

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. (Modified rational interpolation.) Given n pairs of complex numbers $(a_1, b_1), \dots, (a_n, b_n)$ where the a_i are pairwise distinct and a parameter $0 \leq d \leq n$, the *modified rational interpolation*¹ problem asks us to find polynomials $f, g \in \mathbb{C}[x]$ such that $\deg(f) < d$, $\deg(g) \leq n - d$, and $f(a_i) = b_i g(a_i)$ for all $1 \leq i \leq n$. In this exercise, we will design an algorithm to solve this problem.

Let $h \in \mathbb{C}[x]$ be the polynomial of degree less than n that satisfies $h(a_i) = b_i$ for all $1 \leq i \leq n$. In other words, h solves the polynomial interpolation problem for the data $(a_1, b_1), \dots, (a_n, b_n)$.

- (a) Suppose $f, g \in \mathbb{C}[x]$ solve the modified rational interpolation problem. Prove that $f(x) - h(x)g(x)$ is divisible by $x - a_i$ for all $1 \leq i \leq n$.
 - (b) As a consequence of part (a), the polynomial $\prod_{i=1}^n (x - a_i)$ divides $f(x) - h(x)g(x)$. Prove that the extended Euclidean scheme of $h(x)$ and $\prod_{i=1}^n (x - a_i)$ contains polynomials $f, g \in \mathbb{C}[x]$ that solve the modified rational interpolation problem.
 - (c) Design and analyze an algorithm that solves the modified rational interpolation problem.
2. (Resultants and algebraic numbers.) A complex number $a \in \mathbb{C}$ is *algebraic* if there is a polynomial $f \in \mathbb{Q}[x]$ such that $f(a) = 0$. For example, $\sqrt{2}$ is algebraic since it is a root of $x^2 - 2$. On the other hand, both $\pi \approx 3.14$ and $e \approx 2.718$ are not algebraic. The algebraic numbers form a *field*: if $a, b \in \mathbb{C}$ are algebraic, then so are $a + b$, ab , and $1/a$ (assuming $a \neq 0$).

We may compute with algebraic numbers by representing $a \in \mathbb{C}$ by a polynomial $f \in \mathbb{Q}[x]$ such that $f(a) = 0$. To add algebraic numbers $a, b \in \mathbb{C}$, we are given polynomials $f, g \in \mathbb{Q}[x]$ such that $f(a) = g(b) = 0$ and we need to find a polynomial $h \in \mathbb{Q}[x]$ such that $h(a + b) = 0$. We will solve this problem using the resultant.

Let $f, g \in \mathbb{Q}[x]$ be polynomials of positive degree.

- (a) Prove that $c \in \mathbb{C}$ can be written as $c = a + b$ with $f(a) = g(b) = 0$ if and only if the equations $f(x) = g(y - x) = 0$ have a solution with $y = c$.
 - (b) Let $h(y) := \text{res}_x(f(x), g(y - x))$. That is, we regard both $f(x)$ and $g(y - x)$ as polynomials in x whose coefficients are themselves polynomials in y , and take their resultant. Prove that $c \in \mathbb{C}$ is a root of h if and only if $c = a + b$ where $f(a) = g(b) = 0$.
- (Not for submission: can you adapt this construction to find a polynomial $h \in \mathbb{Q}[x]$ whose roots are of the form ab where $f(a) = g(b) = 0$?)

¹If $g(a_i) \neq 0$ for all i , then $f(a_i)/g(a_i) = b_i$, so $f(x)/g(x)$ is a rational function that interpolates the data $(a_i, b_i)_{i=1}^n$. A solution to rational interpolation may not exist, but the modified version of the problem always has a solution.

3. (Division with remainder in parallel.) Recall that for polynomials $f, g \in \mathbb{C}[x]$, there are unique polynomials $q, r \in \mathbb{C}[x]$ that satisfy $f(x) = q(x)g(x) + r(x)$ and $\deg(r) < \deg(g)$. Design and analyze a parallel algorithm that takes the coefficients of f and g as input and computes the coefficients of q and r . Your algorithm should run in $O(\log^2 d)$ parallel time, where $d = \max(\deg(f), \deg(g))$.
4. (Faster parallel Girard–Newton.) Let $f \in \mathbb{C}[x]$ be a degree- n polynomial that factors as $f(x) = \prod_{i=1}^n (x - a_i)$. In class, we saw an algorithm that computes the power sums $a_1^d + \dots + a_n^d$ from the coefficients of f in $O(\log^2 n)$ parallel time. In this exercise, we will develop a faster algorithm that solves this problem in $O(\log n)$ parallel time.

For $d \in \mathbb{N}$, let $p_d := a_1^d + \dots + a_n^d$ be the sum of d^{th} powers of the roots of f . By convention, $p_0 = n$.

- (a) Let $g(x) := \text{rev}(f)(x) = x^n f(1/x)$ be the reversal of f . Prove that $g(x) = \prod_{i=1}^n (1 - a_i x)$.
- (b) Prove that

$$\frac{g'(x)}{g(x)} = - \sum_{i=0}^{\infty} p_{i+1} x^i$$

as formal power series.²

(Hint: $\frac{1}{1-x} = 1 + x + x^2 + x^3 + \dots$ as power series.)

- (c) Using the identity from part (b), design and analyze an algorithm that takes the coefficients of f as input and computes the power sums $p_1, \dots, p_n$. Your algorithm should run in $O(\log n)$ parallel time.

(Hint: Approximate $\frac{1}{1-x}$ as $1 + x + \dots + x^m$. How large does m have to be?)

- (d) The function $\frac{g'(x)}{g(x)}$ is called the *logarithmic derivative* of $g(x)$ because $\frac{g'(x)}{g(x)} = \frac{d}{dx}(\log g(x))$. Taking antiderivatives and exponentiating gives us the formal power series identity

$$g(x) = \exp \left(\sum_{i=1}^{\infty} \frac{p_{i-1}}{i} x^i \right)$$

where $\exp(x) := \sum_{i=0}^{\infty} \frac{x^i}{i!}$ is the exponential power series. Using this identity, design and analyze an algorithm that takes $p_1, \dots, p_n$ as input and computes the coefficients of f . Your algorithm should run in $O(\log n)$ parallel time.

(Hint: Approximate $\exp(x)$ as $\sum_{i=0}^m \frac{x^i}{i!}$. How large does m have to be?)

²Recall that 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.