МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Дискретное преобразование Фурье

ОТЧЁТ

ПО ДИСЦИПЛИНЕ

«ТЕОРЕТИКО-ЧИСЛОВЫЕ МЕТОДЫ В КРИПТОГРАФИИ»

студента 5 курса 531 группы

специальности 10.05.01 Компьютерная безопасность

факультета компьютерных наук и информационных технологий

Сенокосова Владислава Владимировича

Преподаватель, профессор __________ В.А. Молчанов

подпись, дата

Саратов 2024

Содержание

1 Цель работы и порядок выполнения 3
2 Прямое и обратное дискретное преобразование Фурье 4
3 Быстрое прямое и обратное преобразование Фурье 6
4 Вычисления произведения многочленов с помощью быстрого преобразования Фурье 9
5 Произведение целых чисел (алгоритм Шенхаге-Штрассена) 12
6 Тестирование реализованных алгоритмов 14
ЗАКЛЮЧЕНИЕ 16
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 17
ПРИЛОЖЕНИЕ А 18

1 Цель работы и порядок выполнения

Цель работы — изучение свойств дискретного преобразования Фурье и программная реализация его приложений.

Порядок выполнения работы

1. Разобрать определения дискретного преобразования Фурье, обратного дискретного преобразования Фурье и алгоритмы этих преобразований Фурье. Привести программную реализацию быстрого преобразования Фурье и обратного быстрого преобразования Фурье.

2. Рассмотреть приложения дискретного преобразования Фурье и привести программную реализацию алгоритма Шенхаге-Штрассена для умножения целых чисел.

2 Прямое и обратное дискретное преобразование Фурье

Дискретное преобразование Фурье имеет важные приложения в теории чисел и алгебре. Дискретные преобразования Фурье помогают решать дифференциальные уравнения в частных производных и выполнять такие операции, как свёртки. Дискретные преобразования Фурье также активно используются в статистике, при анализе временных рядов. Существуют многомерные дискретные преобразования Фурье. Дискретное преобразование Фурье требует в качестве входа дискретную функцию.

Такие функции часто создаются путём дискретизации (выборки значений из непрерывных функций).

Определение: Дискретное преобразование Фурье (DFT) — это математический алгоритм, который преобразует последовательность дискретных чисел (сигналов) из временной области в частотную область. DFT берет набор N точек (дискретных значений) и вычисляет их спектральное представление, то есть как комбинацию синусоидальных волн различных частот.

Формула для дискретного преобразования Фурье для входной последовательности $x_0, x_1, \dots., x_{N-1}$ задается как:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi ikn}{N}} = \sum_{n=0}^{N-1} x_n (\cos\left(\frac{2\pi kn}{N}\right) - isin(\frac{2\pi kn}{N})), k = 0,1, \dots, N-1$$

$N -$ количество точек (длина последовательности \ выборки),

$X_k -$ выходные коэффициенты преобразования (частотное \ спектральное представление),

x_n – входная последовательность,

$e^{-\frac{2\pi ikn}{N}} -$ комплексные экспоненты, представляющие синусоидальные компоненты.

Обратное дискретное преобразование Фурье (IDFT) позволяет восстановить исходную последовательность x_n по спектральным коэффициентам X_k с помощью формулы:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{2\pi ikn}{N}} = \frac{1}{N}\sum_{k=0}^{N-1} X_k(\cos\left(\frac{2\pi kn}{N}\right) + isin(\frac{2\pi kn}{N})), n = 0, 1, .., N-1$$

DFT используется для анализа периодичности, частотных составляющих сигналов и других задач в вычислительной математике и инженерии.

Алгоритм вычисления строится на применении заданных формул к входным последовательностям x_n или X_k. Временная сложность для вычисления прямого и обратного преобразования Фурье составляет $O(N^2)$.

3 Быстрое прямое и обратное преобразование Фурье

Определение: Пусть R – коммутативное кольцо с единицей. Тогда элемент ω кольца R называется примитивным корнем степени n из единицы, если выполняются свойства:

1. $\omega \neq 1$
2. $\omega^n = 1$
3. $\sum_{j=0}^{n-1} \omega^{ij} = 0, \qquad 1 \leq i < n$

Если, кроме того, элемент n является обратимым элементом кольца R, то можно определить дискретное преобразование Фурье как отображение, которое каждому вектору $a = (a_0, a_1, \dots, a_{n-1}), a_i \in R, 0 \leq i \leq n-1$ ставит в соответствие вектор $F(a) = b = (b_0, b_1, \dots, b_{n-1}),$ где

$$b_i = \sum_{j=0}^{n-1} a_i \omega^{ij}, 0 \leq i \leq n-1$$

Обратное дискретное преобразование Фурье определяется как $F^{-1}(b) = c$, где координаты вектора c равны

$$c_i = \frac{1}{n} \sum_{j=0}^{n-1} b_i \omega^{-ij},\ 0 \leq i \leq n-1$$

То, что прямое и обратное дискретное преобразование Фурье взаимно обратны (т.е $F^{-1}\big(F(a)\big) = a,\ F(F^{-1}(b)) = b$), легко следует из определения примитивного корня.

Заметим, что если вектору a поставить в соответствие многочлен $f(x) = \sum_{i=0}^{n-1} a_i x^i$, то прямое дискретное преобразование Фурье соответствует вычислению значений многочлена в точках $\omega^i, 0 \leq i \leq n-1,$ а обратное – интерполяции многочлена по его значениям в этих точках. То, что данные точки образуют циклическую мультипликативную подгруппу в R, позволяет построить быстрый алгоритм вычисления значения как прямого, так и обратного дискретного преобразования Фурье.

Определение: Пусть $a = [a_0, \dots, a_{n-1}], b = [b_0, \dots, b_{n-1}], c = a \otimes b$. Сверткой векторов a и b называется вектор c размера $2n$ (обозначается $c = a \otimes b$) с компонентами $c_i = \sum_{j+k=i} a_j b_k$, $i = 0, 1, \dots, 2n-1$.

Содержательный смысл понятия свертки: если векторам $a, b,\ c = a \otimes b$ сопоставить многочлены $a(x), b(x), c(x)$ соответственно, то $c(x) = a(x)b(x)$. С учетом этого и того факта, что для задания многочлена степени $2n-1$ необходимо задать его значения в $2n$ точках, становится очевидной следующая теорема.

Теорема (о свертке): Пусть $a = [a_0, \dots, a_{n-1}],\ b = [b_0, \dots, b_{n-1}]$. Тогда $a \otimes b = F^{-1}\big(F(a') * F(b')\big)$, где $*$ - почленное произведение компонент векторов a', b' - векторы длины $2n$ полученные из a и b соответственно дополнением их нулями:

$$a' = [a_0, \dots, a_{n-1}, 0, \dots, 0], b' = [b_0, \dots, b_{n-1}, 0, \dots, 0]$$

Определение: Пусть $a = [a_0, \dots, a_{n-1}],\ b = [b_0, \dots, b_{n-1}], c = a \otimes b$. Положительно обернутой сверткой векторов a и b называется вектор d размера n (обозначается $d = a[+]b$) с компонентами $d_i = c_i + c_{i+n}$, $i = 0, \dots, n-1$. Отрицательной обернутой сверткой векторов a и b называется вектор e размера n (обозначается $e = a[-]b$) с компонентам $e_i = c_i + c_{i+n}, i = 0, \dots, n-1$.

Формулы для компонент положительно и отрицательно обернутых сверток можно записать по-другому, выразив их значения через компоненты векторов a и b:

$$d_i = \sum_{j=0}^{i} a_j b_{i-j} + \sum_{j=i+1}^{n-1} a_j b_{n+i-j},\ \ e_i = \sum_{j=0}^{i} a_j b_{i-j} - \sum_{j=i+1}^{n-1} a_j b_{n+i-j},$$

Теорема (о положительно обернутой свертке):

Пусть $a = [a_0, \dots, a_{n-1}],\ b = [b_0, \dots, b_{n-1}]$. Тогда $a\,[+]\,b = F^{-1}(F(a)F(b))$.

Теорема (об отрицательной обернутой свертке):

Пусть w – примитивный корень $n-$ й степени из 1 в кольце K, γ – элемент K такой, что $\gamma^n = -1, e = a[-]\, b -$ отрицательно обернутая свертка векторов $a = [a_0, \dots, a_{n-1}]$ и $b = [b_0, \dots, b_{n-1}]$,

$$a' = [a_0\, \gamma * a_1 \dots \gamma^{n-1} * a_{n-1}]^T,$$

$$b' = [b_0\, \gamma * b_1 \dots \gamma^{n-1} * b_{n-1}]^T$$

$$e' = [e_0\, \gamma * e_1 \dots \gamma^{n-1} * e_{n-1}]^T$$

Тогда $e' = F^{-1}(F(a')F(b'))$.

Алгоритм быстрого преобразования Фурье:

Вход: К – коммутативное кольцо с 1, $n = 2^k$, $a = [a_0, \dots, a_{n-1}] -$ вектор с элементами из $K, w -$ примитивный корень степени n из 1 в кольце K.

Выход: $b = F(a)$

Будем использовать вспомогательные массивы S и R длины n и обозначать их i-е элементы $S[i], R[i]$ соответственно.

1. Для $i = 0, \dots., n-1$ положить $R[i] = a_i$
2. Для $l = k-1, \dots, 0$ выполнять:

   2.1 Положить $S[i] = R[i]$ для $i = 0, \dots, n-1$

   2.2 Для $i = 0, \dots., n-1$:

   Пусть $i = (i_{k-1}, \dots, i_0)_2$; положить

$$R[i] = S[(i_{k-1}, \dots, i_{l+1} 0\, i_{l-1}, \dots, i_0)_2] + w^{rev\left(\frac{i}{2^l}\right)} * S[(i_{k-1}, \dots, i_{l+1} 1\, i_{l-1}, \dots, i_0)_2]$$

3. Для $i = 0, \dots, n-1$ положить $b_i = R[rev(i)]$

Для вычисления ОДПФ применяется этот же алгоритм, но вместо w используется w^{-1} и в шаге 3 полагаем $b_i = n^{-1} * R[rev(i)]$.

Из алгоритма БПФ и теорем о свертках вытекает

Утверждение: ДПФ, ОДПФ, свертку, положительно и отрицательно обернутые свертки можно вычислить за $O(nlog_2 n)$ шагов.

4 Вычисления произведения многочленов с помощью быстрого преобразования Фурье

Благодаря алгоритму быстрого преобразования Фурье дискретное преобразование Фурье является очень удобным инструментом при проведении вычислений с многочленами. Так, например, с его помощью можно вычислять произведение многочленов со сложностью $0(nlogn)$ (выполнив сначала преобразование Фурье и получив значения многочленов в точках, затем перемножив полученные значения и, наконец, вернуться к коэффициентам многочлена с помощью обратного преобразования Фурье). Однако, оно обладает тем недостатком, что для существования преобразования Фурье над кольцом к требуется выполнение двух условий: существования примитивного корня степени и и обратимости числа и в кольце R, а эти условия выполняются далеко не всегда. Кроме того, как следует из алгоритма быстрого преобразования Фурье, желательно, чтобы число и было некоторой степенью числа 2.

Вычисление произведения многочленов с помощью быстрого преобразования Фурье (БПФ) основано на представлении многочленов в частотной области, где их произведение становится проще. Этот метод позволяет свести сложность вычисления произведения двух многочленов из квадратичной $O(n^2)$ к $O(nlogn)$, используя свойства преобразования Фурье и его обратного преобразования.

Пусть имеется два многочлена:

$$P(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

$$Q(x) = b_0 + b_1x + b_2x^2 + \cdots + b_nx^m$$

Прямая формула для произведения многочленов имеет вид

$$(\sum_{i=0}^{n} a_i x^i) * (\sum_{j=0}^{m} b_j x^j) = \sum_{k=0}^{n+m} x^k \sum_{i+j=k} a_i b_j$$

Сложность при использовании такой формулы составляет $O(n^2)$. Чтобы ускорить умножение двух полиномов, с помощью интерполяции.

Теорема. Пусть есть набор различных точек $x_0, x_1, \dots, x_n$. Многочлен степени n однозначно задаётся своими значениями в этих точках. Часто для интерполяции пользуются методом Гаусса.

Основная идея алгоритма: если мы знаем значения в каких-то различных $n + m$ точках для обоих многочленов $P(x)$ и $Q(x)$, то, попарно перемножив их, мы за $O(n + m)$ операций можем получить значения в тех же точках для многочлена $P(x) * Q(x)$ – а их помощью можно интерполяцией получить исходный многочлен и решить задачу. Однако в данном случае алгоритм будет иметь временную сложность $O(n^3)$ как минимум из-за метода Гаусса.

Для того чтобы ускориться можно рассмотреть основное свойство комплексных чисел, которое нам понадобится для умножения полиномов.

Утверждение: Для любого натурального n есть ровно n комплексных «корней из единицы», то есть чисел w_k, для которых выполнено:

$$w_k^n = 1$$

А именно, это будут числа вида:

$$w_k = e^{\frac{i2\pi k}{n}}$$

Таким образом мы можем преобразовать коэффициенты двух многочленов, в частотную форму, с помощью рассмотренных выше алгоритмов преобразования Фурье.

Алгоритм умножения двух полиномов может быть сформулирован следующим образом.

1. **Определение длины результирующего многочлена**: Входные многочлены A и B имеют размеры n и m. Результат умножения этих многочленов будет многочленом степени $n + m - 1$ Чтобы упростить использование быстрого преобразования Фурье, длину результирующего многочлена нужно округлить до ближайшей степени двойки, так как алгоритм FFT требует длины, равной степени двойки. При необходимости дополнить многочлен до длины n.

2. **Преобразование многочленов в частотную область (FFT)**: Мы применяем быстрое преобразование Фурье (FFT) к каждому из многочленов. Это преобразует коэффициенты многочлена из временной области (коэффициенты при степенях x) в частотную область.

3. **Поэлементное умножение преобразованных коэффициентов**: В частотной области произведение двух многочленов сводится к поэлементному умножению соответствующих коэффициентов преобразованных списков, полученных на шаге 2.

4. **Обратное преобразование Фурье (IFFT)**: Чтобы вернуть результат произведения многочленов в исходную временную область, применяется обратное быстрое преобразование Фурье (IDFT). Это восстанавливает коэффициенты результирующего многочлена.

5. **Округление действительных частей результата**: Так как результат обратного преобразования может содержать небольшие числовые погрешности в виде мнимых частей, из-за численных методов, применённых в преобразованиях Фурье, мнимая часть игнорируется, а действительные части коэффициентов округляются до ближайшего целого числа.

Сложность полученного алгоритма составляет: $O(nlogn)$

5 Произведение целых чисел (алгоритм Шенхаге-Штрассена)

Метод умножения Шенхаге — Штрассена — алгоритм умножения больших целых чисел, основанный на быстром преобразовании Фурье.

Фактически является методом умножения многочленов от одной переменной, превращается в алгоритм умножения чисел, если эти числа представить как многочлены от основы системы счисления, а после получения результата сделать переносы через разряды.

Алгоритм имеет следующие шаги:

Вход: Два числа $u, v - N$ – разрядные двоичные числа, $N = 2^n$

Выход: $y = uv \; mod \; (2^N + 1)$

1. Положить $l = \left[\frac{n}{2}\right], k = n - l, K = 2^k, L = 2^l$ и разбить числа u и v на K групп по L разрядов: $u = (U_{K-1}, \dots, U_0)_{2^L}, v = (V_{K-1}, \dots, V_0)_{2^L}$.
   Далее ищем вектор $W = u[-]v = (W_{k-1}, \dots, W_0)_{2^L}$
2. Для $i = 0, \dots, K-1$ найти
$$\mathrm{W_i'} = W_i \; mod \; K = \left(\sum_{j=0}^{i} U_{i-j}V_j - \sum_{j=i+1}^{K-1} U_{i+K-j}V_j\right) mod \; K.$$
3. Найти $W_i^{"} = W_i \; mod \; (2^{2L} + 1), i = 0, \dots, K-1$:
   a. Положить $\gamma = 2^{2LK}$ и построить векторы
   $$u' = [U_0 \; \gamma U_1 \dots . \gamma^{K-1} U_{K-1}]^T, v' = [V_0 \; \gamma V_1 \dots . \gamma^{K-1} V_{K-1}]^T$$
   b. Положить $m = 2^{2L} + 1, w = 2^{4LK}$ и найти ДПФ $u'' = F(u'), v'' = F(v')$ в Z_m.
   c. Вычислить покомпонентное произведение векторов $c = u''v''$
   d. Вычислить ОДПФ $d = F^{-1}(c)$, положив $w^{-1} = -2^{2L-4LK}$ и $K^{-1} = -2^{2L-k}$
   e. Из векторов $d = [W_0 \; \gamma W_1 \dots . \gamma^{K-1} W_{K-1}]^T$, с учетом $\gamma^{-1} = -2^{2L-2lK}$ найти все W_i".
4. Для $i = 0, \dots, K-1$ вычислить
   $W_i''' = (2^{2L} + 1)\big((W_i' - W_i'') \; mod \; K\big) + W_i''$ и

$$W_i = \begin{cases} W_i''', & \text{если } W_i''' < (i+1)2^{2L} \\ W_i''' - K(2^{2L}+1) & \text{иначе} \end{cases}$$

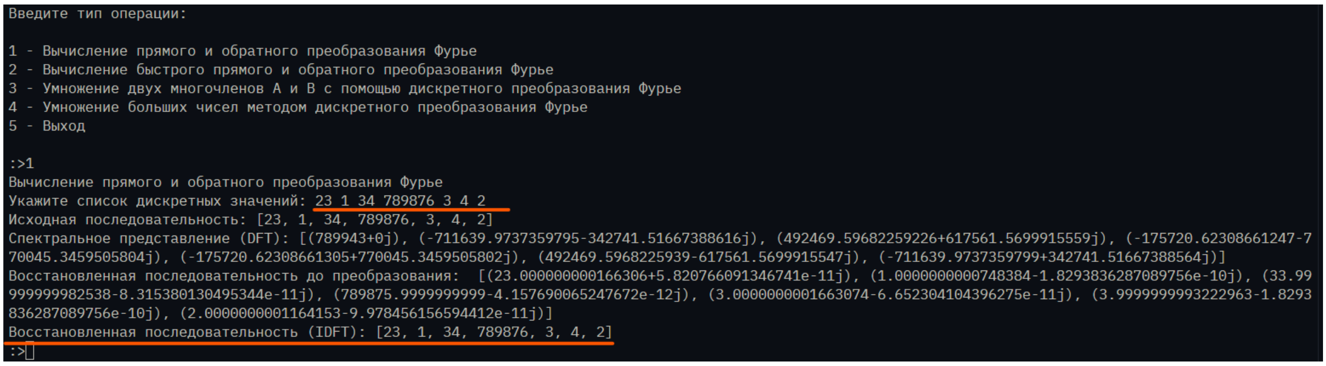
5. $y = \sum_{i=0}^{2K-1} W_i 2^L$

Сложность данного алгоритма составляет $O(n\,log(n)\,log(log(n)))$, где n - количество двоичных цифр в произведении

6 Тестирование реализованных алгоритмов

Программная реализация алгоритмов представлена на языке Python. Дальнейшие вычисления и результаты работы представлены на рисунках.

Рассмотрим работу алгоритма для вычисления прямого и обратного дискретного преобразования Фурье. Тестирование будет производиться на множестве дискретных значений $[23, 1, 34, 789876, 3, 4, 2]$.



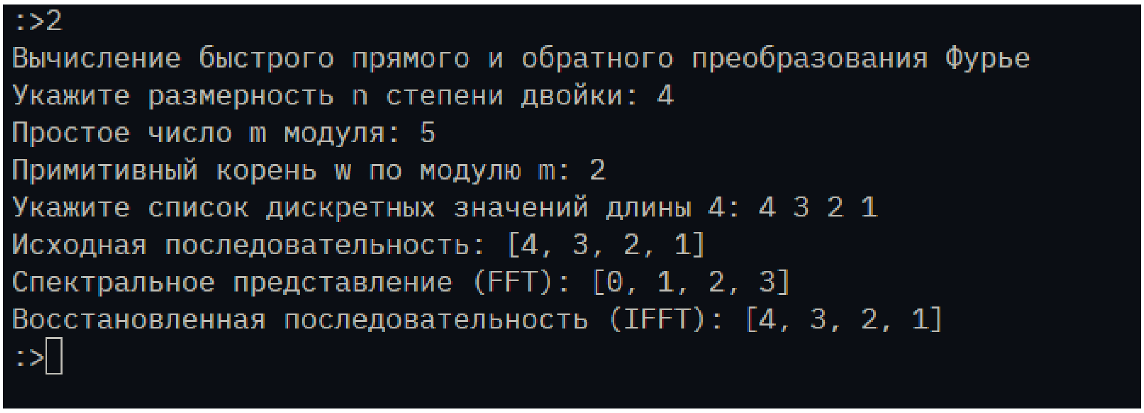
```
Введите тип операции:

1 - Вычисление прямого и обратного преобразования Фурье
2 - Вычисление быстрого прямого и обратного преобразования Фурье
3 - Умножение двух многочленов A и B с помощью дискретного преобразования Фурье
4 - Умножение больших чисел методом дискретного преобразования Фурье
5 - Выход

:>1
Вычисление прямого и обратного преобразования Фурье
Укажите список дискретных значений: 23 1 34 789876 3 4 2
Исходная последовательность: [23, 1, 34, 789876, 3, 4, 2]
Спектральное представление (DFT): [(789943+0j), (-711639.9737359795-342741.51667388616j), (492469.59682259226+617561.5699915559j), (-175720.62308661247-770045.3459505804j), (-175720.62308661305+770045.3459505802j), (492469.5968225939-617561.5699915547j), (-711639.9737359799+342741.51667388564j)]
Восстановленная последовательность до преобразования:  [(23.000000000166306+5.820766091346741e-11j), (1.0000000000748384-1.8293836287089756e-10j), (33.999999999982538-8.315380130495344e-11j), (789875.9999999999-4.157690065247672e-12j), (3.0000000001663074-6.652304104396275e-11j), (3.9999999993222963-1.8293836287089756e-10j), (2.0000000001164153-9.978456156594412e-11j)]
Восстановленная последовательность (IDFT): [23, 1, 34, 789876, 3, 4, 2]
:>
```

Рисунок 1 – Вычисление прямого и обратного преобразования Фурье на дискретных значениях: $[23, 1, 34, 789876, 3, 4, 2]$

Для проверки работоспособности протестируем быстрые алгоритмы прямого и обратного преобразования Фурье. В данном случае вычисления должны происходить в кольце. Пускай будет следующие входные данные: $n = 4, w = 2, \ m = w^{\frac{n}{2}} + 1 = 5, a = [4\ 3\ 2\ 1]$. Результаты тестирования представлены на рисунке.



```
:>2
Вычисление быстрого прямого и обратного преобразования Фурье
Укажите размерность n степени двойки: 4
Простое число m модуля: 5
Примитивный корень w по модулю m: 2
Укажите список дискретных значений длины 4: 4 3 2 1
Исходная последовательность: [4, 3, 2, 1]
Спектральное представление (FFT): [0, 1, 2, 3]
Восстановленная последовательность (IFFT): [4, 3, 2, 1]
:>
```

Рисунок 2 - Вычисление быстрого прямого и обратного преобразования Фурье при заданных значениях: $n = 4, w = 2, m = w^{\frac{n}{2}} + 1 = 5, a = [4\ 3\ 2\ 1]$.

Найдем с помощью алгоритма быстрого преобразования Фурье произведение двух многочленов: $a = 234x^6 + 973x^5 + 9x^3 + 17x^2 + 81x +$

100, $b = 8x^2 + 7861x - 23$ в кольце Z_{100}. В результате произведения получаем многочлен $a * b = 1872x^8 + 1847258x^7 + 7643371x^6 - 22307x^5 + 70885x^4 + 134078x^3 + 637150x^2 + 784237x - 2300$, и в кольце Z_{100} который представлен на рисунке в виде списка коэффициентов при неизвестных. -

```
:>3
Укажите модуль m:100
Умножение двух многочленов A и B с помощью дискретного преобразования Фурье
Укажите многочлены A и B (сначала старшие степени)
A: 234 973 0 9 17 81 100
B: 8 7861 -23
[100, 81, 17, 9, 0, 973, 234] [-23, 7861, 8]
[0, 0, 0, 0, 0, 0, 0, 1872, 1847258, 7643371, -22307, 70885, 134078, 637150, 784237, -2300]
[72, 58, 71, 93, 85, 78, 50, 37, 0]
:>
```

Рисунок 3 – Вычисление произведения многочленов a и b в кольце Z_100

Вычислим произведение следующих чисел: 20000 и 20, 3278642643478528438 и 78658797234, 7 и 7, 123456789 и 987654321. Полученные результаты сравним со встроенной функцией умножения чисел в Python. Как можно заметить из рисунка результаты тестирования корректны.

```
Введите тип операции:

1 - Вычисление прямого и обратного преобразования Фурье
2 - Вычисление быстрого прямого и обратного преобразования Фурье
3 - Умножение двух многочленов A и B с помощью дискретного преобразования Фурье
4 - Умножение больших чисел методом дискретного преобразования Фурье
5 - Выход

:>4
Умножение больших чисел методом дискретного преобразования Фурье
Введите число x: 20000
Введите число y: 20
Результат умножения x и y: 400000
Проверка со встроенным умножением: 400000
:>4
Умножение больших чисел методом дискретного преобразования Фурье
Введите число x: 3278642643478528438
Введите число y: 78658797234
Результат умножения x и y: 257894086896123320837344740492
Проверка со встроенным умножением: 257894086896123320837344740492
:>4
Умножение больших чисел методом дискретного преобразования Фурье
Введите число x: 7
Введите число y: 7
Результат умножения x и y: 49
Проверка со встроенным умножением: 49
:>4
Умножение больших чисел методом дискретного преобразования Фурье
Введите число x: 123456789
Введите число y: 987654321
Результат умножения x и y: 121932631112635269
Проверка со встроенным умножением: 121932631112635269
:>
```

Рисунок 4 – Вычисления произведения заданных чисел

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены и реализованы алгоритмы, связанные с дискретным преобразованием Фурье (DFT) и его быстрым вариантом (FFT). Эти методы доказали свою эффективность в различных приложениях, включая умножение больших чисел с помощью алгоритма Шенхаге-Штрассена и умножения произвольных многочленов. Реализация программного кода позволила на практике убедиться в правильности и производительности данных алгоритмов. Быстрое преобразование Фурье продемонстрировало свою полезность в решении задач, требующих работы с сигналами и большими числами, что подтверждает его важность в вычислительной математике и теории чисел.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Глухов М. М. и др. Введение в теоретико-числовые методы криптографии: учеб. пособие - Москва : Лань, 2011.

2. Маховенко Е.Б. Теоретико-числовые методы в криптографии. М.: Гелиос АРВ, 2006.

3. Черемушкин, А. В. Лекции по арифметическим алгоритмам в криптографии. - Москва : МЦНМО, 2002.

4. Панкратова И.А. Теоретико-числовые методы в криптографии. Томск, 2009.

5. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии. М.:МЦНМО, 2003.

6. Венбо Мао. Современная криптография: теория и практика. М.:Вильямс, 2005.

ПРИЛОЖЕНИЕ А

Реализованные программы для лабораторной работы

```
import math
import cmath

# Прямое дискретное преобразование Фурье (DFT)
def DFT(lst_vals):
    N = len(lst_vals)
    result = []
    for k in range(N):
        X_k = 0
        for n in range(N):
            pow_ = -2 * math.pi * k * n / N
            X_k += lst_vals[n] * cmath.exp(1j * pow_)
        result.append(X_k)
    return result

# Обратное дискретное преобразование Фурье (IDFT)
def IDFT(lst_spectr):
    N = len(lst_spectr)
    result = []
    for n in range(N):
        x_n = 0
        for k in range(N):
            pow_ = 2 * math.pi * k * n / N
            x_n += lst_spectr[k] * cmath.exp(1j * pow_)
        result.append(x_n / N)
    return result

# Быстрое обратное дискретное преобразование
def bit_reverse(i, bits):
    """Возвращает индекс с побитовой реверсией"""
    reverse = 0
    for _ in range(bits):
        reverse = (reverse << 1) | (i & 1)
        i >>= 1
    return reverse

def mod_exp(base, exp, mod):
    """Возведение в степень с использованием модуля,
       модульная экспонентация"""
    result = 1
    while exp > 0:
        if exp % 2 == 1:
            result = (result * base) % mod
        base = (base * base) % mod
        exp //= 2
    return result
```

```python
def FFT(a, omega, p):
    """Прямое быстрое преобразование Фурье в кольце вычетов"""
    n = len(a)
    k = n.bit_length() - 1
    R = a[:]

    for l in range(k - 1, -1, -1):
        S = R[:]
        for i in range(n):
            bit_shift = (i >> l) & 1
            idx_0 = (i & (~(1 << l)))
            idx_1 = idx_0 | (1 << l)
            rev_index = bit_reverse(i // (2**l), k)
            omega_power = mod_exp(omega, rev_index, p)
            R[i] = (S[idx_0] + omega_power * S[idx_1]) % p

    b = [R[bit_reverse(i, k)] for i in range(n)]
    return b

def IFFT(b, w_inv, n_inv, p):
    """Обратное быстрое преобразование Фурье в кольце вычетов"""
    n = len(b)
    k = n.bit_length() - 1
    R = b[:]

    for l in range(k - 1, -1, -1):
        S = R[:]
        for i in range(n):
            bit_shift = (i >> l) & 1
            idx_0 = (i & (~(1 << l)))
            idx_1 = idx_0 | (1 << l)
            rev_index = bit_reverse(i // (2**l), k)
            omega_inv_power = mod_exp(w_inv, rev_index, p)
            R[i] = (S[idx_0] + omega_inv_power * S[idx_1]) % p

    c = [(n_inv * R[bit_reverse(i, k)]) % p for i in range(n)]
    return c

# Проверяем на длину массива
def power_of_two(a):
    n = len(a)
    power_ = 1
    while power_ < n:
        power_ *= 2
    return a + [0] * (power_ - n)

# Основной алгоритм для произведения многочленов с использованием DFT и
IDFT
def mul_polinom(A, B):
    n = 1
    while n < len(A) + len(B):
```

```
        n *= 2
    A += [0] * (n - len(A))
    B += [0] * (n - len(B))
    FA = DFT(A)
    FB = DFT(B)
    FC = [FA[i] * FB[i] for i in range(n)]
    C = IDFT(FC)
    return [round(c.real) for c in C]

# Алгоритм Шенхаге-Штрассена для умножения больших чисел
def mul(x, y):

    a = [int(digit) for digit in str(x)][::-1]
    b = [int(digit) for digit in str(y)][::-1]

    product = mul_polinom(a, b)

    carry = 0
    result = []
    for coeff in product:
        total = coeff + carry
        result.append(total % 10)
        carry = total // 10

    while len(result) > 1 and result[-1] == 0:
        result.pop()

    return int(''.join(map(str, result[::-1])))

if __name__ == "__main__":
    type_ = """Введите тип операции: \n
1 - Вычисление прямого и обратного преобразования Фурье
2 - Вычисление быстрого прямого и обратного преобразования Фурье
3 - Умножение двух многочленов A и B с помощью дискретного
    преобразования Фурье
4 - Умножение больших чисел методом дискретного преобразования Фурье
5 - Выход\n"""
    print(type_)
    param = None
    while param not in ["1", "2", "3", "4"]:
        param = input(":>")
        match param:
            case "1":
                print("Вычисление прямого и обратного
                      преобразования Фурье")
                lst = list(map(lambda x: int(x), input("Укажите
                           список дискретных значений: ").split()))
                spectrum = DFT(lst)
                recovered_lst = IDFT(spectrum)
                print("Исходная последовательность:", lst)
                print("Спектральное представление (DFT):", spectrum)
```

```
            print("Восстановленная последовательность
                  до преобразования: ", recovered_lst)
            print("Восстановленная последовательность (IDFT):",
                   [round(x.real) for x in recovered_lst])
            param = None
        case "2":
            print("Вычисление быстрого прямого и обратного
                    преобразования Фурье")
            n = int(input("Укажите размерность n степени двойки: "))
            m = int(input("Простое число m модуля: "))
            w = int(input("Примитивный корень w по модулю m: "))
            w_inv = mod_exp(w, m - 2, m)
            n_inv = mod_exp(n, m - 2, m)
            lst = list(map(lambda x: int(x), input(f"Укажите список
                       дискретных значений длины {n}: ").split()))
            len_ = len(lst)
            new_lst = power_of_two(lst)
            spectrum = FFT(new_lst, w, m)
            recovered_lst = IFFT(spectrum, w_inv, n_inv, m)
            print("Исходная последовательность:", lst)
            print("Спектральное представление (FFT):",
                  spectrum[:len_])
            print("Восстановленная последовательность (IFFT):",
                   [round(x.real) for x in recovered_lst[:len_]])
            param = None
        case "3":
            m = int(input("Укажите модуль m:"))
            print("Умножение двух многочленов A и B с помощью
                  дискретного преобразования Фурье")
            print("Укажите многочлены A и B (сначала старшие
                  степени)")
            A = list(map(lambda x: int(x), input("A:").split()))[::-1]
            B = list(map(lambda x: int(x), input("B:").split()))[::-1]
            print(A, B)
            res = mul_polinom(A, B)[::-1]
            print(res)
            for j in range(len(res)):
                res[j] = res[j] % m
            for i in range(len(res)):
                if res[i] != 0:
                    break
            print(res[i:])
            param = None
        case "4":
            print("Умножение больших чисел методом дискретного
                  преобразования Фурье")
            x = int(input("Введите число x: "))
            y = int(input("Введите число y: "))
            product = mul(x, y)
            print("Результат умножения x и y:", product)
            print(f"Проверка со встроенным умножением: {x * y}")
```

```
            param = None
        case "5":
            param = "4"
```