CMPT 981/409 Algebra and Computation
Homework 1
Due: Friday, October 2, 2:30pm

Instructions: You may work on and discuss homework problems with one another. Each student must individually write and submit their own solutions on Crowdmark. Be sure to list any collaborations in which you participated and any external sources you used. The use of generative AI in any form is prohibited.

1. (Fast polynomial division with remainder.) Let $f, g \in \mathbb{C}[x]$ be polynomials of degree n and m, respectively. Let $q, r \in \mathbb{C}[x]$ be the quotient and remainder of f divided by g. In other words, the polynomials q and r satisfy $f(x) = q(x)g(x) + r(x)$ and $\deg(r) < \deg(g)$.

   In this exercise, we will develop a fast algorithm to compute the coefficients of q and r. We assume, without loss of generality, that $n \geq m$ and the polynomials f and g are *monic* (i.e., that $f_n = g_m = 1$).

   (a) For a polynomial $f(x) = f_n x^n + \cdots + f_1 x + f_0$, the *reversal* of f is the polynomial

   $$\operatorname{rev}(f)(x) \coloneqq x^n f\left(\frac{1}{x}\right) = f_0 x^n + f_1 x^{n-1} + \cdots + f_n.$$

   In words, $\operatorname{rev}(f)$ is obtained by reversing the order of the coefficients of f. Prove that

   $$\operatorname{rev}(f) = \operatorname{rev}(q)\operatorname{rev}(g) + \operatorname{rev}(r)\, x^k \qquad (\star)$$

   for some $k \geq n - m + 1$.

   (b) The constant term of the polynomial $\operatorname{rev}(g)$ is 1. Because its constant term is nonzero, $\operatorname{rev}(g)$ can be inverted as a formal power series[1] using the identity $\frac{1}{1-x} = 1+x+x^2+x^3+\cdots$. We will design a fast algorithm to compute the first n coefficients of $\operatorname{rev}(g)^{-1}$ using an iterative method called *Newton iteration*.

      i. Let $h_0(x) \coloneqq 1$ and for $i \geq 1$, let

      $$h_i(x) \coloneqq 2h_{i-1}(x) - \operatorname{rev}(g)(x)\, h_{i-1}(x)^2 \operatorname{rem} x^{2^i},$$

      where rem x^{2^i} indicates we drop terms of degree 2^i and higher. Prove that for all $i \geq 0$, there is a polynomial $p_i(x)$ such that

      $$h_i(x)\operatorname{rev}(g)(x) = 1 + p_i(x)\, x^{2^i}.$$

      In other words, the polynomial h_i agrees with the first 2^i coefficients of $\operatorname{rev}(g)^{-1}$.

      ii. Describe and analyze an algorithm to compute the coefficients of h_k for $k = \log_2(n)+1$. How many arithmetic operations does your algorithm use as a function of n?

   (c) Given the coefficients of $h_k(x)$ for $k = \log_2(n) + 1$, describe and analyze an algorithm that computes the coefficients of the quotient $q(x)$ in $O(n \log n)$ arithmetic operations.

   (Hint: multiply $(\star)$ by h_k.)

[1] A *formal power series* is a sum of the form $\sum_{i=0}^{\infty} a_i x^i$. Power series can be added and multiplied in the same way as polynomials. The adjective "formal" means that we ignore issues of convergence: a formal power series may not have a well-defined evaluation at any point other than zero.

(d) Given the coefficients of $q(x)$, describe and analyze an algorithm that computes the coefficients of the remainder $r(x)$ in $O(n \log n)$ arithmetic operations.

2. (Polynomial interpolation.) Given n pairs of complex numbers $(a_1, b_1), \ldots, (a_n, b_n)$ where the a_i are pairwise distinct, there is a unique polynomial $f \in \mathbb{C}[x]$ of degree less than n that satisfies $f(a_i) = b_i$ for all $1 \le i \le n$. The task of computing the coefficients of f from the (a_i, b_i) is called polynomial interpolation.

(a) For each $1 \le i \le n$, describe a polynomial $L_i(x)$ of degree at most $n-1$ that satisfies

$$L_i(a_j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

(b) Using the polynomials $L_1(x), \ldots, L_n(x)$, describe and analyze an algorithm for polynomial interpolation that uses $n^{O(1)}$ arithmetic operations.

3. (Fast evaluation and interpolation with derivatives.) Let $f(x) = f_n x^n + \cdots + f_1 x + f_0$ be a degree-n polynomial and let $a \in \mathbb{C}$. In this exercise, we will design fast algorithms to evaluate all derivatives of f at a and to recover the coefficients of f from the values of its derivatives. You may use, without proof, the fact that polynomial interpolation can be performed using $O(n \log^2 n)$ arithmetic operations (the "fast" version of problem 2).

(a) We can expand f using powers of $x - a$ instead of x. That is, there is a unique representation of f as

$$f(x) = \hat{f}_n \cdot (x-a)^n + \hat{f}_{n-1} \cdot (x-a)^{n-1} + \cdots + \hat{f}_1 \cdot (x-a) + \hat{f}_0.$$

Design and analyze an algorithm that takes $f_n, \ldots, f_1, f_0$ and a as input and computes $\hat{f}_n, \ldots, \hat{f}_1, \hat{f}_0$ using $O(n \log^2 n)$ arithmetic operations.

(b) Design and analyze an algorithm that takes $f_n, \ldots, f_1, f_0$ and a as input and computes the value of f and its derivatives $f', f'', \ldots, f^{(n)}$ at a using $O(n \log^2 n)$ arithmetic operations.

(Hint: first solve the case $a = 0$.)

(c) Design and analyze an algorithm that takes a and $f(a), f'(a), \ldots, f^{(n)}(a)$ as input and computes the coefficients of f using $O(n \log^2 n)$ arithmetic operations.

4. (Sturm's theorem.) Let $f \in \mathbb{R}[x]$ be a polynomial with no multiple roots, or equivalently, a polynomial such that $\gcd(f, f') = 1$. The fundamental theorem of algebra says that f has exactly $\deg(f)$ complex roots. In this exercise, we will develop an algorithm to compute the number of *real* roots of f.

Consider the following modification of the Euclidean algorithm. Set $f_0 \coloneqq f$ and $f_1 \coloneqq f'$. For each $i \ge 2$, let q_i and r_i be the quotient and remainder, respectively, of f_{i-2} divided by f_{i-1}. We will set $f_i \coloneqq -r_i$, so

$$f_{i-2}(x) = q_i(x) f_{i-1}(x) - f_i(x).$$

Let $f_0, f_1, \ldots, f_\ell$ be the resulting sequence of polynomials, called the *Sturm sequence* of f.

(a) For every $1 \le i \le \ell$, prove that f_{i-1} and f_i have no common real roots.

(b) For a real number $c \in \mathbb{R}$, let $v(c)$ be the the number of sign alternations in the sequence $f_0(c), \ldots, f_\ell(c)$. A *sign alternation* occurs at index i if either $f_i(c) < 0$ and $f_{i+1}(c) \geq 0$, or $f_i(c) > 0$ and $f_{i+1}(c) \leq 0$. We will analyze the behaviour of $v(c)$ on a real interval (a, b).

i. Suppose the interval (a, b) contains exactly one root of one of the polynomials $f_1, f_2, \ldots, f_\ell$ and no roots of f_0. Show that the value of $v(c)$ is the same for all $c \in (a, b)$.

ii. Suppose the interval (a, b) contains one root of f_0 and no roots of $f_1, \ldots, f_\ell$. Show that the value of $v(c)$ decreases by one at the root of f_0.

(c) Prove that the number of real roots of f in the interval (a, b) is $v(a) - v(b)$.

(d) Conclude that there is an algorithm that takes as input real numbers $a, b \in \mathbb{R}$ and a degree-n polynomial $f \in \mathbb{R}[x]$ with no multiple roots and computes the number of real roots of f in the interval (a, b) using $n^{O(1)}$ arithmetic operations.

(Not for submission: can you relax the assumption that f has no multiple roots? What about counting all real roots of f?)