Assignment 1

CS 487/687 – CM 730
due Tuesday January 29, 5pm

January 15, 2019

Submission by email to cs487@student.cs.uwaterloo.ca

1. (10 marks) Show the steps of Karatsuba’s algorithm for input polynomials $F := 1 - x + x^2 + x^3$ and $G := 4 + 3x + 2x^2 + x^3$. You can implement the algorithm and submit your code (which should print enough information for us to understand what it does), or do it by hand.

2. (10 marks) Denote by $K(n)$ the number of operations in Karatsuba’s algorithm, for inputs of degree less than $n$, with $n$ a power of 2. A (very) close look at Karatsuba’s algorithm shows that we can take $K(1) = 1$ and

   $K(n) = 3K(n/2) + 7/2n - 3 \ (n > 1)$.

   We modify Karatsuba’s algorithm as follows: we fix a threshold $T$, and for $n \leq T$, we use the naive algorithm instead of entering recursive calls. Recall that we gave in class the number of operations for the naive algorithm.

   You have to find, any way you want, which value of $T$ minimizes the total number of operations for any $n = 2^k$, $k = 0, \ldots, 10$ (there is such a $T$). You can try an analytic solution (I did not) or write a little program that does the job for you. In this case, submit your code, and a short explanation of what it is doing.

3. (15 marks) Professor X plans to stun the scientific world by giving a linear time algorithm for multiplication of polynomials in $\mathbb{F}_2[x]$. Let’s see how that plays out.

   (a) First, let us work over $\mathbb{Q}$ instead. Suppose that we have an algorithm to compute squares in $\mathbb{Q}[x]$ in time $S(n)$ (for inputs of degree less than $n$). As a technicality, we assume that $S(n)$ is non-decreasing, and $S(n) \geq n$. Here, we count operations ($+, -, \times$) in $\mathbb{Q}$ at unit cost.

   Show that you can deduce from it an algorithm to do multiplication in $\mathbb{Q}[x]$ in time $O(S(n))$ for inputs of degree less than $n$. Hint: expand $(a + b)^2$, and another similar square.
Here is the genius idea behind Professor X’s plan: he claims that he has a linear time algorithm to compute squares of polynomials in $F_2[x]$ (here, we count operations in $F_2$ at unit cost).

Show that Professor X is right. Hint: compute $(1 + x)^2$ and (for example) $(1 + x + x^3)^2$.

Now, Professor X wants to apply the algorithm of question (a) over $F_2$. Will he succeed?

Reminder: $F_2 = \{0, 1\}$ — see the slides for the $+$ and $\times$ laws

4. (18 marks) In this problem, we study another form of FFT. Let $n$ be a positive integer, and assume that $n$ is a power of 2. Let $m := n/2$.

(a) We know that the roots of unity of order $n$ in $\mathbb{C}$ are the roots of $x^n - 1$. Show that they can be partitioned into the roots of $x^m - 1$ and of $x^m + 1$. Explicitly, what are the roots of these two polynomials?

(b) Suppose that $P$ is a polynomial in $\mathbb{C}[x]$ of degree less than $n$, with $n = 2m$. Show that you can compute $P_+ := P \mod (x^m - 1)$ and $P_- := P \mod (x^m + 1)$ in linear time (in $n$).

(c) Show that if $z$ is a root of $x^m - 1$, then $P(z) = P_+(z)$, and if $z$ is a root of $x^m + 1$, then $P(z) = P_-(z)$. Hint: use the Euclidean division equality $P = A_+ \cdot (x^m - 1) + P_+$ (and its analogue).

(d) Let $Q_-(x) := P_-(x/\omega)$, with $\omega = \exp(i\pi/m)$. Given $\omega$ and $P_-$, show how to compute the coefficients of $Q_-$ in linear time.

(e) Show that $z$ is a root of $x^m + 1$ if and only if $\omega z$ is a root of $x^m - 1$, and that in this case $P_-(z) = Q_-(\omega z)$.

(f) Put everything together to get another FFT algorithm of cost $O(n \log(n))$, for $n$ a power of 2.

5. (2 marks) Give a (simple) algorithm that multiplies a polynomial $F = f_0 + f_1 x$ by a polynomial $G = g_0 + \cdots + g_{n-1} x^{n-1}$ in $O(n)$ operations.

6. (15 marks) Consider $a_1, \ldots, a_n$ in $\mathbb{C}$ and define $M := \prod_{j=1}^n (x - a_j)$. To simplify matters, assume that we use FFT to multiply polynomials. If you want, you can assume that $n$ is a power of 2.

(a) Give a simple iterative algorithm that computes $M$ in time $O(n^2)$.

(b) Give a divide-and-conquer algorithm that computes $M$ in subquadratic time. What is the complexity?

7. (2 marks) How much time did you spend on the assignment?