IAM511 Algorithms And Complexity **Fall 2021-2022**
Homework 4 **Due: 02.02.2022**

Implementation

In this homework you will implement polynomial addition, subtraction and multiplication. For polynomial multiplication, you will use a combination of Scholbook and Karatsuba algorithms. You will also implement NTT, inverse NTT and pointwise multiplication. You will compare the running times of these multiplications.

The polynomials will be elements of $\mathbb{F}_q[x]/\langle x^8 - 1\rangle$ where $q = 1009$. Then the largest polynomial you'll use is of the form $a_0 + a_1x + \cdots + a_7x^7$ so, 8-coefficients. To be able to implement 2-way recursive multiplication algorithms for 8-coefficient polynomials, you'll need to implement 4 and 2-coefficient polynomial multiplications as well. You can represent polynomials as `uint32_t` arrays with 8 elements.

Follow the steps below for implementation.

S1. Addition/Subtraction

Implement polynomial addition and subtraction. Note that you'll need to use addition and subtraction within the multiplication algorithms as well, so parametrizing the element count might be helpful.

S2. 2-Coefficient Multiplication

Implement schoolbook multiplication for

$$\begin{aligned} ab = c &= (a_0 + a_1x)(b_0 + b_1x) \\ &= c_0 + c_1x + c_2x^2. \end{aligned}$$

S3. 4-Coefficient Multiplication

Implement a recursive schoolbook multiplication for

$$\begin{aligned} ab = c &= (a_0 + a_1x + a_2x^2 + a_3x^3)(b_0 + b_1x + b_2x^2 + b_3x^3) \\ &= c_0 + c_1x + \cdots + c_6x^6. \end{aligned}$$

The 2-coefficient multiplications you'll call recursively should be the multiplications that you have implemented in step 2.

S4. 8-Coefficient SB Multiplication in $\mathbb{F}_q[x]/\langle x^8 - 1\rangle$

Implement a recursive schoolbook multiplication for

$$\begin{aligned} ab = c &= (a_0 + a_1x + \cdots + a_7x^7)(b_0 + b_1x + \cdots + b_7x^7) \\ &= c_0 + c_1x + \cdots + c_7x^7 \end{aligned}$$

Denote this multiplication as $\mathbf{SB}(a, b)$. The 4-coefficient multiplications you'll call recursively should be the multiplications that you have implemented in step 3. Note that $a, b, c \in \mathbb{F}_q[x]/\langle x^8 - 1\rangle$, so you should reduce the 15-coefficient result back to 8-coefficients.

S5. 8-Coefficient KA Multiplication in $\mathbb{F}_q[x]/\langle x^8 - 1\rangle$

Implement a recursive Karatsuba multiplication for

$$\begin{aligned} ab = c &= (a_0 + a_1x + \cdots + a_7x^7)(b_0 + b_1x + \cdots + b_7x^7) \\ &= c_0 + c_1x + \cdots + c_7x^7 \end{aligned}$$

Denote this multiplication as $\mathbf{KA}(a, b)$ The 4-coefficient multiplications you'll call recursively should be the multiplications that you have implemented in step 3. Note that $a, b, c \in \mathbb{F}_q[x]/\langle x^8 - 1\rangle$, so you should reduce the 15-coefficient result back to 8-coefficients.

S6. NTT

Implement Number Theoretic Transform. Denote this transformation as $\mathbf{NTT}(a)$ where $a \in \mathbb{F}_q[x]/\langle x^8 - 1\rangle$. You don't need to implement a generic NTT that works for any input, you should just implement an NTT algorithm that works for our case, that is multiplying $a, b \in \mathbb{F}_q[x]/\langle x^8 - 1\rangle$. Use 192 as a primitive 8^{th} root of unity over $\mathbb{F}_q$.

S7. iNTT

Implement the inverse NTT. Denote this transformation as $\mathbf{iNTT}(\hat{a})$ where $\hat{a} = \mathbf{NTT}(a), \quad a \in \mathbb{F}_q[x]/\langle x^8 - 1\rangle$. Similar to step 6, don't implement a generic iNTT.

S8. Pointwise Multiplication

Implement a pointwise multiplication where every element of an array is multiplied with the corresponding element from the other array. Denote this multiplication as $\mathbf{PW}(a, b)$ where a, b are results of $\mathbf{NTT}$.

S9. Correctness

Show in code, that the 3 different 8-coefficient multiplications you implemented works correctly:

$$\mathbf{SB}(a, b) = \mathbf{KA}(a, b) = \mathbf{iNTT}(\mathbf{PW}(\mathbf{NTT}(a), \mathbf{NTT}(b))).$$

Print the input and output polynomials. Print the NTT results as well.

Comparison

Time and compare the 3 different 8-coefficient multiplication algorithms you have implemented as they are in step 9. Do your timings fit with the operation counts?

Test Case

You can test your programs with these values.

$$\begin{aligned}
a &= 5 + 12x + 43x^2 + 21x^3 + 132x^4 + 344x^5 + 512x^6 + 246x^7 \\
b &= 604 + 13x + 85x^2 + 0x^3 + 311x^4 + 312x^5 + 932x^6 + 813x^7 \\
ab &= 1002 + 250x + 319x^2 + 137x^3 + 455x^4 + 643x^5 + 630x^6 + 723x^7 \\
&\quad + 876x^8 + 620x^9 + 30x^{10} + 252x^{11} + 174x^{12} + 777x^{13} + 216x^{14} \\
ab \pmod{x^8 - 1} &= 869 + 870x + 349x^2 + 389x^3 + 629x^4 + 411x^5 + 846x^6 + 723x^7 \\
\mathbf{NTT}(a) &= [306, 784, 219, 336, 69, 978, 963, 421] \\
\mathbf{NTT}(b) &= [43, 115, 736, 82, 794, 874, 69, 101]
\end{aligned}$$

Guidelines and Implementation Considerations

- You can randomly generate the polynomials yourself, or you can use polynomials that are hardcoded into the program.
- You don't have to print the polynomials with x's, only the coefficients would be enough. Just make sure to write which side of the printed result (and the array) is the least significant side.
- Check how modulo (%) operator works in C. It may work differently than how you think it does.
- There is a very easy way of reducing polynomials mod $x^n - 1$.
- Time the algorithms, not the whole program.
- Your code must be C or C++.
- Write comments in the code if necessary.
- Send your small report as a pdf, prepared in LaTeX. You can use any of the `.tex` templates available in odtuclass.
- Upload your homework to odtuclass.
- For your codes, only send the `.c/.cpp` and the header files. **PLEASE** don't send the project / solution files, or the executable.

- Include a note about your operating system (win 10, linux distribution etc.) and your IDE (visual studio, devc++, codeblocks...) or your compiler.
- Do not steal your code. You can study other code and give references to them. Copying code and just changing the variable names is not the purpose of this homework.
- This is not a group homework. You can study with others, but don't copy each others code.