ}(\mathcal{A}(n)) = \mathcal{A}(n/2)$
7. n to be a power of 2, then $\mathcal{A}(n)$ is the *required* collapsible set.

The first result...

Theorem

Given polynomial $p(x)$ of degree n, where n is a power of two, then we can compute $p(X)$ in $O(n \log n)$ time, where $X = \mathcal{A}(n)$ is the set of n different powers of the nth root of unity over the complex numbers.

Problem...

We can go, but can we come back?

1. Can multiply two polynomials quickly
2. by transforming them to the point-value pairs representation...
3. over the nth roots of unity.
4. Q: How to transform this representation back to the regular representation.
5. A: Do some confusing math...

Problem...

We can go, but can we come back?

1. Can multiply two polynomials quickly
2. by transforming them to the point-value pairs representation...
3. over the nth roots of unity.
4. Q: How to transform this representation back to the regular representation.
5. A: Do some confusing math...

Problem...

We can go, but can we come back?

1. Can multiply two polynomials quickly
2. by transforming them to the point-value pairs representation...
3. over the nth roots of unity.
4. Q: How to transform this representation back to the regular representation.
5. A: Do some confusing math...

Problem...

We can go, but can we come back?

1. Can multiply two polynomials quickly
2. by transforming them to the point-value pairs representation...
3. over the nth roots of unity.
4. Q: How to transform this representation back to the regular representation.
5. A: Do some confusing math...

Problem...

We can go, but can we come back?

1. Can multiply two polynomials quickly
2. by transforming them to the point-value pairs representation...
3. over the nth roots of unity.
4. Q: How to transform this representation back to the regular representation.
5. A: Do some confusing math...

Recovering the polynomial

Think about FFT as a matrix multiplication operator.
$p(x) = \sum_{i=0}^{n-1} a_i x^i$. Evaluating $p(\cdot)$ on $\mathcal{A}(n)$:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & \gamma_0 & \gamma_0^2 & \gamma_0^3 & \cdots & \gamma_0^{n-1} \\ 1 & \gamma_1 & \gamma_1^2 & \gamma_1^3 & \cdots & \gamma_1^{n-1} \\ 1 & \gamma_2 & \gamma_2^2 & \gamma_2^3 & \cdots & \gamma_2^{n-1} \\ 1 & \gamma_3 & \gamma_3^2 & \gamma_3^3 & \cdots & \gamma_3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \gamma_{n-1} & \gamma_{n-1}^2 & \gamma_{n-1}^3 & \cdots & \gamma_{n-1}^{n-1} \end{pmatrix}}_{\text{the matrix } V} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix},$$

where $\gamma_j = \gamma_j(n) = (\gamma_1(n))^j$ is the jth power of the nth root of unity, and $y_j = p(\gamma_j)$.

The Vandermonde matrix

Because every matrix needs a name

V is the ***Vandermonde*** matrix.
V^{-1}: inverse matrix of V
Vandermonde matrix. And let multiply the above formula from the left. We get:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix} = V \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix} \qquad \Longrightarrow \qquad \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix} = V^{-1} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{pmatrix}.$$

The inverse Vandermonde matrix

..for the rescue

1. Recover the polynomial $p(x)$ from the point-value pairs

$$\left\{(\gamma_0, p(\gamma_0)), (\gamma_1, p(\gamma_1)), \ldots, (\gamma_{n-1}, p(\gamma_{n-1}))\right\}$$

2. by doing a single matrix multiplication of V^{-1} by the vector $[y_0, y_1, \ldots, y_{n-1}]$.
3. Multiplying a vector with n entries with $n \times n$ matrix takes $O(n^2)$ time.
4. No benefit so far...

The inverse Vandermonde matrix

..for the rescue

1. Recover the polynomial $p(x)$ from the point-value pairs

$$\left\{(\gamma_0, p(\gamma_0)), (\gamma_1, p(\gamma_1)), \ldots, (\gamma_{n-1}, p(\gamma_{n-1}))\right\}$$

2. by doing a single matrix multiplication of V^{-1} by the vector $[y_0, y_1, \ldots, y_{n-1}]$.
3. Multiplying a vector with n entries with $n \times n$ matrix takes $O(n^2)$ time.
4. No benefit so far...

The inverse Vandermonde matrix

..for the rescue

1. Recover the polynomial $p(x)$ from the point-value pairs

$$\Big\{(\gamma_0, p(\gamma_0)), (\gamma_1, p(\gamma_1)), \ldots, (\gamma_{n-1}, p(\gamma_{n-1}))\Big\}$$

2. by doing a single matrix multiplication of V^{-1} by the vector $[y_0, y_1, \ldots, y_{n-1}]$.
3. Multiplying a vector with n entries with $n \times n$ matrix takes $O(n^2)$ time.
4. No benefit so far...

The inverse Vandermonde matrix

..for the rescue

1. Recover the polynomial $p(x)$ from the point-value pairs

$$\left\{(\gamma_0, p(\gamma_0)), (\gamma_1, p(\gamma_1)), \ldots, (\gamma_{n-1}, p(\gamma_{n-1}))\right\}$$

2. by doing a single matrix multiplication of V^{-1} by the vector $[y_0, y_1, \ldots, y_{n-1}]$.
3. Multiplying a vector with n entries with $n \times n$ matrix takes $O(n^2)$ time.
4. No benefit so far...

What is the inverse of the Vandermonde matrix

Vandermonde matrix is famous, beautiful and well known – a celebrity matrix

Claim

$$V^{-1} = \frac{1}{n}\begin{pmatrix} 1 & \beta_0 & \beta_0^2 & \beta_0^3 & \cdots & \beta_0^{n-1} \\ 1 & \beta_1 & \beta_1^2 & \beta_1^3 & \cdots & \beta_1^{n-1} \\ 1 & \beta_2 & \beta_2^2 & \beta_2^3 & \cdots & \beta_2^{n-1} \\ 1 & \beta_3 & \beta_3^2 & \beta_3^3 & \cdots & \beta_3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \beta_{n-1} & \beta_{n-1}^2 & \beta_{n-1}^3 & \cdots & \beta_{n-1}^{n-1} \end{pmatrix},$$

where $\beta_j = (\gamma_j(n))^{-1}$.

Proof

Consider the (u, v) entry in the matrix $C = V^{-1}V$. We have

$$C_{u,v} = \sum_{j=0}^{n-1} \frac{(\beta_u)^j (\gamma_j)^v}{n}.$$

As $\gamma_j = (\gamma_1)^j$.Thus,

$$C_{u,v} = \sum_{j=0}^{n-1} \frac{(\beta_u)^j ((\gamma_1)^j)^v}{n} = \sum_{j=0}^{n-1} \frac{(\beta_u)^j ((\gamma_1)^v)^j}{n} = \sum_{j=0}^{n-1} \frac{(\beta_u \gamma_v)^j}{n}.$$

Clearly, if $u = v$ then

$$C_{u,u} = \frac{1}{n} \sum_{j=0}^{n-1} (\beta_u \gamma_u)^j = \frac{1}{n} \sum_{j=0}^{n-1} (1)^j = \frac{n}{n} = 1.$$

Proof continued...

If $u \neq v$ then,

$$\beta_u \gamma_v = (\gamma_u)^{-1} \gamma_v = (\gamma_1)^{-u} \gamma_1^v = (\gamma_1)^{v-u} = \gamma_{v-u}.$$

And

$$C_{u,v} = \frac{1}{n} \sum_{j=0}^{n-1} (\gamma_{v-u})^j = \frac{1}{n} \cdot \frac{\gamma_{v-u}^n - 1}{\gamma_{v-u} - 1} = \frac{1}{n} \cdot \frac{1 - 1}{\gamma_{v-u} - 1} = 0,$$

Proved that the matrix C have ones on the diagonal and zero everywhere else. ■

Recap...

1. n point-value pairs $\{(\gamma_0, y_0), \ldots, (\gamma_{n-1}, y_{n-1})\}$: of polynomial $p(x) = \sum_{i=0}^{n-1} a_i x^i$ over nth roots of unity.
2. Recover coefficients of polynomial by multiplying $[y_0, y_1, \ldots, y_n]$ by V^{-1}:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} = \frac{1}{n} \underbrace{\begin{pmatrix} 1 & \beta_0 & \beta_0^2 & \beta_0^3 & \cdots & \beta_0^{n-1} \\ 1 & \beta_1 & \beta_1^2 & \beta_1^3 & \cdots & \beta_1^{n-1} \\ 1 & \beta_2 & \beta_2^2 & \beta_2^3 & \cdots & \beta_2^{n-1} \\ 1 & \beta_3 & \beta_3^2 & \beta_3^3 & \cdots & \beta_3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \beta_{n-1} & \beta_{n-1}^2 & \beta_{n-1}^3 & \cdots & \beta_{n-1}^{n-1} \end{pmatrix}}_{V^{-1}} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix}.$$

3. $W(x) = \sum_{i=0}^{n-1} (y_i/n) x^i$: $\qquad a_i = W(\beta_i)$.

Recap...

1. n point-value pairs $\{(\gamma_0, y_0), \ldots, (\gamma_{n-1}, y_{n-1})\}$: of polynomial $p(x) = \sum_{i=0}^{n-1} a_i x^i$ over nth roots of unity.
2. Recover coefficients of polynomial by multiplying $[y_0, y_1, \ldots, y_n]$ by V^{-1}:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} = \frac{1}{n} \underbrace{\begin{pmatrix} 1 & \beta_0 & \beta_0^2 & \beta_0^3 & \cdots & \beta_0^{n-1} \\ 1 & \beta_1 & \beta_1^2 & \beta_1^3 & \cdots & \beta_1^{n-1} \\ 1 & \beta_2 & \beta_2^2 & \beta_2^3 & \cdots & \beta_2^{n-1} \\ 1 & \beta_3 & \beta_3^2 & \beta_3^3 & \cdots & \beta_3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \beta_{n-1} & \beta_{n-1}^2 & \beta_{n-1}^3 & \cdots & \beta_{n-1}^{n-1} \end{pmatrix}}_{V^{-1}} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix}.$$

3. $W(x) = \sum_{i=0}^{n-1} (y_i/n) x^i$: $\qquad a_i = W(\beta_i)$.

Recap...

1. n point-value pairs $\{(\gamma_0, y_0), \ldots, (\gamma_{n-1}, y_{n-1})\}$: of polynomial $p(x) = \sum_{i=0}^{n-1} a_i x^i$ over nth roots of unity.
2. Recover coefficients of polynomial by multiplying $[y_0, y_1, \ldots, y_n]$ by V^{-1}:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} = \frac{1}{n} \underbrace{\begin{pmatrix} 1 & \beta_0 & \beta_0^2 & \beta_0^3 & \cdots & \beta_0^{n-1} \\ 1 & \beta_1 & \beta_1^2 & \beta_1^3 & \cdots & \beta_1^{n-1} \\ 1 & \beta_2 & \beta_2^2 & \beta_2^3 & \cdots & \beta_2^{n-1} \\ 1 & \beta_3 & \beta_3^2 & \beta_3^3 & \cdots & \beta_3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \beta_{n-1} & \beta_{n-1}^2 & \beta_{n-1}^3 & \cdots & \beta_{n-1}^{n-1} \end{pmatrix}}_{V^{-1}} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{pmatrix}.$$

3. $W(x) = \sum_{i=0}^{n-1} (y_i/n) x^i$: $\qquad a_i = W(\beta_i)$.

Recovering continued...

1. recover coefficients of $p(\cdot)$...
2. ... compute $W(\cdot)$ on n values: $\beta_0, \ldots, \beta_{n-1}$.
3. $\{\beta_0, \ldots, \beta_{n-1}\} = \{\gamma_0, \ldots, \gamma_{n-1}\}$.
4. Indeed $\beta_i^n = (\gamma_i^{-1})^n = (\gamma_i^n)^{-1} = 1^{-1} = 1$.
5. Apply the **FFTAlg** algorithm on $W(x)$ to compute $a_0, \ldots, a_{n-1}$.

Recovering continued...

1. recover coefficients of $p(\cdot)$...
2. ... compute $W(\cdot)$ on n values: $\beta_0, \ldots, \beta_{n-1}$.
3. $\{\beta_0, \ldots, \beta_{n-1}\} = \{\gamma_0, \ldots, \gamma_{n-1}\}$.
4. Indeed $\beta_i^n = (\gamma_i^{-1})^n = (\gamma_i^n)^{-1} = 1^{-1} = 1$.
5. Apply the **FFTAlg** algorithm on $W(x)$ to compute $a_0, \ldots, a_{n-1}$.

Recovering continued...

1. recover coefficients of $p(\cdot)$...
2. ... compute $W(\cdot)$ on n values: $\beta_0, \ldots, \beta_{n-1}$.
3. $\{\beta_0, \ldots, \beta_{n-1}\} = \{\gamma_0, \ldots, \gamma_{n-1}\}$.
4. Indeed $\beta_i^n = (\gamma_i^{-1})^n = (\gamma_i^n)^{-1} = 1^{-1} = 1$.
5. Apply the **FFTAlg** algorithm on $W(x)$ to compute $a_0, \ldots, a_{n-1}$.

Recovering continued...

1. recover coefficients of $p(\cdot)$...
2. ... compute $W(\cdot)$ on n values: $\beta_0, \ldots, \beta_{n-1}$.
3. $\{\beta_0, \ldots, \beta_{n-1}\} = \{\gamma_0, \ldots, \gamma_{n-1}\}$.
4. Indeed $\beta_i^n = (\gamma_i^{-1})^n = (\gamma_i^n)^{-1} = 1^{-1} = 1$.
5. Apply the **FFTAlg** algorithm on $W(x)$ to compute $a_0, \ldots, a_{n-1}$.

Recovering continued...

1. recover coefficients of $p(\cdot)$...
2. ... compute $W(\cdot)$ on n values: $\beta_0, \ldots, \beta_{n-1}$.
3. $\{\beta_0, \ldots, \beta_{n-1}\} = \{\gamma_0, \ldots, \gamma_{n-1}\}$.
4. Indeed $\beta_i^n = (\gamma_i^{-1})^n = (\gamma_i^n)^{-1} = 1^{-1} = 1$.
5. Apply the **FFTAlg** algorithm on $W(x)$ to compute $a_0, \ldots, a_{n-1}$.

Result

Theorem

Given n point-value pairs of a polynomial $p(x)$ of degree $n-1$ over the set of n powers of the nth roots of unity, we can recover the polynomial $p(x)$ in $O(n \log n)$ time.

Theorem

Given two polynomials of degree n, they can be multiplied in $O(n \log n)$ time.

Part II

Convolutions

Convolutions

1. Two vectors: $A = [a_0, a_1, \ldots, a_n]$ and $B = [b_0, \ldots, b_n]$.
2. ***dot product*** $A \cdot B = \langle A, B \rangle = \sum_{i=0}^{n} a_i b_i$.
3. A_r: shifting of A by $n - r$ locations to the left
4. Padded with zeros:, $a_j = 0$ for $j \notin \{0, \ldots, n\}$).
5. $A_r = \left[a_{n-r}, a_{n+1-r}, a_{n+2-r}, \ldots, a_{2n-r}\right]$
   where $a_j = 0$ if $j \notin \left[0, \ldots, n\right]$.
6. **Observation**: $A_n = A$.

Convolutions

1. Two vectors: $A = [a_0, a_1, \ldots, a_n]$ and $B = [b_0, \ldots, b_n]$.
2. ***dot product*** $A \cdot B = \langle A, B \rangle = \sum_{i=0}^{n} a_i b_i$.
3. A_r: shifting of A by $n - r$ locations to the left
4. Padded with zeros:, $a_j = 0$ for $j \notin \{0, \ldots, n\}$).
5. $A_r = \left[a_{n-r}, a_{n+1-r}, a_{n+2-r}, \ldots, a_{2n-r}\right]$
   where $a_j = 0$ if $j \notin \left[0, \ldots, n\right]$.
6. **Observation**: $A_n = A$.

Convolutions

1. Two vectors: $A = [a_0, a_1, \ldots, a_n]$ and $B = [b_0, \ldots, b_n]$.
2. ***dot product*** $A \cdot B = \langle A, B \rangle = \sum_{i=0}^{n} a_i b_i$.
3. A_r: shifting of A by $n - r$ locations to the left
4. Padded with zeros:, $a_j = 0$ for $j \notin \{0, \ldots, n\}$).
5. $A_r = \left[a_{n-r}, a_{n+1-r}, a_{n+2-r}, \ldots, a_{2n-r}\right]$ where $a_j = 0$ if $j \notin \left[0, \ldots, n\right]$.
6. **Observation**: $A_n = A$.

Convolutions

1. Two vectors: $A = [a_0, a_1, \ldots, a_n]$ and $B = [b_0, \ldots, b_n]$.
2. ***dot product*** $A \cdot B = \langle A, B \rangle = \sum_{i=0}^{n} a_i b_i$.
3. A_r: shifting of A by $n - r$ locations to the left
4. Padded with zeros:, $a_j = 0$ for $j \notin \{0, \ldots, n\}$).
5. $A_r = \left[a_{n-r}, a_{n+1-r}, a_{n+2-r}, \ldots, a_{2n-r}\right]$ where $a_j = 0$ if $j \notin \left[0, \ldots, n\right]$.
6. **Observation**: $A_n = A$.

Convolutions

1. Two vectors: $A = [a_0, a_1, \ldots, a_n]$ and $B = [b_0, \ldots, b_n]$.
2. ***dot product*** $A \cdot B = \langle A, B \rangle = \sum_{i=0}^{n} a_i b_i$.
3. A_r: shifting of A by $n - r$ locations to the left
4. Padded with zeros:, $a_j = 0$ for $j \notin \{0, \ldots, n\}$).
5. $A_r = \left[a_{n-r}, a_{n+1-r}, a_{n+2-r}, \ldots, a_{2n-r}\right]$
   where $a_j = 0$ if $j \notin \left[0, \ldots, n\right]$.
6. **Observation**: $A_n = A$.

Convolutions

1. Two vectors: $A = [a_0, a_1, \ldots, a_n]$ and $B = [b_0, \ldots, b_n]$.
2. ***dot product*** $A \cdot B = \langle A, B \rangle = \sum_{i=0}^{n} a_i b_i$.
3. A_r: shifting of A by $n - r$ locations to the left
4. Padded with zeros:, $a_j = 0$ for $j \notin \{0, \ldots, n\}$).
5. $A_r = \left[a_{n-r}, a_{n+1-r}, a_{n+2-r}, \ldots, a_{2n-r}\right]$
   where $a_j = 0$ if $j \notin \left[0, \ldots, n\right]$.
6. **Observation**: $A_n = A$.

Example of shifting

Example

For $A = [3, 7, 9, 15]$, $n = 3$
$A_2 = [7, 9, 15, 0]$,
$A_5 = [0, 0, 3, 7]$.

Definition

Definition

Let $c_i = A_i \cdot B = \sum_{j=n-i}^{2n-i} a_j b_{j-n+i}$, for $i = 0, \ldots, 2n$. The vector $[c_0, \ldots, c_{2n}]$ is the ***convolution*** of A and B.

question

How to compute the convolution of two vectors of length n?

Convolution via multiplication polynomials

1. $p(x) = \sum_{i=0}^{n} \alpha_i x^i$, and $q(x) = \sum_{i=0}^{n} \beta_i x^i$.
2. Coefficient of x^i in $r(x) = p(x)q(x)$ is $d_i = \sum_{j=0}^{i} \alpha_j \beta_{i-j}$.
3. Want to compute $c_i = A_i \cdot B = \sum_{j=n-i}^{2n-i} a_j b_{j-n+i}$.
4. Set $\alpha_i = a_i$ and $\beta_l = b_{n-l-1}$.

Convolution via multiplication polynomials

1. $p(x) = \sum_{i=0}^{n} \alpha_i x^i$, and $q(x) = \sum_{i=0}^{n} \beta_i x^i$.
2. Coefficient of x^i in $r(x) = p(x)q(x)$ is $d_i = \sum_{j=0}^{i} \alpha_j \beta_{i-j}$.
3. Want to compute $c_i = A_i \cdot B = \sum_{j=n-i}^{2n-i} a_j b_{j-n+i}$.
4. Set $\alpha_i = a_i$ and $\beta_l = b_{n-l-1}$.

Convolution via multiplication polynomials

1. $p(x) = \sum_{i=0}^{n} \alpha_i x^i$, and $q(x) = \sum_{i=0}^{n} \beta_i x^i$.
2. Coefficient of x^i in $r(x) = p(x)q(x)$ is $d_i = \sum_{j=0}^{i} \alpha_j \beta_{i-j}$.
3. Want to compute $c_i = A_i \cdot B = \sum_{j=n-i}^{2n-i} a_j b_{j-n+i}$.
4. Set $\alpha_i = a_i$ and $\beta_l = b_{n-l-1}$.

Convolution via multiplication polynomials

1. $p(x) = \sum_{i=0}^{n} \alpha_i x^i$, and $q(x) = \sum_{i=0}^{n} \beta_i x^i$.
2. Coefficient of x^i in $r(x) = p(x)q(x)$ is $d_i = \sum_{j=0}^{i} \alpha_j \beta_{i-j}$.
3. Want to compute $c_i = A_i \cdot B = \sum_{j=n-i}^{2n-i} a_j b_{j-n+i}$.
4. Set $\alpha_i = a_i$ and $\beta_l = b_{n-l-1}$.

Convolution by example

1. Consider coefficient of x^2 in product of $p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ and $q(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3$.
2. Sum of the entries on the anti diagonal:

	a_0+	$a_1 x$	$+a_2 x^2$	$+a_3 x^3$
b_0			$a_2 b_0 x^2$	
$+b_1 x$		$a_1 b_1 x^2$		
$+b_2 x^2$	$a_0 b_2 x^2$			
$+b_3 x^3$				

3. entry in the ith row and jth column is $a_i b_j$.

Convolution by example

1. Consider coefficient of x^2 in product of $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ and $q(x) = b_0 + b_1x + b_2x^2 + b_3x^3$.
2. Sum of the entries on the anti diagonal:

	a_0+	a_1x	$+a_2x^2$	$+a_3x^3$
b_0			$a_2b_0x^2$	
$+b_1x$		$a_1b_1x^2$		
$+b_2x^2$	$a_0b_2x^2$			
$+b_3x^3$				

3. entry in the ith row and jth column is a_ib_j.

Convolution by example

1. Consider coefficient of x^2 in product of $p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ and $q(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3$.
2. Sum of the entries on the anti diagonal:

	a_0+	$a_1 x$	$+a_2 x^2$	$+a_3 x^3$
b_0			$a_2 b_0 x^2$	
$+b_1 x$		$a_1 b_1 x^2$		
$+b_2 x^2$	$a_0 b_2 x^2$			
$+b_3 x^3$				

3. entry in the ith row and jth column is $a_i b_j$.

Convolution

Theorem

Given two vectors $A = [a_0, a_1, \ldots, a_n]$, $B = [b_0, \ldots, b_n]$ *one can compute their convolution in* $O(n \log n)$ *time.*

Proof.

Let $p(x) = \sum_{i=0}^{n} a_{n-i} x^i$ and let $q(x) = \sum_{i=0}^{n} b_i x^i$. Compute $r(x) = p(x) q(x)$ in $O(n \log n)$ time using the convolution theorem. Let $c_0, \ldots, c_{2n}$ be the coefficients of $r(x)$. It is easy to verify, as described above, that $[c_0, \ldots, c_{2n}]$ is the convolution of A and B. $\square$

Notes

Notes

Notes

Notes