EXERCISE FOR CSE202 – WEEK 2

The starting point in the presentation of Karatsuba's algorithm in the course is that a polynomial of degree 2 can be reconstructed from its values at 3 points. The aim of this exercise is to analyze how generalizations of this idea lead to faster multiplication algorithms.

Observe first more generally that given $k+1$ points $(a_0, \ldots, a_k)$ and a polynomial $P = p_0 + \cdots + p_k x^k$, the values of P at these points are given by the product

$$\begin{pmatrix} P(a_0) \\ \vdots \\ P(a_k) \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & a_0 & \cdots & a_0^k \\ & \cdots & \cdots & \\ 1 & a_k & \cdots & a_k^k \end{pmatrix}}_{V(a_0,\ldots,a_k)} \begin{pmatrix} p_0 \\ \vdots \\ p_k \end{pmatrix}.$$

The matrix $V(a_0, \ldots, a_k)$ is known as a Vandermonde matrix and it is invertible when the a_i's are all distinct. In that case, multiplication by the inverse V^{-1} solves the interpolation problem: it recovers the coefficients of a polynomial from its values at $(a_0, \ldots, a_k)$.

We first clarify the use of this idea by designing a variant of Karatsuba's algorithm. With the three points $\{-1, 0, 1\}$, the inverse of the Vandermonde matrix is

$$V(-1,0,1)^{-1} = \frac{1}{2}\begin{pmatrix} 0 & 2 & 0 \\ -1 & 0 & 1 \\ 1 & -2 & 1 \end{pmatrix}.$$

This implies that for any polynomial P of degree at most 2,

$$P(T) = \frac{1}{2}\left(2P(0) + (P(1) - P(-1))T + (P(1) - 2P(0) + P(-1))T^2\right).$$

Question 1. *By proceeding as in the derivation of Karatsuba's algorithm in the lecture, design a recursive algorithm for polynomial multiplication that relies on this formula. Give its asymptotic complexity.*

Hint. *Split the two input polynomials F and G exactly as in Karatsuba, $F = F_0 + x^k F_1$ and $G = G_0 + x^k G_1$, and apply the interpolation approach presented above.*

Solution : We first establish a polynomial identity for the product of degree 1 polynomials. If $F = F_0 + TF_1$ and $G = G_0 + TG_1$, then the evaluations of FG at $0, 1, -1$ are given by

$$H_0 = F_0 G_0, \quad \tilde{H}_1 = (F_0 + F_1)(G_0 + G_1), \quad \tilde{H}_{-1} = (F_0 - F_1)(G_0 - G_1). \tag{1}$$

The formula above then gives

$$H = FG = \frac{1}{2}\left(2H_0 + T(\tilde{H}_1 - \tilde{H}_{-1}) + T^2(\tilde{H}_1 - 2\tilde{H}_0 + \tilde{H}_{-1})\right).$$

The algorithm obtained by evaluating this formula at $T = x^k$ is the following:
Input: two polynomials F and G of degree $< n$
Output: their product $H = FG$

(1) If $n = 1$ return FG
(2) Let $k := \lceil n/2 \rceil$
(3) Split $F = F_0 + x^k F_1, G = G_0 + x^k G_1$ with $\deg F_i, G_i < k$
(4) Compute recursively the three products from Eq. (**??**)
(5) Return $\frac{1}{2}\left(2H_0 + x^k(\tilde{H}_1 - \tilde{H} - 1) + x^{2k}(\tilde{H}_1 - 2\tilde{H}_0 + \tilde{H}_{-1})\right)$.

The complexity obeys the same recurrence as Karatsuba's algorithm:

$$C(n) \leq 3C(\lceil n/2 \rceil) + \lambda n.$$

Its complexity is therefore the same one as well, namely $C(n) = O(n^{\log_2 3})$. □

Next, we consider the five points $-2, -1, 0, 1, 2$. While the details are unimportant, it may fix the ideas to see the inverse

$$V(-2,-1,0,1,2)^{-1} = \frac{1}{24}\begin{pmatrix} 0 & 0 & 24 & 0 & 0 \\ 2 & -16 & 0 & 16 & -2 \\ -1 & 16 & -30 & 16 & -1 \\ -2 & 4 & 0 & -4 & 2 \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix}.$$

Question 2. *Proceeding as before, but now splitting the polynomials to be multiplied into three rather than two parts, give the outline of a recursive algorithm for polynomial multiplication. It is not necessary to write explicitly the coefficients used at each step, but the information should be sufficient for you to show that its complexity obeys the recurrence $C(n) \leq 5C(\lceil n/3 \rceil) + \lambda n$, from which you should deduce the asymptotic complexity of your algorithm and see that it improves upon Karatsuba's algorithm.*

Hint. *Split the two input polynomials F and G as $F = F_0 + x^k F_1 + x^{2k} F_2$ and $G = G_0 + x^k G_1 + x^{2k} G_2$, and apply the interpolation approach presented above, which now concerns a product of degree 2 polynomials.*

Solution : Again, we start by setting up a polynomial identity for the product of polynomials of degree 2. If $F = F_0 + F_1 T + F_2 T^2$ and $G = G_0 + G_1 T + G_2 T^2$ then the evaluations of FG at $-2, -1, 0, 1, 2$ are

$$\begin{cases} \tilde{H}_{-2} & := (F_0 - 2F_1 + 4F_2)(G_0 - 2G_1 + 4G_2); \\ \tilde{H}_{-1} & := (F_0 - F_1 + F_2)(G_0 - G_1 + G_2) \\ H_0 & := F_0 G_0; \\ \tilde{H}_1 & := (F_0 + F_1 + F_2)(G_0 + G_1 + G_2); \\ \tilde{H}_2 & := (F_0 + 2F_1 + 4F_2)(G_0 + 2G_1 + 4G_2). \end{cases} \tag{2}$$

The coefficients of $H = FG$ are recovered by multiplication by the matrix above.

The algorithm obtained by evaluating this identity at $T = X^k$ becomes:
Input: two polynomials F and G of degree $< n$
Output: their product $H = FG$

(1) If $n < 3$ use the naive algorithm
(2) Let $k := \lceil n/3 \rceil$
(3) Split $F = F_0 + x^k F_1 + x^{2k} F_2, G = G_0 + x^k G_1 + x^{2k} G_2$ with $\deg F_i, G_i < k$
(4) Compute recursively the five products from Eq. (**??**)

(5) Return the linear combinations $(h_0, \dots, h_4)$ of these values given by $V(-2,-1,0,1,2)^{-1}$ to obtain the coefficients of
$$H = h_0 + h_1 x^k + \cdots + h_4 x^{4k}$$
and return H.

This algorithm makes 5 recursive calls in degree at most $\lceil n/3 \rceil$, plus a number of operations that remains linear in n. It follows that its complexity $C(n)$ satisfies
$$C(n) \leq 5C(\lceil n/3 \rceil) + \lambda n.$$
Proceedings as in the lecture, this leads to a complexity estimate
$$C(n) = O(n^{\log_3 5}).$$
Since numerically $\log_3 5 \approx 1.465$, this is better than $\log_2 3 \approx 1.58$. □

Question 3. *Without entering into any detail, give the outline of a generalization of this method that would split each polynomial into m parts recursively, for a given m. Its special cases when $m = 2, 3$ should correspond to the algorithms of the previous questions. Indicate its complexity in the form $O(n^{f(m)})$ for some f to be determined and give the limit of $f(m)$ as $m \to \infty$. Conclude on the complexity of multiplication.*

Solution : The generalization looks as follows:

(1) If $n < m$ use the naive algorithm
(2) Let $k := \lceil n/m \rceil$
(3) Split F and G into m parts with each part of degree $< k$
(4) Compute recursively the $(2m-1)$ products of the linear combinations of these parts corresponding to evaluations at $(-m, -m+1, \dots, m-1, m)$.
(5) Recover the result by linear combinations whose coefficients are given by $V(-m, \dots, m)^{-1}$.

The complexity then obeys the recurrence
$$C(n) \leq (2m-1)C(\lceil n/m \rceil) + \lambda n,$$
whence, by the same reasoning
$$C(n) = O(n^{\log_m (2m-1)}).$$
As $m \to \infty$,
$$\log_m (2m-1) = \frac{\log(2m-1)}{\log m} \to 1.$$
It follows that asymptotically, multiplication can be performed in $O(n^{1+o(1)})$ operations, ie, almost as fast as addition. □