CS 5200 Homework - 2

Instructor Avah Banerjee

Mar 5, 2024, 9 AM CST

Problem 1. (60 pts) In a distant alien civilization on a planet named Zahlenraumreich, a unique system of numbers—let's call them *außerirdischer* numbers—is used for complex calculations and spatial navigation. These außerirdischer numbers, comprise four components and obey specific multiplication rules. An außerirdischer number X can be represented as $X = a + b\flat + c\sharp + d\natural = (a, b, c, d)$ (written compactly), where a, b, c, and d are real numbers, and $\flat, \sharp, \natural$ are the fundamental außerirdischer units with the following properties: The multiplication rules for the fundamental außerirdischer units $\flat$, $\sharp$, and $\natural$ are summarized in the following table:

Expression	**Result**
$\flat^2$	-1
$\sharp^2$	-1
$\natural^2$	-1
$\flat\sharp\natural$	-1
$\flat\sharp$	$\natural$
$\sharp\flat$	$-\natural$
$\sharp\natural$	$\flat$
$\natural\sharp$	$-\flat$
$\natural\flat$	$\sharp$
$\flat\natural$	$-\sharp$

1. Given two außerirdischer numbers $X = (a, b, c, d)$ and $Y = (e, f, g, h)$, determine their product in terms of their components using the above multiplication table.

2. Show how to multiply two außerirdischer numbers X and Y using two 2×2 complex matrices, leveraging the matrix representation of complex numbers and the properties of außerirdischer numbers.

3. Devise a rule that can multiply two außerirdischer numbers with less than 13 real multiplications, optimizing the computational process based on the algebraic properties of außerirdischer numbers.

4. Provide some evidence (does not have to a formal proof) that at least 7 real multiplications are necessary to multiply two außerirdischer numbers, considering the constraints of their algebraic structure and the efficiency of the multiplication process.

Problem 2. (40 pts) In this assignment, you will be implementing the Fast Fourier Transform (FFT) algorithm in Python, performing computations in modular arithmetic.

1. Implement a function to find a primitive root τ modulo a given prime p where all powers of τ are distinct modulo p (that is $\tau^i \not\equiv \tau^j (\mod p)$ when $i \neq j$ and $1 \leq i, j \leq p-1$). Write a Python function `find_primitive_root(p)` that returns a primitive root modulo p, where p is a prime number. The function should also verify that the sum of the powers of τ from τ to τ^{p-1} is equal to 0 modulo p. What is the running time (basic arithmetic operations over constant number bits are assumed to take constant time) of `find_primitive_root(p)` if p is b bits long ?

2. Write a Python function to perform the FFT using the matrix method (replace the n^{th} root of unity with primitive root τ) modulo p. Specifically, implement the FFT in a function `fft(sequence, omega, p)` where `sequence` is the input array of coefficients, τ is the primitive root found earlier, and p is the prime modulo. Perform the appropriate matrix multiplication (mod p) to obtain the transformed sequence modulo p.