Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

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

Кафедра теоретических основ компьютерной безопасности и криптографии

**Лабораторная работа №2**

студента 5 курса 531 группы

специальности 10.05.01 «Компьютерная безопасность»

факультета компьютерных наук и информационных технологий

Енца Михаила Владимировича

|  |  |  |
| --- | --- | --- |
| Преподаватель  профессор | \_\_\_\_\_\_\_\_\_\_ | В. А.  Молчанов |
|  | подпись, дата |  |
| Заведующий кафедрой  д.ф.-м.н., доцент | \_\_\_\_\_\_\_\_\_\_ | М. Б. Абросимов |
|  | подпись, дата |  |

Саратов 2018

**1. Постановка задачи.**

Изучение свойств дискретного преобразования Фурье и программная реализация его приложений.

**2. Теоретические сведения по рассмотренным темам с их обоснованием.**

*Дискретное преобразование Фурье.*

Пусть имеется многочлен -ой степени:

Не теряя общности, можно считать, что  является степенью 2. Если в действительности не является степенью 2, то просто добавим недостающие коэффициенты, положив их равными нулю.

Из теории функций комплексного переменного известно, что комплексных корней -ой степени из единицы существует ровно . Обозначим эти корни через , тогда известно, что  . Кроме того, один из этих корней   (называемый главным значением корня -ой степени из единицы) таков, что все остальные корни являются его степенями: .

*Дискретным преобразованием Фурье* (DFT) многочлена называются значения этого многочлена в точках , т.е. это вектор:

Аналогично определяется и *обратное дискретное преобразование Фурье* (InverseDFT). Обратное ДПФ для вектора значений многочлена – это вектор коэффициентов многочлена :

*Быстрое преобразование Фурье.*

*Быстрое преобразование Фурье* (БПФ или FFT) — это метод, позволяющий вычислять ДПФ за время . Этот метод основывается на свойствах комплексных корней из единицы.

Основная идея БПФ заключается в разделении вектора коэффициентов на два вектора, рекурсивном вычислении ДПФ для них, и объединении результатов в одно БПФ.

Пусть имеется многочлен  степени , где  — степень двойки, и :

Разделим его на два многочлена, один — с чётными, а другой — с нечётными коэффициентами:

Нетрудно убедиться, что:

Многочлены  и  имеют вдвое меньшую степень, чем многочлен . Если мы сможем, за линейное время, по вычисленным  и ,вычислить , то мы и получим искомый алгоритм быстрого преобразования Фурье.

Пусть имеем вычисленные вектора  и  . Найдём выражения для .

Сразу получаем значения для первой половины коэффициентов:

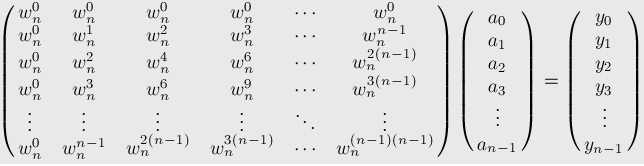
Для второй половины коэффициентов после преобразований также получаем простую формулу:

(Здесь воспользовались (1), а также тождествами   и  .)

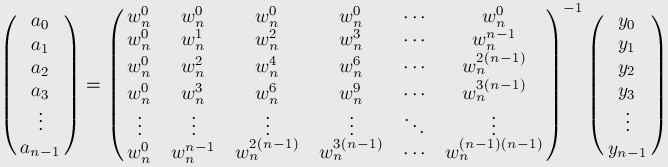
В результате получаем формулы для вычисления всего вектора :

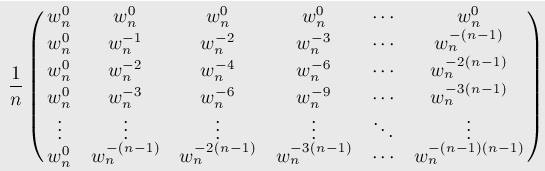
*Обратное быстрое преобразование Фурье.*

Пусть дан вектор  — значения многочлена  степени  в точках . Требуется восстановить коэффициенты многочлена. Эта задача называется *интерполяцией*, для этой задачи есть и общие алгоритмы решения, однако в данном случае будет получен простой алгоритм (простой тем, что он практически не отличается от прямого БПФ).



Тогда вектор можно найти, умножив вектор на обратную матрицу к матрице, стоящей слева (которая называется матрицей Вандермонда):



Непосредственной проверкой можно убедиться в том, что эта обратная матрица такова:

Таким образом, получаем формулу:

Сравнивая её с формулой для :

Можно заметить, что эти две задачи почти ничем не отличаются, поэтому коэффициенты  можно находить таким же алгоритмом "разделяй и властвуй", как и прямое БПФ, только вместо  везде надо использовать , а каждый элемент результата надо разделить на .

Таким образом, вычисление обратного ДПФ почти не отличается от вычисления прямого ДПФ, и его также можно выполнять за время .

Алгоритм быстрого преобразования Фурье.

*Вход*. Вектор .

*Выход*. Вектор , где Для .

1. Положить

2. Для выполнить следующие действия.

2.1. Положить .

2.2. Пока :

2.2.1. Представить полином в виде

2.2.2. Положить

2.2.3. Положить

2.2.4. Положить

2.3. Положить и вернуться на шаг 2.2.

3. Для положить

4. Результат:

**Коды программ, реализующей рассмотренные алгоритмы**

Программа реализована на языке Python (версия интерпретатора 3.6).

def fft(a, m, n, k):

w = primitive\_root(m, n)

r = {(0, k): a}

for s in range(k - 1, -1, -1):

t = 0

while t < n - 1:

a\_new = [r[(t, s + 1)][j] for j in range(pow(2, s + 1))]

e = int(bin(t // pow(2, s))[2:].zfill(k)[::-1], 2)

r[(t, s)] = [(a\_new[j] + pow(w, e) \* a\_new[j + pow(2, s)]) % m for j in range(pow(2, s))]

r[(t + pow(2, s), s)] = [(a\_new[j] + pow(w, e + n // 2) \* a\_new[j + pow(2, s)]) % m for j in

range(pow(2, s))]

t += pow(2, s + 1)

b = [0] \* n

for i in range(n):

b[int(bin(i)[2:].zfill(k)[::-1], 2)] = r[(i, 0)][0]

print('Вектор b = {}'.format(str(b)))

return b

def ifft(b, m, n, k):

w = utils.inverse(primitive\_root(m, n), m) % m

r = {(0, k): b}

for s in range(k - 1, -1, -1):

t = 0

while t < n - 1:

b\_new = [r[(t, s + 1)][j] for j in range(pow(2, s + 1))]

e = int(bin(t // pow(2, s))[2:].zfill(k)[::-1], 2)

r[(t, s)] = [(b\_new[j] + pow(w, e) \* b\_new[j + pow(2, s)]) % m for j in range(pow(2, s))]

r[(t + pow(2, s), s)] = [(b\_new[j] + pow(w, e + n // 2) \* b\_new[j + pow(2, s)]) % m for j in

range(pow(2, s))]

t += pow(2, s + 1)

a = [0] \* n

for i in range(n):

# print(int(bin(i)[2:].zfill(k)[::-1], 2))

# print(utils.inverse(m, n % m))

a[int(bin(i)[2:].zfill(k)[::-1], 2)] = (utils.inverse(n % m, m) \* r[(i, 0)][0]) % m

print('Вектор a = {}'.format(str(a)))

**Оценки сложности рассмотренных алгоритмов**

Сложность алгоритма «быстрое преобразование Фурье» и «быстрое обратное преобразование Фурье» равна .

**Результаты тестирования програмы**

Пример выполнения FFT:

Модуль .

Вектор

Вектор

Пример выполнения IFFT:

Модуль .

Вектор

Вектор