Chapter 1
Divide and Conquer

Algorithm Theory
WS 2018/19

Fabian Kuhn

Divide-And-Conquer Principle

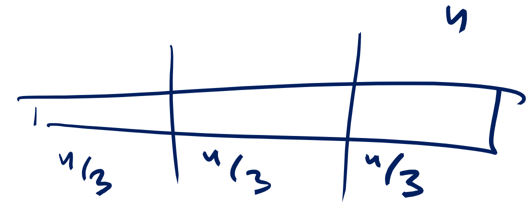
- Important algorithm design method

- Examples from basic alg. & data structures class (Informatik 2):
  - Sorting: Mergesort, Quicksort
  - Binary search

- Further examples
  - Median
  - Compairing orders
  - Convex hull / Delaunay triangulation / Voronoi diagram
  - Closest pairs
  - Line intersections
  - Polynomial multiplication / FFT
  - ...

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 $n_1, \dots, n_k < n$ ($k \geq 2$).

quick sort

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.

merge sort

Running Time

Recurrence relation:

$$T(n) = \overbrace{2}^{\#\text{subpr.}} \cdot \overbrace{T(n/2)}^{\text{size of subproblems}} + \underbrace{c \cdot n}_{\text{cost for divide \& combine}}, \qquad T(1) = a$$

Solution:

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

$$\boxed{T(n) = O(n \cdot \log n)}$$

Recurrence Relations: Master Theorem

Recurrence relation

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

$n^{\log_b a}$

Cases

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

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

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

$$T(n) = \Theta\left(f(n)\right)$$

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

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

mergesort
$a = b = 2$
$f(n) = n$
$\Rightarrow c = 1$

Polynomials

Real polynomial p in one variable x: n coefficients

$a_i \in \mathbb{R}$

$$p(x) = a_{n-1}x^{n-1} + \ldots + a_1x^1 + a_0$$

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

Degree of p: largest power of x in p ($n-1$ 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)

Divide-&-Conquer Polynomial 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   n is power of 2
- Divide polynomial $p(x) = a_{n-1}x^{n-1} + \cdots + a_0$ into 2 polynomials of degree ${}^{n}/_{2} - 1$:   Karatsuba

$$p_0(x) = \underline{a_{n/2-1}}\, x^{n/2-1} + \cdots + \underline{a_0}$$

$$p_1(x) = \underline{a_{n-1}}\, x^{n/2-1} + \cdots + \underline{a_{n/2}}$$

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

- Similarly: $q(x) = q_1(x) \cdot x^{n/2} + q_0(x)$

Divide-&-Conquer Polynomial Multiplication

- **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:

$n^{\log_2 4} = n^2$

$$T(n) = 4T(n/2) + O(n)$$

- Leads to $T(n) = \Theta(n^2)$ like the naive algorithm…
  - follows immediately by using the master theorem

Karatsuba Algorithm

- Recursive multiplication:

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

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

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

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

- Gives: $T(n) = O(n^{1.58496\ldots})$ (see Master theorem)

Representation of Polynomials

Coefficient representation:

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

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

- Coefficient vector $\boldsymbol{a} = \underbrace{(a_0, a_1, \dots a_{n-1})}$

- Example:

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

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

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

Representation of Polynomials

Point-value representation:

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

$$p = \{(x_0, p(x_0)), (x_1, p(x_1)), \dots, (x_{n-1}, p(x_{n-1}))\}$$

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

$$\boldsymbol{p(x) = a_{n-1}x^{n-1} + \cdots + a_0, \qquad q(x) = b_{n-1}x^{n-1} + \cdots + b_0}$$

Evaluation: Horner's method: Time $O(n)$

Addition:

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

- Time: $O(n)$

Multiplication:

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

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

Operations: Point-Value Representation

$$\boldsymbol{p = \{(x_0, p(x_0)), \dots, (x_{n-1}, p(x_{n-1}))\}}$$
$$\boldsymbol{q = \{(x_0, q(x_0)), \dots, (x_{n-1}, q(x_{n-1}))\}}$$

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

Addition:

$$p + q = \{(x_0, p(x_0) + q(x_0)), \dots, (x_{n-1}, p(x_{n-1}) + q(x_{n-1}))\}$$

- Time: $O(n)$

Multiplication:

$$p \cdot q = \{(x_0, p(x_0) \cdot q(x_0)), \dots, (x_{2n-2}, p(x_{2n-2}) \cdot q(x_{2n-2}))\}$$

- Time: $O(n)$

Evaluation: Polynomial interpolation can be done in $O(n^2)$

Operations on Polynomials

Cost depending on representation:

	Coefficient	Roots	Point-Value
Evaluation	$O(n)$	$O(n)$	$O(n^2)$
Addition	$O(n)$	∞	$O(n)$
Multiplication	$O(n^{1.58})$	$O(n)$	$O(n)$

?

?

Faster Polynomial Multiplication?

Multiplication is fast when using the point-value representation

degree $2n-2$

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

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

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

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

↓ **Point-wise multiplication** $O(n)$ time

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

↓ **Interpolation** ?

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

Coefficients to Point-Value Representation

Given: Polynomial $p(x)$ by the coefficient vector $(a_0, a_1, \dots, a_{N-1})$

Goal: Compute $p(x)$ for all x in a given set X

- Where X is of size $|X| = N$
- Assume that N is a power of 2

Divide and Conquer Approach

- Divide $p(x)$ of degree $N - 1$ (N is even) into 2 polynomials of degree $N/_2 - 1$ differently than in Karatsuba's algorithm

- $p_0(y) = a_0 + a_2 y + a_4 y^2 + \cdots + a_{N-2} y^{N/_2 - 1}$ (even coeff.)
  $p_1(y) = a_1 + a_3 y + a_5 y^2 + \cdots + a_{N-1} y^{N/_2 - 1}$ (odd coeff.)

$p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots + a_{N-2} x^{N-2} + a_{N-1} x^{N-1}$

Coefficients to Point-Value Representation

Goal: Compute $p(x)$ for all x in a given set X of size $|X| = N$

- Divide $p(x)$ of degr. $N-1$ into 2 polynomials of degr. ${}^{N}/_{2}-1$

$$p_0(y) = a_0 + a_2 y + a_4 y^2 + \dots + a_{N-2} y^{N/2-1} \quad \text{(even coeff.)}$$
$$p_1(y) = a_1 + a_3 y + a_5 y^2 + \dots + a_{N-1} y^{N/2-1} \quad \text{(odd coeff.)}$$

Let's first look at the "combine" step: need $p(x)$ for all $x \in X$

$$\forall x \in X: \quad p(x) = p_0(x^2) + x \cdot p_1(x^2)$$

- Recursively compute $p_0(y)$ and $p_1(y)$ for all $y \in X^2$
  - Where $X^2 := \{x^2 : x \in X\}$

$X = \{2, 3, 7, 10\}, \quad X^2 = \{4, 9, 49, 100\}$

- Generally, we have $|X^2| = |X|$

$|X^2| < |X|$

Analysis

$|X^2| \leq |X|$ $\quad |X| \geq n$

Recurrence formula for the given algorithm:

$$T(n, |X|) = 2T(n/2, |X^2|) + O(n + |X|)$$

#coeff. #points

$$\leq 2T(n/2, |X|) + O(n + |X|)$$

at start:

$|X| = \Theta(n)$

$$T(n, n) = O(n^2)$$

Faster Algorithm?

- In order to have a faster algorithm, we need $|X^2| < |X|$

$n = |X|$ best we can hope for: $|X^2| = |X|/2$

$T(n, |X|) = 2T(n/2, |X^2|) + O(n + |X|)$

$= 2T(n/2, |X|/2) + O(n + |X|)$

$T'(n) = 2T'(n/2) + O(n)$

$\Rightarrow T(n,n) = T'(n) = O(n \log n)$

$T'(k) := T(k, k)$

$\{1\}$ $\quad X^2$

$\{-1, 1\}$ $\quad X$

$\{-i, i, -1, 1\}$

$\underbrace{\{-i, i, -1, 1, \frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}, \frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}}, -\frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}}, -\frac{1}{\sqrt{2}} - \frac{i}{\sqrt{2}}\}}_{\text{all solutions to } x^8 = 1}$

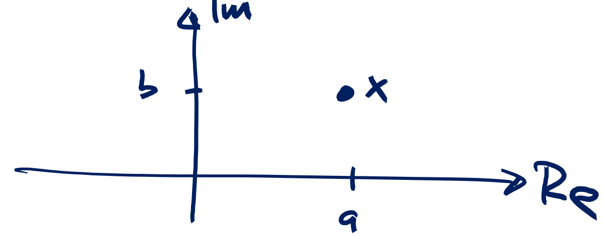
$x^N = 1$

Choice of X

complex number $x \in \mathbb{C}$

$x = a + i \cdot b$

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

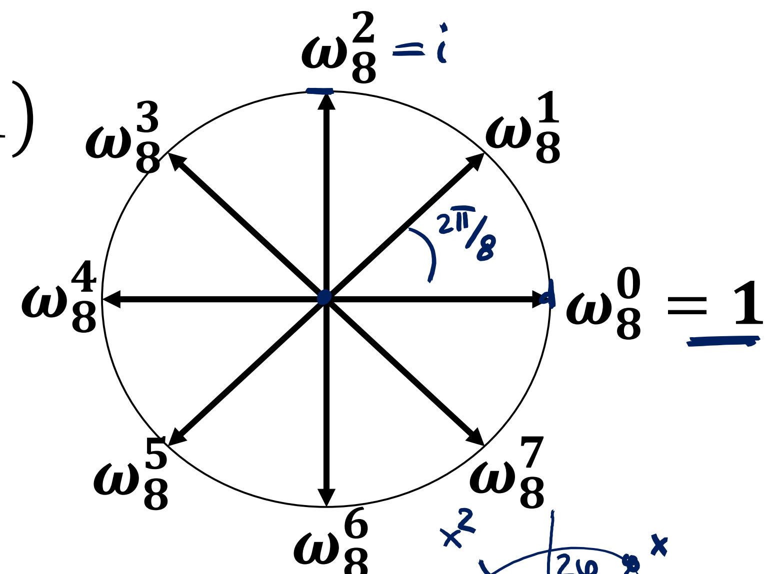


Consider the N complex roots of unity:

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

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

points on unit circle $\cos\varphi + i \cdot \sin\varphi = e^{2i\cdot\varphi}$

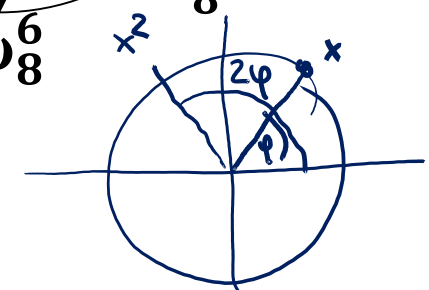


Powers of ω_n (roots of unity):

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

$x_k = e^{\frac{2\pi i}{N}\cdot k}$

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



Properties of the Roots of Unity

- **Cancellation Lemma:**

$x_k = \omega_n^k := e^{\frac{2\pi i}{n} \cdot k}$

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:**

$$\omega_n^k = e^{\frac{2\pi i}{n} \cdot k} = \underbrace{e^{\frac{2\pi i}{d \cdot n} \cdot d \cdot k}}_{\omega_{dn}^{dk}}$$

$$e^{\frac{2\pi i}{n}(k+n)} = \underbrace{e^{\frac{2\pi i}{n} \cdot k}}_{\omega_n^k} \cdot \underbrace{e^{2\pi i}}_{=1}$$

$\frac{2\pi}{n}$ $\qquad$ $\frac{2\pi}{d \cdot n}$

Properties of the Roots of Unity

$\omega_{2k} = e^{\frac{2\pi i}{2k}}$

Claim: If $X = \left\{\omega_{2k}^{i} : i \in \{0, \dots, 2k-1\}\right\}$, we have

$$\boldsymbol{X^2 = \left\{\omega_k^i : i \in \{0, \dots, k-1\}\right\}}, \qquad \boldsymbol{|X^2| = \frac{|X|}{2}}$$

$\omega_{2k}^{2i} = \omega_k^{i}$

Cancellation lemma

$|X| = 2k$

$|X^2| = k$

Analysis

New recurrence formula:

$$T(N, |X|) \leq 2 \cdot T\left({}^{N}\!/_{2}, {}^{|X|}\!/_{2}\right) + O(N + |X|)$$

for $|X| = N$

$O(N \log N)$