Chapter 1
Divide and Conquer

Closest Pair,
Polynomial Multiplication

Algorithm Theory
WS 2014/15

Fabian Kuhn

Formulation of the D&C principle

Divide-and-conquer method for solving a problem instance of size n:

1. Divide

$n \leq c$: Solve the problem directly.

$n > c$: Divide the problem into k subproblems of sizes $n_1, \ldots, n_k < n$ $(k \geq 2)$.

2. Conquer

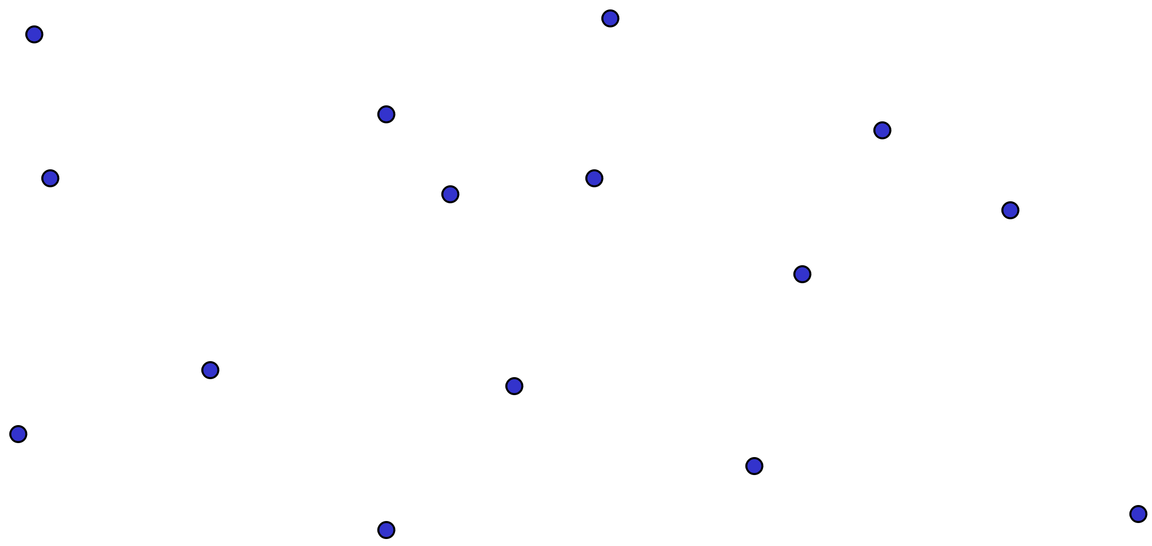
Solve the k subproblems in the same way (recursively).

3. Combine

Combine the partial solutions to generate a solution for the original instance.

Geometric divide-and-conquer

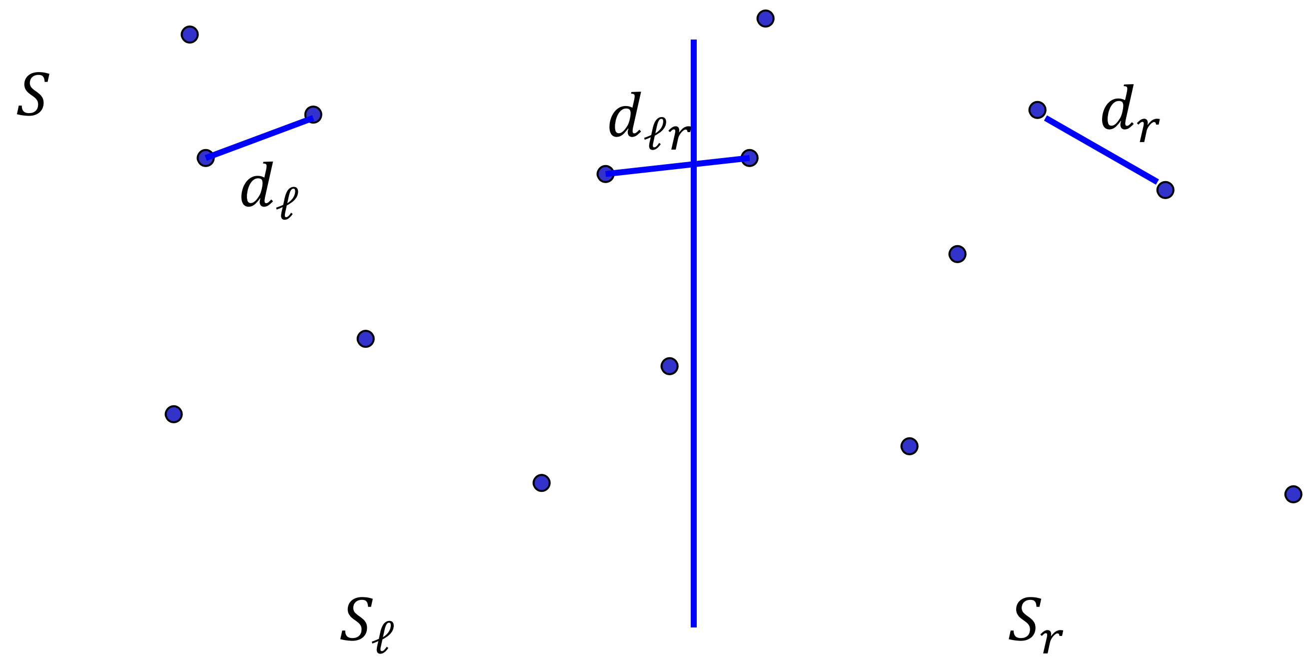
Closest Pair Problem: Given a set S of n points, find a pair of points with the smallest distance.



Naive solution:

Divide-and-conquer solution

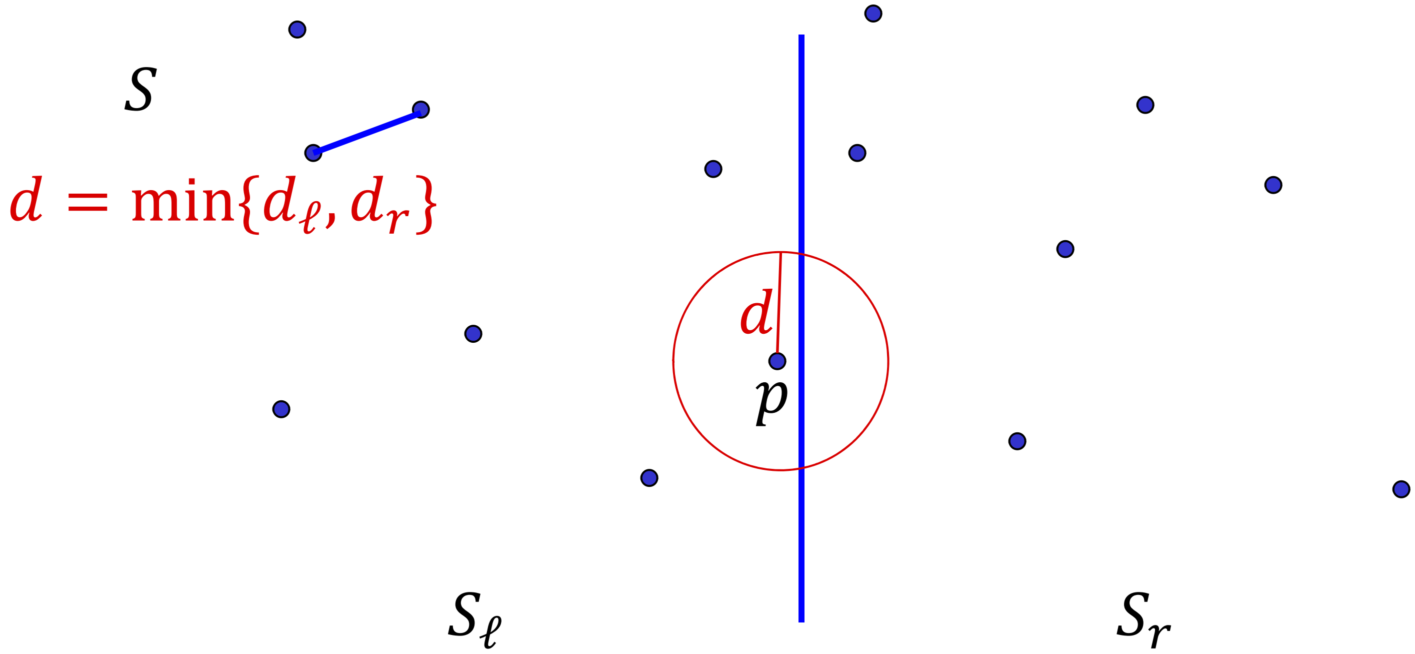
1. **Divide:** Divide S into two equal sized sets S_ℓ und S_r.
2. **Conquer:** $d_\ell = \mathrm{mindist}(S_\ell)$ $\quad d_r = \mathrm{mindist}(S_r)$
3. **Combine:** $d_{\ell r} = \min\{d(p_\ell, p_r) \mid p_\ell \in S_\ell, p_r \in S_r\}$
   return $\min\{d_\ell, d_r, d_{\ell r}\}$



Divide-and-conquer solution

1. **Divide:** Divide S into two equal sized sets S_ℓ und S_r.
2. **Conquer:** $d_\ell = \text{mindist}(S_\ell)$ $\quad d_r = \text{mindist}(S_r)$
3. **Combine:** $d_{\ell r} = \min\{d(p_\ell, p_r) \mid p_\ell \in S_\ell, p_r \in S_r\}$
   return $\min\{d_\ell, d_r, d_{\ell r}\}$

Computation of $d_{\ell r}$:



Merge step

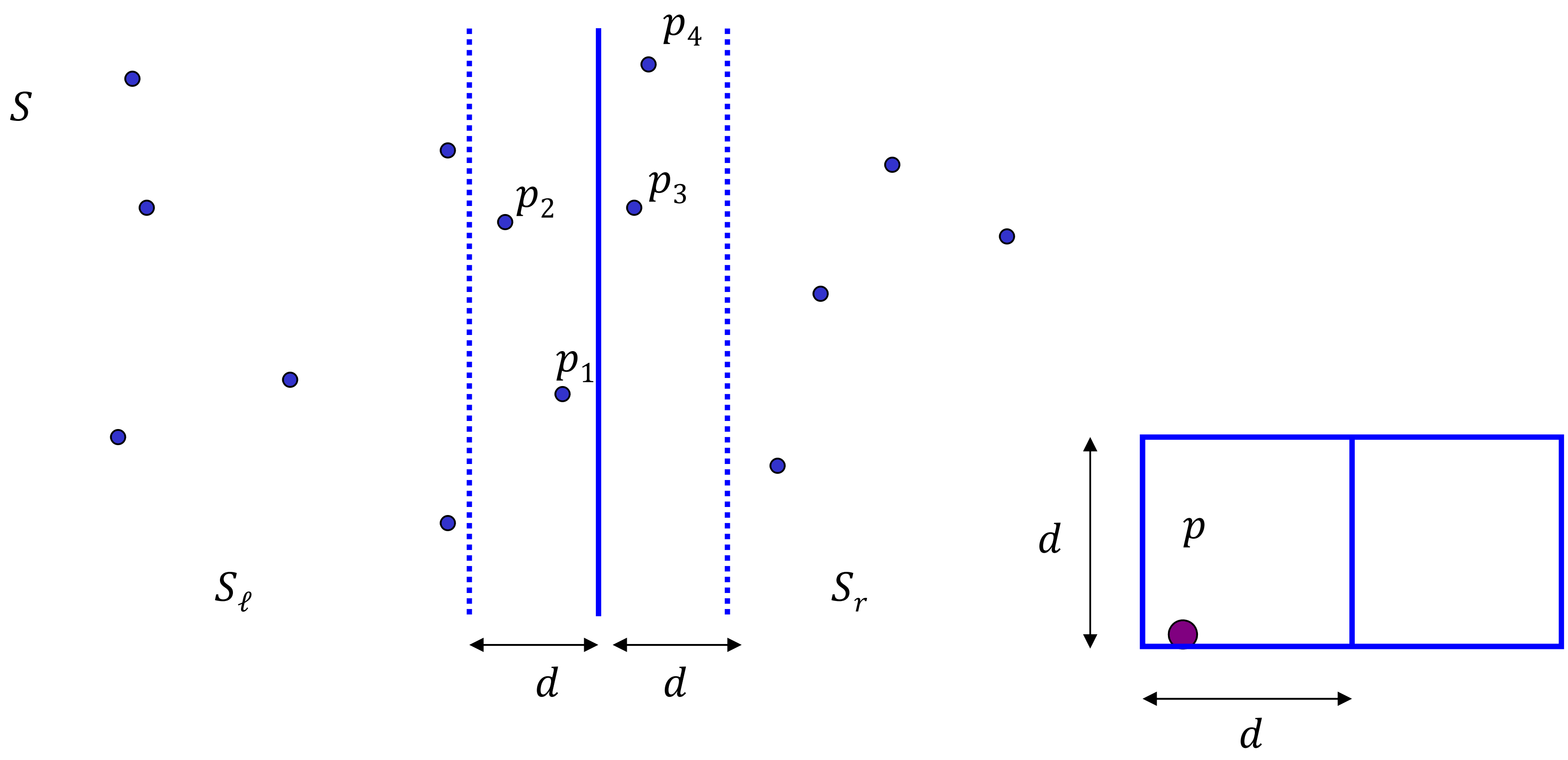
- Assume that the points in both halfs are sorted by increasing y-coordinates

1. Consider only points within distance $< d$ of the bisection line, in the order of increasing y-coordinates.

2. For each point p consider all points q within y-distance less than d

3. There are at most 7 such points.

Combine step



$d = \min\{d_\ell, dr\}$

Implementation

- Initially sort the points in S in order of increasing x-coordinates
- While computing closest pair, also sort S according to y-coord.
  - Partition S into S_ℓ and S_r, solve and sort sub-problems recursively
  - Merge to get sorted S according to y-coordinates
  - Center points: points within x-distance $d = \min\{d_\ell, d_r\}$ of center
  - Go through center points in S in order of incr. y-coordinates

Running Time

Recurrence relation:

$$T(n) = 2 \cdot T(n/2) + c \cdot n, \qquad T(1) = a$$

Solution:

- Same as for computing number of number of inversions, merge sort (and many others…)

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

Recurrence Relations: Master Theorem

Recurrence relation

$$\boldsymbol{T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n), \qquad T(n) = O(1) \text{ for } n \leq n_0}$$

Cases

- $f(n) = O(n^c),\ c < \log_b a$

  $$\boldsymbol{T(n) = \Theta\left(n^{\log_b a}\right)}$$

- $f(n) = \Omega(n^c),\ c > \log_b a$

  $$\boldsymbol{T(n) = \Theta\big(f(n)\big)}$$

- $f(n) = \Theta\left(n^c \cdot \log^k n\right),\ c = \log_b a$

  $$\boldsymbol{T(n) = \Theta\left(n^c \cdot \log^{k+1} n\right)}$$

Polynomials

Real polynomial p in one variable x:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x^1 + a_0$$

Coefficients of p: $a_0, a_1, \dots, a_n \in \mathbb{R}$

Degree of p: largest power of x in p (n in the above case)

Example:

$$p(x) = 3x^3 - 15x^2 + 18x$$

Set of all real-valued polynomials in x: $\mathbb{R}[x]$ (polynomial ring)

Operations: Addition

- Given: Polynomials $p, q \in \mathbb{R}[x]$ of degree n

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$
$$q(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_1 x + b_0$$

- Compute sum $p(x) + q(x)$:

$$\begin{aligned} p(x) + q(x) &= (a_n x^n + \cdots + a_0) + (b_n x^n + \cdots + b_0) \\ &= (a_n + b_n) x^n + \cdots + (a_1 + b_1) x + (a_0 + b_0) \end{aligned}$$

Operations: Multiplication

- Given: Polynomials $p, q \in \mathbb{R}[x]$ of degree n

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$
$$q(x) = b_n x^n + b_{n-1} x^{n-1} + \cdots + b_1 x + b_0$$

- Product $p(x) \cdot q(x)$:

$$p(x) \cdot q(x) = (a_n x^n + \cdots + a_0) \cdot (b_n x^n + \cdots + b_0)$$

$$= c_{2n} x^{2n} + c_{2n-1} x^{2n-1} + \cdots + c_1 x + c_0$$

- Obtaining c_i: what products of monomials have degree i?

$$\text{For } 0 \leq i \leq 2n\text{: } c_i = \sum_{j=0}^{i} a_j b_{i-j}$$

where $a_i = b_i = 0$ for $i > n$.

Operations: Evaluation

- Given: Polynomial $p \in \mathbb{R}[x]$ of degree n

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

- **Horner's method** for evaluation at specific value x_0:

$$p(x_0) = \big(\dots\big((a_n x_0 + a_{n-1})x_0 + a_{n-2}\big)x_0 + \cdots + a_1\big)x_0 + a_0$$

- Pseudo-code:

$p \coloneqq a_n;\, i \coloneqq n;$
while $(i > 0)$ **do**
    $i \coloneqq i - 1;$
    $p \coloneqq p \cdot x_0 + a_i$
end

- Running time: $O(n)$

Representation of Polynomials

Coefficient representation:

- Polynomial $p(x) \in \mathbb{R}[x]$ of degree n is given by its $n+1$ coefficients $a_0, \dots, a_n$:

$$p(x) = a_n x^n + \cdots + a_1 x + a_0$$

- Example:

$$p(x) = 3x^3 - 15x^2 + 18x$$

- The most typical (and probably most natural) representation of polynomials

Representation of Polynomials

Product of linear factors:

- Polynomial $p(x) \in \mathbb{C}[x]$ of degree n is given by its n roots

$$p(x) = a_n \cdot (x - x_1) \cdot (x - x_2) \cdot \ldots \cdot (x - x_n)$$

- Example:

$$p(x) = 3x(x - 2)(x - 3)$$

- Every polynomial has exactly n roots $x_i \in \mathbb{C}$ for which $p(x_i) = 0$
  - Polynomial is uniquely defined by the n roots and a_n
- We will not use this representation…

Representation of Polynomials

Point-value representation:

- Polynomial $p(x) \in \mathbb{R}[x]$ of degree n is given by $n+1$ point-value pairs:

$$p = \{(x_0, p(x_0)), (x_1, p(x_1)), \dots, (x_n, p(x_n))\}$$

where $x_i \neq x_j$ for $i \neq j$.

- Example: The polynomial

$$p(x) = 3x(x-2)(x-3)$$

is uniquely defined by the four point-value pairs $(0,0), (1,6), (2,0), (3,0)$.

Operations: Coefficient Representation

Deg.-n polynomials $p(x) = a_n x^n + \cdots + a_0$, $q(x) = b_n x^n + \cdots + b_0$

Addition:

$$p(x) + q(x) = (a_n + b_n)x^n + \cdots + (a_0 + b_0)$$

- Time: $O(n)$

Multiplication:

$$p(x) \cdot q(x) = c_{2n}x^{2n} + \cdots + c_0, \qquad \text{where } c_i = \sum_{j=0}^{i} a_j b_{i-j}$$

- Naive solution: Need to compute product $a_i b_j$ for all $0 \leq i, j \leq n$
- Time: $O(n^2)$

Operations Point-Value Representation

Degree-n polynomials

$$p = \{(x_0, p(x_0)), \dots, (x_n, p(x_n))\}, q = \{(x_0, q(x_0)), \dots, (x_n, q(x_n))\}$$

- Note: we use the same points $x_0, \dots, x_n$ for both polynomials

Addition:

$$p + q = \{(x_0, p(x_0) + q(x_0)), \dots, (x_n, p(x_n) + q(x_n))\}$$

- Time: $O(n)$

Multiplication:

$$p \cdot q = \{(x_0, p(x_0) \cdot q(x_0)), \dots, (x_n, p(x_n) \cdot q(x_n))\}$$

- Time: $O(n)$

Faster Multiplication?

- Multiplication is slow $\left(\Theta(n^2)\right)$ when using the standard coefficient representation

- Try divide-and-conquer to get a faster algorithm

- Assume: degree is $n-1$, n is even
- Divide polynomial $p(x) = a_{n-1}x^{n-1} + \cdots + a_0$ into 2 polynomials of degree ${}^{n}/_{2} - 1$:

$$p_0(x) = a_{n/_2-1}x^{n/_2-1} + \cdots + a_0$$
$$p_1(x) = a_{n-1}x^{n/_2-1} + \cdots + a_{n/_2}$$

$$p(x) = p_1(x) \cdot x^{n/_2} + p_0(x)$$

- Similarly: $q(x) = q_1(x) \cdot x^{n/_2} + q_0(x)$

Use Divide-And-Conquer

- **Divide:**
  $p(x) = p_1(x) \cdot x^{n/2} + p_0(x), \qquad q(x) = q_1(x) \cdot x^{n/2} + q_0(x)$

- **Multiplication:**
  $$p(x)q(x) = p_1(x)q_1(x) \cdot x^n + (p_0(x)q_1(x) + p_1(x)q_0(x)) \cdot x^{n/2} + p_0(x)q_0(x)$$

- 4 multiplications of degree $n/2 - 1$ polynomials:
  $$T(n) = 4T(n/2) + O(n)$$

- Leads to $T(n) = \Theta(n^2)$ like the naive algorithm... (see exercises)

More Clever Recursive Solution

- **Recall that**

$$p(x)q(x) = p_1(x)q_1(x) \cdot x^n + (p_0(x)q_1(x) + p_1(x)q_0(x)) \cdot x^{n/2} + p_0(x)q_0(x)$$

- Compute $r(x) = (p_0(x) + p_1(x)) \cdot (q_0(x) + q_1(x))$:

Karatsuba Algorithm

- Recursive multiplication:

$$r(x) = (p_0(x) + p_1(x)) \cdot (q_0(x) + q_1(x))$$

$$\begin{aligned} p(x)q(x) = p_1(x)q_1(x) \cdot x^n \\ + (r(x) - p_0(x)q_0(x) + p_1(x)q_1(x)) \cdot x^{n/2} \\ + p_0(x)q_0(x) \end{aligned}$$

- Recursively do 3 multiplications of degr. $(n/2 - 1)$-polynomials

$$T(n) = 3T(n/2) + O(n)$$

- Gives: $T(n) = O(n^{1.59})$ (see Master theorem)

Faster Polynomial Multiplication?

Multiplication is fast when using the point-value representation

Idea to compute $p(x) \cdot q(x)$ (for polynomials of degree $< n$):

p, q of degree $n-1$, n coefficients

↓ **Evaluation** at points $x_0, x_1, \dots, x_{2n-1}$

$2 \times 2n$ point-value pairs $(x_i, p(x_i))$ and $(x_i, q(x_i))$

↓ **Point-wise multiplication**

$2n$ point-value pairs $(x_i, p(x_i)q(x_i))$

↓ **Interpolation**

$p(x)q(x)$ of degree $2n-2$, $2n-1$ coefficients

Point-Value Representation of p, q

- Select points $x_0, x_1, \dots, x_{N-1}$ to evaluate p and q in a clever way

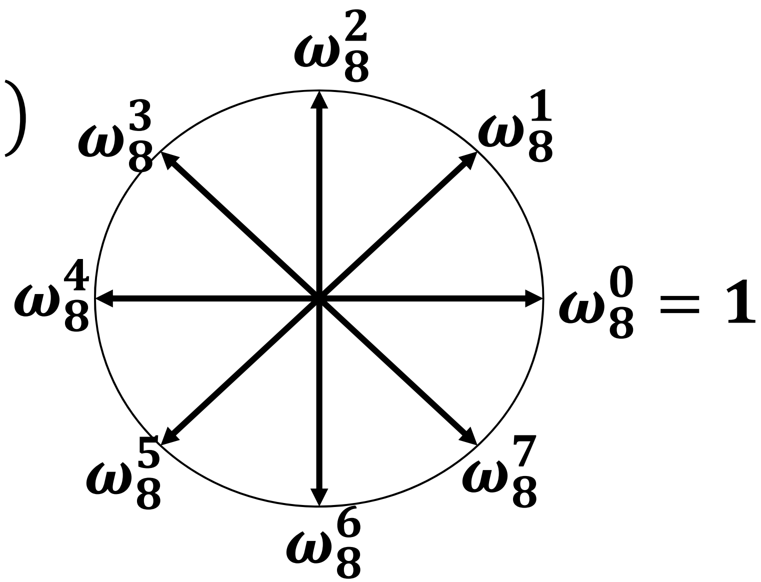
Consider the N powers of the principle Nth root of unity:

Principle root of unity: $\boldsymbol{\omega_N = e^{2\pi i/N}}$

$$\left(i = \sqrt{-1}, \qquad e^{2\pi i} = 1\right)$$

Powers of $\boldsymbol{\omega_n}$ (roots of unity):

$$1 = \omega_N^0, \omega_N^1, \dots, \omega_N^{N-1}$$



Note: $\omega_N^k = e^{2\pi i k/N} = \cos\frac{2\pi k}{N} + i \cdot \sin\frac{2\pi k}{N}$

Discrete Fourier Transform

- The values $p(\omega_N^i)$ for $i = 0, \ldots, N-1$ uniquely define a polynomial p of degree $< N$.

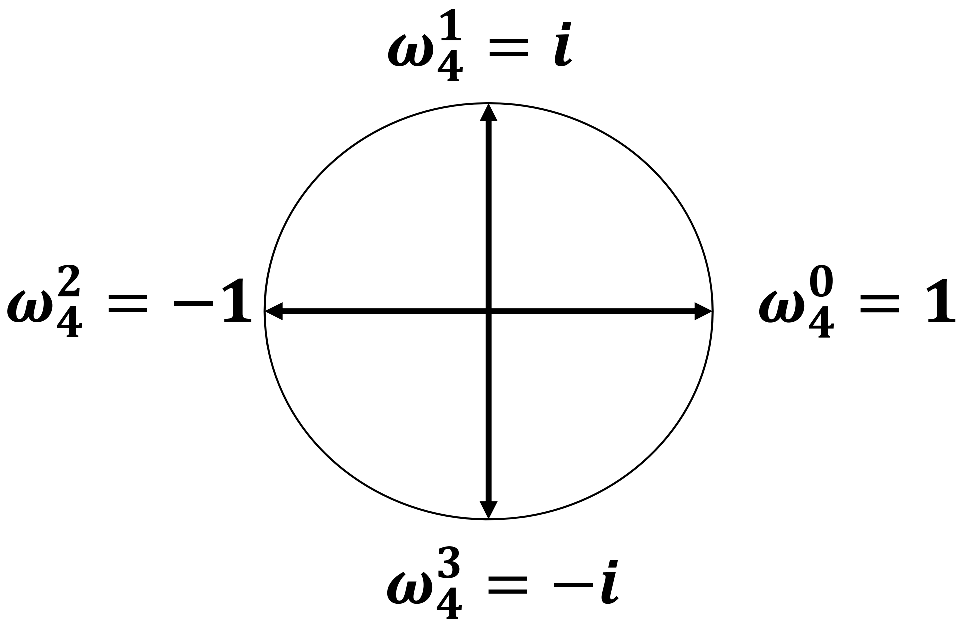
Discrete Fourier Transform (DFT):

- Assume $a = (a_0, \ldots, a_{N-1})$ is the coefficient vector of poly. p
$$(p(x) = a_{N-1}x^{N-1} + \cdots + a_1 x + a_0)$$

$$\mathrm{DFT}_N(a) \coloneqq \left(p(\omega_N^0), p(\omega_N^1), \ldots, p(\omega_N^{N-1})\right)$$

Example

- Consider polynomial $p(x) = 3x^3 - 15x^2 + 18x$
- Choose $N = 4$
- Roots of unity:



Example

- Consider polynomial $p(x) = 3x^3 - 15x^2 + 18x$

- $N = 4$, roots of unity: $\omega_4^0 = 1, \omega_4^1 = i, \omega_4^2 = -1, \omega_4^3 = -i$

- Evaluate $p(x)$ at ω_4^k:

$$\left(\omega_4^0, p(\omega_4^0)\right) = \left(1, p(1)\right) = (1,6)$$
$$\left(\omega_4^1, p(\omega_4^1)\right) = \left(i, p(i)\right) = (i, 15 + 15i)$$
$$\left(\omega_4^2, p(\omega_4^2)\right) = \left(-1, p(-1)\right) = (-1, -36)$$
$$\left(\omega_4^3, p(\omega_4^3)\right) = \left(-i, p(-i)\right) = (-i, 15 - 15i)$$

- For $a = (3, -15, 18, 0)$:

$$\mathbf{DFT_4}(\boldsymbol{a}) = (\mathbf{6}, \mathbf{15} + \mathbf{15}\boldsymbol{i}, -\mathbf{36}, \mathbf{15} - \mathbf{15}\boldsymbol{i})$$

Faster Polynomial Multiplication?

Idea to compute $p(x) \cdot q(x)$ (for polynomials of degree $< n$):

p, q of degree $n-1$, n coefficients

↓ **Evaluation** at points $\omega_{2n}^{0}, \omega_{2n}^{1}, \dots, \omega_{2n}^{2n-1}$

$2 \times 2n$ point-value pairs $\left(\omega_{2n}^{k}, p\left(\omega_{2n}^{k}\right)\right)$ and $\left(\omega_{2n}^{k}, q\left(\omega_{2n}^{k}\right)\right)$

↓ **Point-wise multiplication**

$2n$ point-value pairs $\left(\omega_{2n}^{k}, p\left(\omega_{2n}^{k}\right) q\left(\omega_{2n}^{k}\right)\right)$

↓ **Interpolation**

$p(x)q(x)$ of degree $2n-2$, $2n-1$ coefficients

Properties of the Roots of Unity

- **Cancellation Lemma:**

  For all integers $n > 0$, $k \geq 0$, and $d > 0$, we have:

$$\boldsymbol{\omega_{dn}^{dk} = \omega_n^k}, \qquad \boldsymbol{\omega_n^{k+n} = \omega_n^k}$$

- **Proof:**