Задание на 8-ю неделю: 26.03–1.04. [0.12] Умножение многочленов и алгоритм БПФ Раздел 8 программы Литература: [Кормен 1, гл.6 и §32] [Кормен 2, гл 6. и §30], [ДПВ] [Виноградов]

Нужно обязательно прочитать статью (она выложена на нашем сайте): P. Clifford, R. Clifford. Simple deterministic wildcard matching. Information Processing Letters 101 (2007) 53–54. и посмотреть сайт

`https://en.wikipedia.org/wiki/Circulant_matrix`

КЛЮЧЕВЫЕ СЛОВА (минимальный необходимый объем понятий и навыков по этому разделу): Дискретное преобразование Фурье (ДПФ); быстрое преобразование Фурье (БПФ); схемы БПФ, перемножение многочленов с помощью БПФ. Поиск подстрок посредством БПФ. Циркулянты. Решение линейных уравнений с циркулянтными матрицами с помощью БПФ. Системы линейных уравнений с теплицевыми и ганкелевыми матрицами.

Замечание для семинаристов.

Вы уже знакомы с парой процедур, которые до какой-то степени определяют лицо нашего предмета (и не только нашего!). Это, конечно, алгоритм Евклида и метод Гаусса решения систем линейных уравнений. В этом задании мы разберем третий великий алгоритм: быстрое преобразование Фурье (БПФ). Несмотря на то, что этот метод был предложен значительно позже двух своих знаменитых собратьев[1], но в настоящее время он вряд ли уступает им по распространенности. Можно сказать, что в обыденной жизни вы встречаетесь с ним даже чаще, поскольку БПФ "вшито" чуть ли ни в каждый мобильник.

По определению, *дискретное преобразование Фурье (ДПФ)* — это отображение, переводящее последовательность коэффициентов (вообще говоря, комплексных) многочлена $f(x) = a_0 + a_1x^1 + \ldots + a_{n-1}x^{n-1} \in \mathbb{C}[x]$ в последовательность его значений в корнях n-й степени из единицы, т.е. $(a_0, a_1, \ldots, a_n) \mapsto (f(\omega_n^0), f(\omega_n^1), \ldots, f(\omega_n^{n-1}))$, где $\omega_n = \exp\frac{2\pi i}{n}$. На первый взгляд, кажется, что вычисление требует $\Omega(n^2)$ элементарных арифметических операций, ведь нужно подставить каждый корень из единицы в многочлен $f(\cdot)$ и повторить операцию n раз.

Обратите внимание, что мы пользуемся не нашей стандартной битовой моделью сложности, а используем "наивную" 'нью-мерическую' модель вычислений с плавающей запятой (float)". Иначе говоря, считаем, что у нас есть возможность достаточно точно производить ПРИБЛИЖЕННЫЕ операции с плавающей точкой так, что они существенно не повлияют на ответ. Сложностью называем полное число арифметических операций. Такая модель широко распространена на практике, но требует определенной квалификации вычислителя, причем, если степени полиномов и/или порядок используемых чисел достаточно велики, то уже обычным рецептом — повторением вычислений с двойной точностью — не отделаешься. Мы еще вернемся к этому вопросу.

Быстрое преобразование Фурье (БПФ) — это алгоритм, позволяющий вычислить ДПФ, используя $O(n \log n)$ операций.

Какое все это имеет отношение к мобильным телефонам? Приведу небольшой комментарий, хотя уверен, что вам все это известно на несколько порядков лучше, чем мне. Как известно, на заре эпохи вычислительных машин цифровые и аналоговые устройства серьезно конкурировали, но в настоящее время остались лишь островки аналоговых вычислений[2]. Иначе говоря, любой непрерывный сигнал $\phi(t)$, будь то электромагнитный импульс, посылаемый вашим мобильником, давление в аэродинамичесой трубе, распределение цветов на экране вашего монитора и т.д. должен быть отцифрован (это, как все понимают, может быть очень даже нетривиальной задачей), т.е. представлен массивом коэффициентов его значений в достаточно большом количестве точек $f(t) \mapsto (a_0, a_1, \ldots, a_{n-1})$. Здесь t может быть временем, пространственной координатой и т.д. Предположим теперь, что моделируемое устройство, на которое поступает сигнал (модем, телефон, экран, система управления ракетой и т.д.) линейное[3] (грубо говоря, при изменении интенсивности вдвое воздействие также изменяется в два раза и выполняется принцип суперпозициии: реакция от суммы сигналов равна сумме реакций), тогда его поведение может быть описано как импульсная или весовая функция в технике, фундаментальное решение или функция Грина в математике и физике и пр. По определению, импульсная функция — это реакция системы на единичный импульс в нулевой момент $t = 0$, т.е. на δ-функцию. В нашей дискретной модели реакция тоже должна задаваться массивом $(b_0, b_1, \ldots, b_{n-1})$ (считаем, что после i тактов на выходе b_i, а через n тактов реакция затухает).

Итак, пусть в момент времени $T = 0$ на вход поступает сигнал с интенсивностью a_0, который преобразуется системой в $b_0 \cdot a_0$, т.е. отклик системы в момент $T = 0$ равен a_0b_0. Далее, в момент $T = 1$ на вход поступает сигнал с интенсивностью a_1, вызывающий реакцию b_0a_1, которую согласно принципу суперпозиции нужно просуммировать с *остаточным* воздействием сигнала, пришедшего в момент $T = 0$, т.е. a_0b_1, и отклик системы равен $a_1b_0 + a_0b_1$ и т.д. Таким образом, при наших предположениях о поведении системы получаем, что если $T < n$ (т.е. сигнал еще поступает), то отклик равен $c_T = \sum_{i=0}^{T} a_ib_{T-i}$, а если $T \geq n$ (т.е. входной сигнал уже затух), то $c_T = \sum_{i=T-n+1}^{n-1} a_ib_{T-i}$. В момент $T = 2n-2$ отклик равен $c_{2n-2} = a_{n-1}b_{n-1}$, а далее отклик равен нулю. Таким образом, отклик системы $c_0, c_1, \ldots, c_{2n-2}$ в точности совпадает с последовательнстью коэффициенов многочлена- произведения:

$$\sum_{i=0}^{2n-2} c_ix^i \stackrel{def}{=} \sum_{i=0}^{n-1} a_ix^i \sum_{i=0}^{n-1} b_ix^i.$$

Итак, если мы хотим эффективно обрабатывать сигналы, то нужно научиться быстро умножать полиномы. Обычный способ умножения "столбиком" требует порядка n^2 операций, а алгоритм БПФ позволяет умножать многочлены почти на порядок быстрее. Но сначала разберемся, как, собственно, реализовать само это быстрое преобразование. Для этого мы используем симметрии корней из единицы. Заметим[4], что если n четное, то квадраты всех корней из единицы степени n образуют $\frac{n}{2}$ корней из единицы степени $\frac{n}{2}$ (можете ли вы сказать, сколько корней степени n перейдет в один корень степени $\frac{n}{2}$?). Теперь все готово для алгоритма. Попробуем применить технику "разделяй-и-властвуй". Итак, надо вычислить $f(\omega_n^k)$, $k = 0, \ldots, n-1$. Перепишем выражение по четным и нечетным индексам (считаем n четным) $f(\omega_n^k) = \sum_{i=0}^{\frac{n}{2}-1} a_{2i}\omega_n^{2ik} + \sum_{i=0}^{\frac{n}{2}-1} a_{2i+1}\omega_n^{2ik+k} = \sum_{i=0}^{\frac{n}{2}-1} a_{2i}(\omega_n^2)^{ik} + \omega_n^k \sum_{i=0}^{\frac{n}{2}-1} a_{2i+1}(\omega_n^2)^{ik}$, и задача разбивается на две аналогичные задачи для многочленов степени $\frac{n}{2} - 1$, поскольку ω_n^2 — это корень из единицы степени $\frac{n}{2}$. Получаем следующую рекуррентность для трудоемкости алгоритма $T(n) = 2T(\frac{n}{2}) + O(n)$, откуда $T(n) = O(n \log n)$

Остается применить изложенную технику для умножения многочленов. Для этого нужно вспомнить, что произвольный многочлен степени $n - 1$ может быть однозначно восстановлен по его значениям в произвольных n точках. Поэтому вместо того, чтобы пересчитывать коэффициенты многочлена $h(x) = f(x) \cdot g(x) = (\sum_{i=0}^{n-1} a_ix^i)(\sum_{i=0}^{n-1} b_ix^i) = \sum_{i=0}^{2n-2} c_ix^i = \sum_{i=0}^{2n-2}(\sum_{j=0}^{i} a_j \cdot b_{i-j})x^i$ (прямое вычисление коэффициентов требует $O(n^2)$ операций) мы сначала вычислим значения многочленов $f(x)$ и $g(x)$ в некоторых $2n - 1$ точках интерполяции, получив массивы $F = \{f_i\}$

[1] Алгоритм опубликовали в 1965 г. Д.Кули (James Cooley) и Д.Тьюки (John Tukey) — последний, между прочим, был консультантом по научным вопросам Джона Кеннеди,— но известно, что он применялся ранее и другими авторами. Как и следовало ожидать, его использовал "король математиков" К.-Ф. Гаусс в начале 19 века (записи были расшифрованы недавно).

[2] Хотя, с другой стороны, столь популярную в последнее время и даже проникшую в таблоиды модель квантовых вычислений, до которых мы, возможно, доберемся, можно рассматривать как явный реванш аналоговых вычислительных устройств. Если их удастся реализовать, то можно будет эффективно выполнять некоторые процедуры, весьма трудоемкие для обычных компьютеров, например факторизовать числа.

[3] Это почти общее предположение. Если же система нелинейная, то рассматривают ее линеаризацию.

[4] Хорошо бы действительно понять эти простые, но фундаментальные факты. Все они, конечно, доказываются в Кормене, но лучше попробовать получить их самому — это буквально одна строчка выкладок

и $G = \{g_i\}$. Покомпонентно перемножая F и G, получаем массив $H = \{f_i \cdot g_i\}$, который, по построению, отвечает значениям многочлена-произведения $h(\cdot)$ в тех же точках интерполяции, и $h(\cdot)$ можно однозначно восстановить и получить ответ.

Итак мы получили массив значений многочлена-произведения в корнях из единицы, и теперь нам нужно восстановить его коэффициенты. Конечно, если выбирать точки интерполяции совершенно произвольно, то не понятно, почему такой метод мог бы быть эффективнее прямого вычисления коэффициентов h. Однако, выбор точек интеполяции в корнях из единицы позволяет построить $O(n \log n)$-алгоритм перемножения многочленов! Точнее говоря, трудоемкость процедуры будет $O(n \log n \cdot M)$, где параметр M отвечает за трудоемкость отдельных арифметических операций. Конечно, если считать, что точность фиксирована, то M — константа, связанная с конкретной реализацией арифметических операций с плавающей точкой, и такая запись не очень осмыслена. Значит нужно попробовать реализовать точные вычисления, считая, как обычно, коэффициенты многочленов целыми. (Такая постановка реализуется во многих системах символьных вычислений или, например, в алгоритмах быстрого перемножения больших чисел, которые, как мы помним, как раз и используют точное БПФ в качестве подпрограмы.) Мы не будем сейчас останавливаться на этом подробно (в качестве частичных рецептов посмотрите упр. 5 и задачу 32.5 из Кормена (1-е издание, — во втором издании номер параграфа 30).

А сейчас ответим на исходный вопрос, почему можно быстро за $O(n \log n)$ операций восстановить коэффициенты многочлена по его ДПФ. По определению, ДПФ задается умножением матрицы типа Вандермонда $\tilde{f} = \|(\omega_n)^{ij}\|_{i,j=0,1,\ldots,n-1}$ на вектор-столбец $(a_0, a_1, \ldots, a_{n-1})$. Действительно, для корней из единицы справедливо тождество: для любого k выполнено

$$\sum_{i=0}^{n-1} (\omega_n^k)^i = \delta_{0k \bmod n} \cdot n \tag{1}$$

(δ_{ij} — это символ Кронекера — дискретная δ-функция), поэтому обратную матрицу можно задать так:

$$\tilde{f}^{-1} = \frac{1}{n} \|(\omega_n^{-1})^{ij}\|_{i,j=0,1,\ldots,n-1} \tag{2}$$

и по аналогичной причине и обратное ДПФ (умножение матрицы $\tilde{f}^{-1}$ на массив коэффициентов) можно вычислить, используя $O(n \log n)$ операций.

Еще один момент, который я хотел бы затронуть в связи с БПФ, это его возможное нерекурсивное исполнение. Оказывается, что БПФ можно задать явной схемой, которая к тому же допускает параллельное выполнение (см. рис. в книге [ДПВ]).

На схеме ребра-провода "несут" информацию слева направо, т. е. каждому ребру приписано (комплексное) число. Метка i на ребре означает, что число, приписанное проводу, нужно умножить на ω^i. В вершинах, обозначенных черными кружками, нужно просуммировать числа, приписанные проводам, приходящим *слева*. По схеме видно, что ее основу составляет знаменитая "бабочка", отвечающая используемой в БПФ группировке степеней на четные и нечетные и переходу к половинной размерности:

$$s_i = t_i + \omega^i u_i, \qquad s_{i+n/2} = t_i - \omega^i u_i.$$

Зачем нужно писать явные схемы, когда уже построен рекурсивный алгоритм? Во-первых, рекурсия может быть неэффективной при реализации, а явная схема проще. Во-вторых, схема показывает, как запустить алгоритм параллельно. В третьих, используя структуру схемы можно решать полезные смежные задачи, например, искать некоторые ошибки, см., задачу **№ 4**.

Мы разобрались, как быстро перемножать многочлены с комплексными коэффициентами. А можно ли придумать похожий трюк для умножения многочленов с коэффициентами в кольце вычетов $\mathbb{Z}_n$? Если бы это удалось, то при умножении многочленов с целыми коэффициентами можно было бы обойти упомянутый выше вопрос о точности вычислений в ДПФ, рассматривая достаточно большие модули и проводя операции в $\mathbb{Z}_n$. Частичный ответ на этот вопрос будет дан в задаче

Быстрое вычисление свертки посредством БПФ используется во многих областях, например, в такой важной области, как анализ изображений. Изображение можно считать числовой матрицей [большого] размера, а типичной задачей может быть выделение [сравнительно небольшого] блока с заданными свойствами (под блоком понимается не минор, а подматрица исходной матрицы, в которой столбцы и строки идут подряд). Интересно, что содержательна и одномерная задача — поиск подстрок, с которой успешно справляется произвольный текстовый редактор. Казалось бы, эту задачу мы успешно решили в курсе ТРЯП, построив **оптимальный** линейный КМП-алгоритм. Но оказывается, что примерно в то же время (в 1974 г.), когда были предложены и КМП, и Boyer-Moore алгоритмы, был предложен быстрый алгоритм, основанный на БПФ. Трудоемкость БПФ-алгоритма поиска подстрок отличается от оптимально логарифмическим фактором.

Идея БПФ-алгоритма поиска подстрок следующая. Нужно найти подстроку (образец) $p_0, \ldots, p_{m-1}$ в строке (тексте) $t_0, \ldots, t_{n-1}$, здесь p_i, t_j — это символы некоторого алфавита. Говорят, что подстрока входит с i-й позиции, если $p_j = t_{i+j}$, $j = 0, \ldots, m-1$. Если считать буквы алфавита различными целыми числами, то вхождение подстроки с i-й позиции эквивалентно обнулению суммы квадратов: $B_i = \sum_{j=0}^{m-1} (p_j - t_{i+j})^2 = \sum_{j=0}^{m-1} (p_j^2 - 2p_j t_{i+j} + t_{i+j}^2) = 0$, а вычисление массива чисел $\{B_i,\ i = 0, \ldots, n-m\}$ позволяет определить все места вхождения подстроки в текст.

"Наивный" алгоритм — прямой перебор требует $O(mn)$ операций. Но в курса ТРЯП изучался более быстрый, линейный, алгоритм. С помощью БПФ можно построить $O(n \log m)$-процедуру. Сумма $S = \sum_{j=0}^{m-1} p_j^2$ присутствует как слагаемое в каждом B_i, и вычисляется за $O(n)$ шагов.

Рассмотрим два полинома степени не выше n: $T(x) = t_{n-1}x^{n-1} + \cdots + t_1 x + t_0$, $P(x) = p_0 x^{n-1} + \cdots + p_{m-1}$ Их произведение можно вычислить с помощью БПФ за $O(n \log n)$. Посмотрим на коэффициент их произведения $C(x)$ при x^{m-1+i} $(0 \le i \le n-m)$:

$$c_{m-1+i} = p_0 t_i + p_1 t_{i+1} + p_2 t_{i+2} + \cdots + p_{m-2} t_{m-2+i} + p_{m-1} t_{m-1+i} = \sum_{j=0}^{m-1} p_j t_{j+i}$$

Как видим, это одно из слагаемых для B_i. Таким образом, алгоритм для подсчета всех B_i таков. Сначала вычисляем за $O(n \log n)$ шагов коэффициенты произведения $C(x)$ указаных многочленов. Далее за максимум $O(n)$ шагов вычисляем сумму квадратов S и за $O(n)$ шагов считаем сумму первых m квадратов t_i, т. е. $H = \sum_{j=0}^{m-1} t_j^2$.

Далее вычисляем $B_0 = S - 2 \cdot c_{m-1} + H$ и $B_1 = S - 2 \cdot c_m + \sum_{j=0}^{m-1} t_{j+1}^2 = B_1 + 2 \cdot c_{m-1} - 2 \cdot c_m - t_1^2 + t_m^2$.

Для этого нам потребуется $O(1)$ операций. Аналогично получим $B_i = B_{i-1} + 2(c_{m-2+i} - c_{m-1+i}) - t_{i-1}^2 + t_{m-1+i}^2$, т. е. для получения каждого следующего члена требуется $O(1)$ операций. Таким образом, для вычисления всех членов потребуется еще $O(n)$ операций. Таким образом, весь алгоритм асимптотически требует $O(n \log n)$ операций.

Ту же идею для проверки вхождения подстроки можно использовать, если разрешается использовать символ джокера (в курсе ТРЯП он обозначался знаком «?»). Тогда в тех же обозначениях нужно вычислить массив $\{A_i,\ i = 0, \ldots, n-m\}$, где $A_i = \sum_{j=0}^{m-1} (p_j^3 t_{i+j} - 2p_j^2 t_{i+j}^2 + p_j t_{i+j}^3)$.

Наконец, еще одно важное приложение ДПФ связано с эффективным решением специальных систем линейных уравнений, в которых матрицы коэффициентов имеют специальные симметрии, а именно, постоянны вдоль каких-то "диагоналей" и тогда умножение на такую матрицу можно представить сверткой. В качестве примера таких матриц обычно рассматриваются циркулянты (строки получены циклическим сдвигом фиксированного вектора), теплицевы или ганкелевы матрицы, у которых на всех диагоналях, параллельных главной (соответственно, параллельных побочной), стоят равные элементы.

Вы должны знать, как, используя БПФ, решать системы линейных уравнений порядка $n \times n$ с циркулянтной матрицей, используя $o(n \log n)$ операций, и представлять себе, как выполнить эту задачу для случая теплицевых или ганкелевых матриц, используя $O(n^2)$ операций.

Задача 1. $(0.02 + 0.02)$. (i) Найдите произведение мно-

гочленов $A(x) = 3x + 2$ и $B(x) = x^2 + 1$, используя рекурсивный $O(n \log n)$-алгоритм БПФ.

Для этого нужно выбрать n, Найти БПФ обоих полиномов, затем вычислить ДПФ многочлена- произведения и на последнем шаге вычислить обратное ДПФ, т. е. умножить вектор-столбец, полученный в предыдущем пункте, на обратную матрицу ДПФ. Используя тождество (2), и при этом вычисление можно тоже использовать БПФ (каким образом?).

(*ii*) Вычислите **обратное ДПФ** массива (I — мнимая единица) $A = [10, 3I\sqrt{2} + 2 + 2I, 0, 3I\sqrt{2} + 2 - 2I, -2, -3I\sqrt{2} + 2 + 2I, 0, -3I\sqrt{2} + 2 - 2I]$, используя схему БПФ для $n = 8$ на рис. 1. Возможно, вычисление будет удобнее проводить символически для $\omega = \exp\left(\frac{\pi I}{4}\right)$.

Задача 2. (0.01 + 0.02). (*i*) Постройте $O(n \log n)$-БПФ-алгоритм для поиска подстрок в тексте с "джокерами". (*ii*) Покажите, как понизить трудоемкость вашей процедуры до $O(n \log m)$.

Задача 3. (0.01). Используя ДПФ, найдите решение системы линейных уравнений $Cx = b$. где C — это циркулянтная матрица, порожденная вектором столбцом $(1, 2, 4, 8)^t$, а $b^t = (16, 8, 4, 2)$.

Задача Д–1. (0.03). Дан множество различных чисел $A \subseteq \{1, \ldots, m\}$. Рассмотрим множество $A+A$, образованное суммами элементов A. Докажите или опровергните существование процедур построения $A+A$, имеющих субквадратичную трудоемкость $o(m^2)$.

Задача 4. (0.02). (**[Кормен 1, упр. 32.3-4] или [Кормен 2, упр. 30.3-4]**. Предположим, что в схеме БПФ, изображенной на рис. 1, вышел из строя ровно один сумматор, причем он выдает число 0 независимо от входа. Сколько и каких последовательностей нужно подать на вход, чтобы идентифицировать дефектный сумматор?

ДПФ и БПФ в кольце вычетов $\mathbb{Z}_n$

Сначала мы рассмотрим простой случай, когда кольцо — это простое поле GF_p и попробуем просто осуществить в нем ДПФ. Понятно, что роль корня из единицы должен играть первообразный корень, поскольку мы знаем, что определитель Вандермонда, составленный из степеней произвольного первоообразного корня, обладает всеми нужными нам свойствами, причем в арифметике $\mathbb{Z}_p$, потому что речь идет опять о сумме конечной геометрической прогрессии, период которой делится на порядок данного элемента. Итак для того, чтобы перемножить многочлены можно взять достаточно большое (насколько большое?) простое число p, перемножить многочлены $\pmod p$, используя ДПФ в конечном поле, а затем восстановить коэффициенты.

Задача 5. (0.03). Выберите подходящее простое число p и перемножьте многочлены $A(x)$ и $B(x)$ над конечным полем, а затем восстановите многочлен-произведение над $\mathbb{Z}$.. Обоснуйте выбор p. Скажем, можно ли взять $p = 5$? Или следует выбрать $p = 7$ или больше?

В предыдущей задаче мы обосновали ДПФ над конечным полем, но можно ли провести трюк с БПФ? Об этом следующие задачи. Напоминаем, что, на самом деле, мы пытаемся уточнить предыдущую модель, когда предполагалось, что арифметические операции проводятся точно.

Предположим, что в $\mathbb{Z}_n$ есть примитивный корень ξ степени[5] 2^k. Неформально модуль n будет означать максимальную границу модулей коэффициентов, которые могут встретиться в сомножителях и в произведении, а степень произведения не должна превышать $l = 2^k - 1$. Степень двойки требуется для того, чтобы прошла рекурсия в алгоритме БПФ

Задача Д–2. (0.01 + 0.01 + 3 × 0.02).
(*i*) Покажите, что 3 является примитивным корнем 8-й степени в простом поле $\mathbb{Z}_{41}$.

Рассмотрим многочлен $h \in \mathbb{Z}_n[x]$ степени не выше $l = 2^k - 1$. Матрицу $\Xi = \|\xi^{ij}\|_{i,j=0,1,\ldots,l}$ назовем матрицей ДПФ. По определению, ДПФ многочлена h равно произведению матрицы Ξ на вектор-столбец коэффициентов h

(*ii*) Покажите, что для любого $j = 0, 1, \ldots, k-1$ элемент ξ^{2^j} — есть примитивный корень степени 2^{k-j} в $\mathbb{Z}_n$.

Из последнего упражнения вытекает, что можно быстро вычислять ДПФ произвольного многочлена h степени не выше l в $\mathbb{Z}_n$ ровно таким же рекурсивным алгоритмом, который мы использовали в $\mathbb{C}$.

(*iii*) Считая, что многочлены A и B (из задания) заданы над $\mathbb{Z}_{41}$ и $\xi = 3$ вычислите рекурсивным алгоритмом их ДПФ.

Перемножая покомпонентно вычисленные в предыдущем пункте ДПФ многочленов A и B, мы получаем ДПФ произведения C. Теперь нам осталось проверить, что можно эффективно восстановить коэффициенты C по его ДПФ. Для этого нужно проверить выполнения равенства, аналогичного (2).

(*iv*) Докажите, что $\Xi^{-1} = (l+1)^{-1}\|(\xi^{-1})^{ij}\|_{i,j=0,1,\ldots,l}$. Формула, конечно, справедлива для произвольной степени примитивного элемента ξ, а не только для степени 2^k, как в нашей спецификации.

(*v*) Используя результаты (*iii*) и (*iv*), вычислите ДПФ многочлена C и восстановите его коэффициенты рекурсивным алгоритмом БПФ [умножения обратной матрицы Ξ^{-1} на вектор столбец ДПФ].

Конечно, выбранный нами размер поля может оказаться недостаточным для того, чтобы произвести вычисления, поэтому дополнительно приведите обоснование корректности (или некорректности) выбора размера поля (само вычисление должно быть выполнено в любом случае).

[5] Это, по определению значит, что $\xi^i \neq 1$, $i = 0, 1, \ldots 2^k - 1$, а $\xi^{2^k} = 1$.