Algorithm Theory

Chapter 1
Divide and Conquer

Part III:
Operations on Polynomials, Karatsuba Alg.

Fabian Kuhn

Polynomials

Real polynomial p in one variable x:

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

Coefficients of p: $a_0, a_1, \dots, a_{n-1} \in \mathbb{R}$

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 on Polynomials

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

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

- How expensive are basic operations on these polynomials?
  - **Evaluation:** What is $p(x_0)$ for a given value $x_0 \in \mathbb{R}$?
  - **Addition:** Compute the polynomial $p(x) + q(x)$
  - **Multiplication:** Compute the polynomial $p(x) \cdot q(x)$

We will focus on multiplication.

- **Computational Models**
  - **RAM (random access machine):** standard model for algorithm analysis
    - Reading / writing one memory cell costs 1 time unit
    - Basic arithmetic op. on integers cost 1 time unit (if integers fit in a mem. cell)
  - **Real RAM:**
    - Also basic arithmetic operations on real numbers cost 1 time unit
    - We will now use this assumption

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:

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

- Pseudo-code:

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

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

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

- Running time: $O(n)$

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_k: what products of monomials have degree i?

$$\text{For } 0 \leq k \leq 2n-2\text{:}\ c_k = \sum_{i=0}^{k} a_i b_{k-i}$$

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

- Running time naïve algorithm: $O(n^2)$

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

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

Algorithm:

- Compute (recursively): $p_0(x) \cdot q_0\,(x)$ $p_1(x) \cdot q_1\,(x)$

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

- $p(x)q(x) = \square \cdot x^n + \big(\underbrace{\square - \square - \square}_{\square}\big) \cdot x^{n/2} + \square$

Karatsuba Algorithm

- **Recursive multiplication:**

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

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

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

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

$= \log_2 3$

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