Number Theoretic Transform (NTT)

Introduction

The discrete Fourier transform (DFT) of size n and its most important O(n log(n)) Cooley-Tukey fast Fourier transform (FFT) algorithms are defined over the complex numbers and based on a primitive (i.e., exact) n-th root of unity that in class (last lectures) we write as ω_n. Both DFT and FFTs can be ported to finite fields Z/pZ (integers modulo a prime p), by replacing ω_n by a primitive n-th root of unity in Z/pZ, which exists iff n divides p-1. The DFT is then called number-theoretic transform (NTT), see [1,2] for a good brief introduction.

The goal is to implement and optimize the NTT over Z/pZ, where for each NTT size n, p can be suitably chosen.

As a baseline start with the simple iterative Cooley-Tukey type NTT algorithm in [4], which you can then also use for testing.

The optimized version should be recursive, e.g., see [3] and [5] for radix 2 and 4 FFTs that can be easily ported to NTT as starting point. Choose some of the FFT optimizations shown in class [6,7], all of which are in principle applicable (of course when ported to arithmetic modulo p), and anything else you can think of based on what you learned in class.

References

[1]https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=fd609d722157c0b09d675d245853d9230eb221aa , Chapter 4.
[2] https://www.nayuki.io/page/number-theoretic-transform-integer-dft
[3] https://en.wikipedia.org/wiki/Cooley–Tukey_FFT_algorithm#Pseudocode
[4] https://eprint.iacr.org/2017/727.pdf, Algorithm 1.
[5] https://users.ece.cmu.edu/~franzf/papers/gttse07.pdf, Section 6
[6] https://acl.inf.ethz.ch/teaching/fastcode/2023/slides/12-transforms.pdf
[7] https://acl.inf.ethz.ch/teaching/fastcode/2023/slides/13-fftw.pdf