Problem 1

Continued fractions & Extended Euclidean Algorithm

Let $\mathbb{K}$ be a field, and $f_1, \ldots, f_\ell \in \mathbb{K}$. Then

$$f_1 + \cfrac{1}{f_2 + \cfrac{1}{\cdots \cfrac{}{f_{\ell-1} + \cfrac{1}{f_\ell}}}}$$

is the *continued fraction*, denoted by $C(f_1, \ldots, f_\ell)$. Now assume R is a Euclidean Domain and $\mathbb{K}$ its field of fractions. For $(r_0, r_1) \in R^2$, let $q_i \in R$, for $1 \leq i \leq \ell$, be the quotients in the extended Euclidean algorithm.

1. Show that
$$\frac{r_0}{r_1} = C(q_1, \ldots, q_l).$$

2. A convenient way to represent the continued fraction expansion is as a list $[q_1, q_2, \ldots, q_l]$. Write a Macaulay 2 procedure to compute the continued fraction expansion of two polynomials in $\mathbb{Q}[x]$. Run your algorithm on $r_0 = x^{20}$ and $r_1 = x^{19} + 2x^{18} + x \in \mathbb{Q}[x]$.

Problem 2

Binary GCD Algorithm

Consider the following algorithm to compute the GCD of two positive integers.

Algorithm:

`Input:` $a, b \in \mathbb{Z}_{>0}$

`Output:` $\gcd(a, b) \in \mathbb{Z}_{>0}$

1. if $a = b$ then return a;
2. if both a and b are even then return $2\gcd(a/2, b/2)$;
3. if exactly one number is even, say a, then return $\gcd(a/2, b)$;
4. if both a and b are odd, with, say $a > b$, then return $\gcd((a-b)/2, b)$;

1. Implement the above algorithm in Macaulay 2 (call it `binarygcd`) and show it works on the pairs $(34, 21)$, $(136, 51)$, $(481, 325)$, $(8771, 3206)$.

2. Prove the algorithm above works correctly. Use induction (you figure out what to base the induction on).

3. Find a good upper bound on the recursion depth, and use this to prove a running time of $O(\ell^2)$ bit operations on inputs of size ℓ (that is, $\lg a, \lg b \leq \ell$).

4. Modify the algorithm so that it additionally computes $s, t \in \mathbb{Z}$ such that $sa + tb = \gcd(a, b)$. Give your answer in the form of a Macaulay 2 function called `ebinarygcd` and test it on the pairs from part (1).

Problem 3

Polynomial Evaluation

Suppose you are given as input a polynomial $f \in R[y]$ of degree n, together with a matrix $A \in R[x]^{n\times n}$ filled with polynomials bounded in degree by $d > 0$.

1. Assuming the naive cost model, derive the cost of computing $f(A)$ using Horner's scheme. *Note:* You are counting ring operations from R, and your cost estimates should be in terms of the input parameters n and d.

2. Assuming the naive cost model, derive the cost of computing $f(A)$ using the baby-steps/giant-steps approach of Patterson and Stockmeyer.

3. Now assume Karatsuba is used for the polynomial multiplication, and derive the cost of computing $f(A)$ using the baby-steps/giant-steps approach.

Problem 4

Karatsuba's algorithm

Let R be a ring (commutative, with 1) and $f, g \in R[x, y]$ (polynomials in the two variables x and y). Assume that f and g have degrees less than m in y and n in x. Let $h = f \cdot g$ be the product of f and g.

1. Viewing f and g as polynomials in x with coefficients from $R[y]$, bound the cost of operations in R to compute h assuming the classical school method for univariate polynomial multiplication.

2. Now bound the number of operations from R to compute h when Karatsuba's algorithm is used.

Problem 5

Fast Fourier Transform

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 $m := n/2$.

1. 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?

2. 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).

3. 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).

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

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

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

Problem 6

Fast computation of elementary symmetric polynomials.

Consider the elementary symmetric polynomial of degree d in n variables.

$$E_d(x_1, \ldots, x_n) = \sum_{\substack{S \subset [n] \\ |S| = d}} \prod_{i \in S} x_i$$

Prove that for any pair (n, d) where $n \geq d$ the elementary symmetric polynomial can be computed by a depth-3 circuit of size $\mathsf{poly}(n, d)$. That is, the elementary symmetric polynomials can also be computed really fast in the parallel model.

1. Consider the polynomial
   $$p(x_1, \ldots, x_n, t) = \prod_{i=1}^{n} (t + x_i)$$
   as a polynomial in $\mathbb{C}[x_1, \ldots, x_n][t]$. For $0 \leq d \leq n$, what is the coefficient of monomial t^d in p?

2. Show how to obtain the elementary symmetric polynomial $E_d(x_1, \ldots, x_n)$ via interpolation.

3. Conclude by expressing $E_d(x_1, \ldots, x_n)$ as a $\mathsf{poly}(n, d)$-sized, depth 3 algebraic circuit.