Chapter 1
Divide and Conquer

Algorithm Theory
WS 2019/20

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

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.

Recurrence Relations

Recurrence relation

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

Recurrence Relations: Master Theorem

Recurrence relation

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

Cases

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

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

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

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

- $f(n) = \Theta\left(n^c \cdot \log^k n\right),\ k \geq 0, c = \log_b a$

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

Polynomials

Real polynomial p in one variable x:

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

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)

Operations: Evaluation

- Given: Polynomial $p \in \mathbb{R}[x]$ of degree $n-1$

$$p(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0$$

Operations: Evaluation

- Given: Polynomial $p \in \mathbb{R}[x]$ of degree $n-1$

$$p(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0$$

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

$$p(x_0) = \big(\ldots\big((a_{n-1}x_0 + a_{n-2})x_0 + a_{n-3}\big)x_0 + \cdots + a_1\big)x_0 + a_0$$

- Pseudo-code:

$p \coloneqq a_{n-1};\ i \coloneqq n-1;$
while $(i > 0)$ **do**
  $i \coloneqq i - 1;$
  $p \coloneqq p \cdot x_0 + a_i$

- Running time: $O(n)$

Operations: Addition

- Given: Polynomials $p, q \in \mathbb{R}[x]$ of degree $n-1$

$$p(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0$$
$$q(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \cdots + b_1 x + b_0$$

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

$$\begin{aligned} p(x) + q(x) & \\ &= (a_{n-1}x^{n-1} + \cdots + a_0) + (b_{n-1}x^{n-1} + \cdots + b_0) \\ &= (a_{n-1} + b_{n-1})x^{n-1} + \cdots + (a_1 + b_1)x + (a_0 + b_0) \end{aligned}$$

Operations: Multiplication

- Given: Polynomials $p, q \in \mathbb{R}[x]$ of degree $n-1$

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

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

$$\begin{aligned} p(x) \cdot q(x) &= (a_{n-1}x^{n-1} + \cdots + a_0) \cdot (b_{n-1}x^{n-1} + \cdots + b_0) \\ &= c_{2n-2}x^{2n-2} + c_{2n-3}x^{2n-3} + \cdots + c_1 x + c_0 \end{aligned}$$

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

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

where $a_i = b_i = 0$ for $i \ge n$.

- Running time naïve algorithm:

Faster Multiplication?

- Multiplication is slow $\left(\Theta(n^2)\right)$

- 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…
  - follows immediately by using the master theorem

More Clever Recursive Solution

- **Recall that**

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

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

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. $\left(n/2 - 1\right)$-polynomials

$$T(n) = 3T\left(n/2\right) + O(n)$$

- Gives: $T(n) = O\left(n^{1.58496\ldots}\right)$ (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 $a_0, \dots, a_{n-1}$:

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

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

- 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 degr. $n-1$ is given by its $n-1$ roots

$$p(x) = a_{n-1} \cdot (x - x_1) \cdot (x - x_2) \cdot \ldots \cdot (x - x_{n-1})$$

- Example:

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

- Every polynomial has exactly $n-1$ roots $x_i \in \mathbb{C}$ (s.t. $p(x_i) = 0$)
  - Polynomial is uniquely defined by the $n-1$ roots and a_{n-1}

- We will not use this representation…

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 $0 \leq i, j \leq n$
- Time: $O(n^2)$

Operations: Linear Factors (Roots)

$$\boldsymbol{p(x) = a_{n-1} \cdot (x - x_1) \cdot \ldots \cdot (x - x_{n-1})}$$
$$\boldsymbol{q(x) = b_{n-1} \cdot (x - x_1) \cdot \ldots \cdot (x - x_{n-1})}$$

Evaluation:

- Just plug in the value where the poly. is evaluated: Time $O(n)$

Multiplication:

- Concatenate the two representations: Time $O(n)$

Addition:

- Need to find the roots of $p(x) + q(x)$
- For polynomials of degree > 4, this is not possible by using basic arithmetic operations ($+, -, \cdot, /, \sqrt[a]{b}$)
- In the usual computational model impossible
  – Numerically, the roots can be computed to arbitrary precision

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

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

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

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

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

↓ **Point-wise multiplication**

$2n$ 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.)

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. $N/_2 - 1$

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

Let's first look at the "combine" step:

- We need to compute $p(x)$ for all $x \in X$ after recursive calls for polynomials p_0 and p_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. $N/2 - 1$

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

Let's first look at the "combine" step:

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

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

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

Analysis

Recurrence formula for the given algorithm:

Faster Algorithm?

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

Choice of X

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

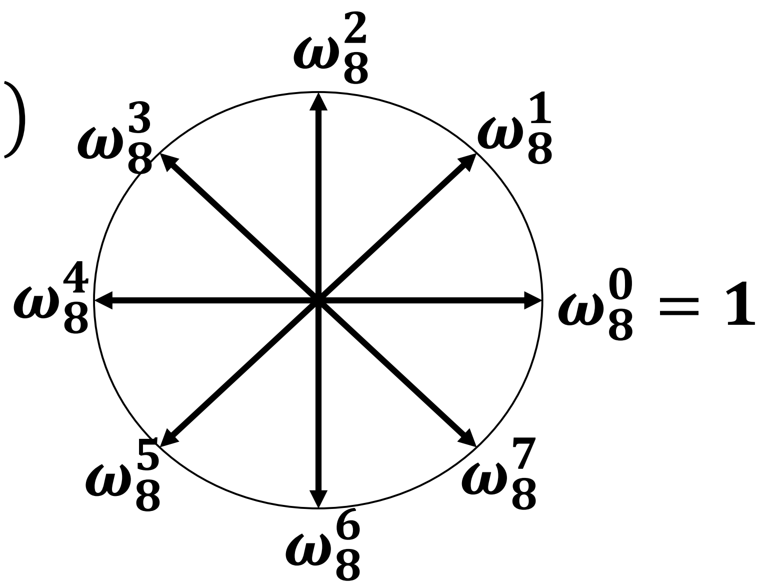
Consider the N complex roots of unity:

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

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

Powers of ω_N (roots of unity):

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



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

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:**

Properties of the Roots of Unity

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

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

Analysis

New recurrence formula:

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

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