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

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:

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

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

$\mathrm{X}_k -$ выходные коэффициенты преобразования (частотное \ спектральное представление),

x_n – входная последовательность,

$e^{-\frac{2\pi i k n}{N}} -$ 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

Преимущество быстрого преобразования Фурье (FFT\БПФ) перед обычным дискретным преобразованием Фурье (DFT\ДПФ) заключается в значительном уменьшении времени вычислений. Данный алгоритма имеет сложность $O(NlogN)$ благодаря рекурсивному делению последовательности на более маленькие части. Это делает алгоритм значительно быстрее на больших входных данных.

Наиболее распространённым алгоритмом быстрого преобразования Фурье является алгоритм Кули — Тьюки, при котором дискретное преобразование Фурье от $N = N_1 N_2$ выражается как сумма дискретных преобразований Фурье более малых размерностей N_1 и N_2 рекурсивно для того, чтобы достичь сложность $O(Nlog(N))$. В вычислительной технике наиболее часто используется рекурсивное разложение преобразований надвое, то есть с основанием 2 (хотя может быть использовано любое основание), а количество входных отсчётов является степенью двойки. Для случаев, когда дискретное преобразование считается от количества отсчётов, являющихся простыми числами, могут быть использованы алгоритмы Блуштайна и Рейдера.

Например, для вычисления быстрого преобразования Фурье по алгоритму Кули — Тьюки с основанием 2 для вектора $\vec{x}$, состоящего из N элементов:

$$\vec{X} = A\vec{x}$$

где A имеют вид:

$$a_N^{mn} = e^{-\frac{2\pi imn}{N}}$$

дискретное преобразование можно выразить как сумму двух частей: сумму чётных индексов $m = 2n$ и сумму нечетных индексов $m = 2n + 1$:

$$X_m = \sum_{n=0}^{N-1} x_n a_N^{nm} = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} a_N^{2nm} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n} a_N^{(2n+1)m}$$

Коэффициенты a_N^{2nm} и $a_N^{(2n+1)m}$ можно переписать следующим образом:

$$a_N^{2nm} = e^{-\frac{2\pi i(2nm)}{N}} = e^{-\frac{2\pi i(mn)}{\frac{N}{2}}} = a_{\frac{N}{2}}^{nm}$$

$$a_N^{(2n+1)m} = e^{-\frac{2\pi i m}{N}} a_{\frac{N}{2}}^{nm}$$

В результате:

$$X_m = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} a_{\frac{N}{2}}^{nm} + e^{-\frac{2\pi i m}{N}} \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} a_{\frac{N}{2}}^{nm}$$

Вычисление данного выражения можно упростить, используя:

1. Свойство периодичности ДПФ:

$$a_{\frac{N}{2}}^{\left(m+\frac{N}{2}\right)n} = a_{\frac{N}{2}}^{nm}$$

2. Коэффициент поворота БПФ удовлетворяет следующему равенству:

$$e^{\left(-\frac{2\pi i}{N}\right)\left(m+\frac{N}{2}\right)} = e^{-\frac{2\pi i m}{N}-\pi i} = e^{-\pi i} e^{-\frac{2\pi i m}{N}} = -e^{-\frac{2\pi i m}{N}}$$

В результате упрощений, обозначив дискретное преобразование Фурье чётных индексов x_{2m} через E_m и преобразование нечётных индексов x_{2m+1} через O_m для $0 \le m \le \frac{N}{2}$ получается:

$$X_m = E_m + e^{-\frac{2\pi i m}{N}} O_m$$

$$X_{m+\frac{N}{2}} = E_m - e^{-\frac{2\pi i m}{N}} O_m$$

При рекурсивном делении дискретного преобразования Фурье от N входных значений на сумму 2 дискретных преобразований по $\frac{N}{2}$ входных значений сложность алгоритма становится равной $O\big(Nlog(N)\big)$.

Таким образом в программной реализации необходимо поделить исходную последовательность дискретных значений на две, после чего применить формулы X_m и $X_{m+\frac{N}{2}}$.

Обратное быстрое преобразование Фурье (IFFT\ОБПФ)

Эффективный алгоритм вычисления прямого (БПФ) можно использовать и для обратного преобразования.

Обратим внимание, что комплексные экспоненты в выражениях для прямого и обратного ДПФ являются комплексно-сопряженными:

$$e^{\frac{i2\pi nk}{N}} = \left(e^{-\frac{i2\pi nk}{N}}\right)^*$$

где (…) * — оператор комплексного сопряжения.

Нетрудно показать, что для двух комплексных чисел $x = a + ib$ и $y = c + id$ справедливо следующее равенство: $xy^* = (x^*y)^*$. Применительно для выражения ОДПФ можно записать:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k \left(e^{\frac{-2\pi ikn}{N}}\right)^* = \frac{1}{N}\left(\sum_{k=0}^{N-1} X_k e^{-\frac{i2\pi nk}{N}}\right)^*$$

Таким образом, берется комплексно-сопряженный спектр X_k^* выполняется прямое ДПФ и результат подвергается комплексному сопряжению. Если вместо ДПФ использовать БПФ, то получим обратное быстрое преобразование Фурье (ОБПФ). При этом для выполнения комплексного сопряжения необходимо лишь поменять знак перед мнимой частью спектра до вызова функции БПФ и результата после БПФ.

Асимптотическая сложность ОБПФ может быть получена из сложности БПФ и составляет $O\big(2Nlog(N)\big) \approx O\big(Nlog(N)\big)$.

4 Вычисления произведения многочленов с помощью быстрого преобразования Фурье

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 a и b

Выход: Произведение чисел a и b

Шаг 1: Представить числа a и b в виде списка коэффициентов полинома

Шаг 2: Перемножить полученные полиномы с помощью быстрого преобразования Фурье

Шаг 3: Обработать переносы в многочлене полученном на шаге 2

Шаг 4: Выдать результат произведения после преобразования переносов

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 $[23, 1, 34, 789876, 3, 4, 2]$.

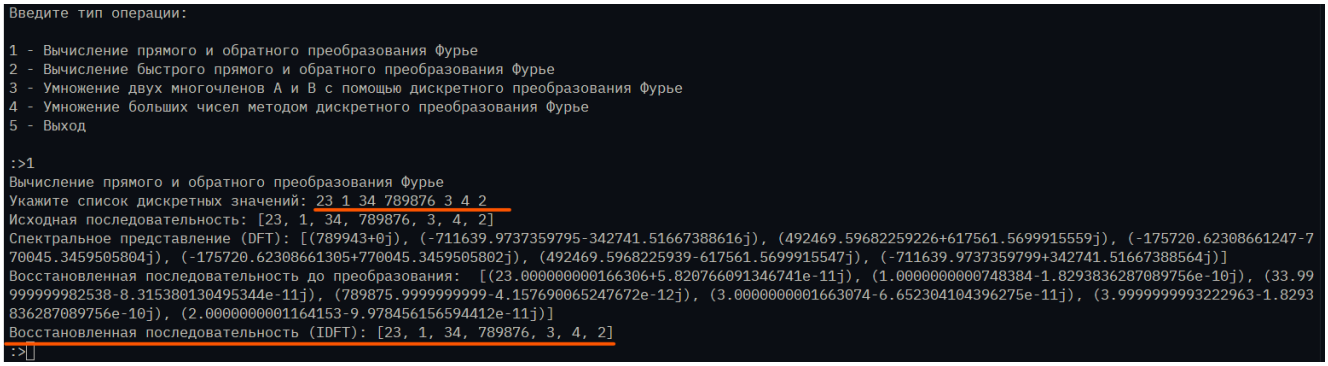

Рисунок 1 – Вычисление прямого и обратного преобразования Фурье на дискретных значениях: $[23, 1, 34, 789876, 3, 4, 2]$

Для проверки работоспособности протестируем быстрые алгоритмы прямого и обратного преобразования Фурье. Результаты тестирования представлены на рисунке.

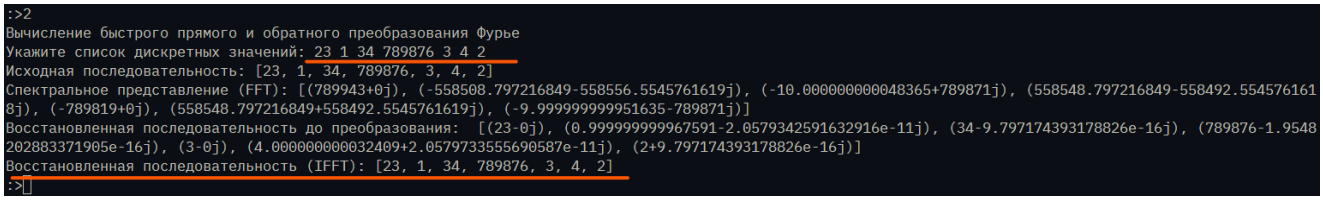

Рисунок 2 - Вычисление быстрого прямого и обратного преобразования Фурье при заданных значениях: $[23, 1, 34, 789876, 3, 4, 2]$

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

```
:>3
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]
[1872, 1847258, 7643371, -22307, 70885, 134078, 637150, 784237, -2300]
:>
```

Рисунок 3 – Вычисление произведения многочленов a и b

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

# Быстрое дискретное преобразование
def FFT(a):
    n = len(a)
    if n <= 1:
        return a

    # Разделяем на четные и нечетные индексы
    even = FFT(a[0::2])
    odd = FFT(a[1::2])

    # Вычисляем корень из единицы
    T = [math.e**(-2j * math.pi * k / n) * odd[k] for k in range(n // 2)]

    # Собираем результаты
    return [even[k] + T[k] for k in range(n // 2)] + \
           [even[k] - T[k] for k in range(n // 2)]

# Быстрое обратное дискретное преобразование
def IFFT(a):
    n = len(a)
    if n <= 1:
        return a
    a_conj = [x.conjugate() for x in a]
    y = FFT(a_conj)
    return [x.conjugate() / n for x in y]

# Проверяем на длину массива
def power_of_two(a):
    n = len(a)
    power_of_two = 1
    while power_of_two < n:
        power_of_two *= 2
```

```
    return a + [0] * (power_of_two - n)

# 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

# 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")
                lst = list(map(lambda x: int(x), input("Укажите
                           список дискретных значений: ").split()))
                len_ = len(lst)
                new_lst = power_of_two(lst)
                spectrum = FFT(new_lst)
                recovered_lst = IFFT(spectrum)
                print("Исходная последовательность:", lst)
                print("Спектральное представление (FFT):",
                       spectrum[:len_])
                print("Восстановленная последовательность до
                       преобразования: ", recovered_lst[:len_])
                print("Восстановленная последовательность (IFFT):",
                      [round(x.real) for x in recovered_lst[:len_]])
                param = None
            case "3":
                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]
                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"
```