Дискретное преобразование Фурье. Семинар 8. 28 марта 2019 г.

Подготовил: Горбунов Э.

Источники: [Кормен 1, §32]; [Кормен 2, §30]; P. Clifford, R. Clifford, Simple deterministic wildcard matching; Циркулянт; матрица Тёплица; матрица Ганкеля; ДПФ; лекции Тыртышникова Е.Е., лекция 34

Ключевые слова: дискретное преобразование Фурье (ДПФ), быстрое преобразование Фурье (БПФ), перемножение многочленов с помощью БПФ, поиск подстрок при помощи БПФ, циркулянты, решение систем линейных уравнений с циркулянтными матрицами

Мотивировка

Мы подобрались к одному из важнейших алгоритмов нашего курса — быстрому преобразованию Фурье (БПФ)[1]. Давайте для начала поймём, почему БПФ является столь важным методом[2]. Кроме того, сразу договоримся, что в данном семинаре, говоря о временной сложности, мы будем иметь в виду число арифметических операций.

Как известно, на заре эпохи вычислительных машин цифровые и аналоговые устройства серьезно конкурировали, но в настоящее время остались лишь островки аналоговых вычислений[3]. 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[4] (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 $a_0 b_0$. Далее, в момент $T = 1$ на вход поступает сигнал с интенсивностью a_1, вызывающий реакцию $b_0 a_1$, которую согласно принципу суперпозиции нужно просуммировать с *остаточным* воздействием сигнала, пришедшего в момент $T = 0$, т. е. $a_0 b_1$, и отклик системы равен $a_1 b_0 + a_0 b_1$ и т.д. Таким образом, при наших предположениях о поведении системы получаем, что если $T < n$ (т. е. сигнал еще поступает), то отклик равен $c_T = \sum\limits_{i=0}^{T} a_i b_{T-i}$, а если $T \geq n$ (т. е. входной сигнал уже затух), то $c_T = \sum\limits_{i=T-n+1}^{n-1} a_i b_{T-i}$. 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:

$$\sum_{i=0}^{2n-2} c_i x^i \stackrel{\text{def}}{=} \sum_{i=0}^{n-1} a_i x^i \sum_{i=0}^{n-1} b_i x^i.$$

[1] Описание данного алгоритма будет введено позднее. Сам метод был предложен сравнительно недавно, а именно, в 1965 г. (Д. Кули и Д. Тьюки). Однако этот алгоритм был известен некоторым математикам и ранее. Например, его использовал «король математиков» Карл Фридрих Гаусс ещё в начале 19-го века (записи были расшифрованы недавно).

[2] Следующий мотивационный пример взят из канонического задания.

[3] 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.

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

Итак, если мы хотим эффективно обрабатывать сигналы, то нужно научиться быстро умножать полиномы. Обычный способ умножения «столбиком» требует порядка n^2 арифметических операций, а **алгоритм БПФ позволяет умножать многочлены почти на порядок быстрее**.

Представление многочленов

Задание многочлена вектором коэффициентов

Произвольный многочлен $f(x)$ от переменной x над полем F имеет вид

$$f(x) = \sum_{j=0}^{n-1} a_j x^j, \quad a_0, a_1, \ldots, a_{n-1} \in F.$$

Далее мы будем считать, что $F = \mathbb{C}$ — поле комплексных чисел. Из общего вида многочлена следует, что многочлен может быть задан вектором коэффициентов $a = (a_0, a_1, \ldots, a_{n-1})^\top$. Представление многочлена коэффициентами удобно в целом ряде случаев. Например, операция **вычисления значения многочлена $f(x)$ в данной точке** x_0 может быть выполнена за время $\Theta(n)$ по **схеме Горнера**:

$$f(x_0) = a_0 + x_0(a_1 + x_0(a_2 + \ldots + x_0(a_{n-1}))\ldots).$$

Сложение многочленов, заданных векторами коэффициентов, также требует времени $\Theta(n)$ (нужно покомпонентно сложить вектора коэффициентов).

Рассмотрим умножение двух многочленов $f(x)$ и $g(x)$ степени ниже n. Если $f(x) = \sum\limits_{j=0}^{n-1} a_j x^j$ и $g(x) = \sum\limits_{j=0}^{n-1} b_j x^j$, то

$$h(x) \stackrel{\text{def}}{=} f(x) \cdot g(x) = \sum_{k=0}^{2n-2} c_k x^k, \quad \text{где } c_k = \sum_{j=0}^{k} a_j b_{k-j}.$$

Если непосредственно использовать формулу выше для коэффициентов c_k, то на их вычисление придётся потратить время $\Theta(n^2)$ (каждый коэффициент многочлена f нужно умножить на каждый коэффициент многочлена g). Как мы видим, умножение многочленов — самая трудоёмкая арифметическая операция при работе с многочленами. Поэтому нам нужно научиться выполнять её быстро.

Задание многочлена набором значений

Зафиксируем n различных точек $x_0, x_1, \ldots, x_{n-1}$. Многочлен степени ниже n однозначно определяется своими значениями в этих точках, то есть при помощи n пар аргумент-значение

$$\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}, \quad \text{где} \quad y_k = f(x_k) \quad \text{для} \quad k = 0, 1, 2, \ldots, n-1.$$

Если вычислять отдельно значение полинома в каждой из n точек по схеме Горнера за $\Theta(n)$ шагов, то переход от коэффициентов многочлена $f(x)$ к его значениям требует времени $\Theta(n^2)$. Далее мы увидим, что время можно сократить — при подходящем выборе точек достаточно $O(n \log n)$ операций.

В таком случае, важно уметь производить и обратный переход — от набора значений многочлена, к его коэффициентам. Этот переход называется **интерполяцией**, которая, как утверждает следующая теорема, выполняется однозначно.

Теорема 1. (Однозначность интерполяции). Для любого множества пар $\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$ пар аргумент-значение, где все x_i различны, существует единственный многочлен $f(x)$ степени ниже n, для которого $y_k = f(x_k)$ для $k = 0, 1, \ldots, n-1$.

Доказательство. Уравнения $y_k = f(x_k)$, $k = 0, 1, 2, \ldots, n-1$ можно записать в матричном виде:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \ldots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix}$$

Левая матрица называется матрицей Вандермонда и обозначается $V(x_0, x_1, \ldots, x_{n-1})$. Можно показать, что её определитель равен $\prod\limits_{j<k}(x_k - x_j)$. Значит, в силу того, что точки $x_0, x_1, \ldots, x_{n-1}$ попарно различны, то данный определитель не равен нулю, то есть матрица обратима. Но это как раз и означает, что указанная линейная система имеет единственной решение, которое определяется по формуле

$$a = \left(V(x_0, x_1, \ldots, x_{n-1})\right)^{-1} y.$$

□

Это доказательство сводит задачу интерполяции к решению системы линейных уравнений. Если это делать классическими методами (использовать LU-разложение), потребуется время $\Theta(n^3)$. Более быстрый алгоритм интерполяции основывается на формуле Лагранжа:

$$f(x) = \sum_{k=0}^{n-1} y_k \frac{\prod\limits_{j\neq k}(x - x_j)}{\prod\limits_{j\neq k}(x_k - x_j)}.$$

В качестве упражнения предлагаю показать, что коэффициенты многочлена f можно вычислить за время $\Theta(n^2)$, если использовать интерполяционную формулу Лагранжа[5].

Итак, мы выяснили, что переходить от набора n коэффициентов к набору значений в n точках и обратно мы можем за $O(n^2)$ операций. Однако, как уже было отмечено ранее, при помощи БПФ эти переходы можно делать за $O(n \log n)$.

Отметим, что представление многочлена с помощью значений в заданных точках удобно для многих операций: например, для сложения достаточно сложить значения многочленов в каждой из точек. Аналогичная ситуация и с умножением (за исключением того, что значения нужно перемножать и точек нужно брать $2n - 1$ штук, а не n).

Дискретное преобразование Фурье. Быстрое преобразование Фурье

Основная идея алгоритма ДПФ состоит в использовании так называемых комплексных корней из единицы в качестве точек, в которых задаются значения многочлена.

Комплексные корни из единицы

Комплексным корнем степени n из единицы называют такое комплексное число ω, что $\omega^n = 1$. Имеется ровно n комплексных корней степени n из единицы. Они имеют вид $e^{\frac{2\pi i k}{n}}$ для[6] $k = 0, 1, \ldots, n-1$. Значение $\omega_n = e^{\frac{2\pi i}{n}}$ называется **главным значением корня степени n из единицы**, а остальные корни являются его степенями. Более того, комплексные корни степени n из единицы образуют группу по умножению, изоморфную группе $(\mathbb{Z}_n, +)$: действительно, из $\omega_n^n = \omega_n^0 = 1$ следует, что $\omega_n^j \omega_n^k = \omega_n^{j+k} = \omega_n^{(j+k) \mod n}$. Аналогично, $\omega_n^{-1} = \omega_n^{n-1}$. Докажем некоторые свойства корней из единицы.

Лемма 1 (О сокращении)**.** Для любых целых $n \geqslant 0$, $k \geqslant 0$ и $d > 0$ выполняется

$$\omega_{dn}^{dk} = \omega_n^k.$$

Доказательство. $\omega_{dn}^{dk} = \left(e^{\frac{2\pi i}{dn}}\right)^{dk} = \left(e^{\frac{2\pi i}{n}}\right)^{k} = \omega_n^k.$ □

Следствие 1. Для любого чётного $n > 0$ выполняется

$$\omega_n^{\frac{n}{2}} = \omega_2 = -1.$$

[5] Подсказка: покажите, что достаточно посчитать за $\Theta(n^2)$ произведение $\prod\limits_j (x - x_j)$ по всем j, а затем для каждого k поделить на многочлен $(x - x_k)$ за время $\Theta(n)$.

[6] Полезно помнить формулу Эйлера: $e^{iu} = \cos u + i \sin u$.

Лемма 2 (О делении пополам)**.** Если $n > 0$ чётно, то, возведя в квадрат все n комплексных корней степени n из единицы, мы получим все $\frac{n}{2}$ комплексных корней степени $\frac{n}{2}$ из единицы.

Доказательство. Корни ω_n^k и $\omega_n^{k+\frac{n}{2}}$ отличаются знаком и при возведении в квадрат дают одно и то же число $\omega_n^{2k} = \omega_{\frac{n}{2}}^k$. □

Лемма о делении пополам позволит свести вычисление значений многочлена в корнях степени n из единицы к вычислению значений других многочленов в корнях степени $\frac{n}{2}$ из единицы.

Лемма 3 (О сложении)**.** Для любого целого $n \geqslant 0$ и неотрицательного целого k, не кратного n, выполнено равенство

$$\sum_{j=0}^{n-1} (\omega_n^k)^j = 0.$$

Доказательство. Имеем

$$\sum_{j=0}^{n-1} (\omega_n^k)^j = \frac{(\omega_n^k)^n - 1}{\omega_n^k - 1} = \frac{(\omega_n^n)^k - 1}{\omega_n^k - 1} = \frac{1^k - 1}{\omega_n^k - 1} = 0,$$

где знаменатель не равен нулю, так как k не кратно n. □

Дискретное преобразование Фурье

Дискретное преобразование Фурье — это отображение из $\mathbb{C}^n$ в $\mathbb{C}^n$, ставящее в соответствие вектору $a = (a_0, a_1, \ldots, a_{n-1})^\top$ вектор $y = (y_0, y_1, \ldots, y_{n-1})^\top$, где $y_j = f(\omega_n^j)$, $j = 0, 1, \ldots, n-1$ и

$$f(x) = \sum_{j=0}^{n-1} a_j x^j.$$

Иными словами, мы поставили в соответствие вектору коэффициентов многочлена f вектор значений полинома f в корнях степени n из единицы. Вектор y называется дискретным преобразованием Фурье вектора a и обозначается $y = \mathrm{DFT}_n(a)$.

Быстрое преобразование Фурье

На первый взгляд кажется, что ДПФ вычисляется за время $\Theta(n^2)$, ведь нужно подставить каждый корень степени n из единицы в многочлен f. Однако существует *быстрое преобразование Фурье* — метод быстрого вычисления преобразования Фурье , использующий свойства комплексных корней из единицы и требующий времени $\Theta(n \log n)$.

Во-первых, будем считать, что $f(x)$ имеет степень $n-1$, где n — это степень двойки (этого достаточно, потому что к многочлену всегда можно добавить нулевые старшие коэффициенты) БПФ использует знакомый нам с первого семинара метода «разделяй и властвуй». Основная идея — выделить в многочлене f отдельно члены чётных и нечётных степеней, записав

$$f(x) = f^{[0]}(x^2) + x f^{[1]}(x^2),$$

где

$$f^{[0]}(x) = a_0 + a_2 x + a_4 x^2 + \ldots + a_{n-2} x^{\frac{n}{2}-1}$$

и

$$f^{[1]}(x) = a_1 + a_3 x + a_5 x^2 + \ldots + a_{n-1} x^{\frac{n}{2}-1}.$$

Тем самым задача вычисления $f(x)$ в точках $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$ сводится к

1) вычислению значений многочленов $f^{[0]}$ и $f^{[1]}$ степени меньше $\frac{n}{2}$ в точках $(\omega_n^0)^2, (\omega_n^1)^2, \ldots, (\omega_n^{n-1})^2$;

2) комбинации результатов.

В силу леммы о делении пополам мы свели исходную задачу к двум подзадачам размера в два раза меньше и тратим на это время $\Theta(n)$ (чтобы получить значение многочлена, зная решение подзадач, нужно сделать порядка n умножений и сложений). Таким образом, рекуррента для времени работы БПФ получается следующей:

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) = \Theta(n \log n).$$

Заметим, что рекуррента получилась в точности такой же, как и для процедуры Merge-Sort.

Интерполяция по значениям в корнях из единицы

Осталось научиться переходить от значений многочлена в комплексных корнях из единицы к его коэффициентам. Оказывается, ДПФ можно записать в матричном виде:

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_n & \omega_n^2 & \omega_n^3 & \dots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & \omega_n^6 & \dots & \omega_n^{2(n-1)} \\ 1 & \omega_n^3 & \omega_n^6 & \omega_n^9 & \dots & \omega_n^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \omega_n^{3(n-1)} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix},$$

то есть $y = V_n a$, где элемент матрицы V_n с индексами (k, j) равен ω_n^{kj} (при $j, k = 0, 1, \dots, n-1$). Обратная операция $a = \mathrm{DFT}_n^{-1}(y)$ состоит в умножении матрицы V_n^{-1} (обратной к V_n) слева на y.

Теорема 2. Элемент с индексами (j, k) матрицы V_n^{-1} равен $\frac{\omega_n^{-kj}}{n}$.

Доказательство. Покажем, что составленная из таких элементов матрица V_n^{-1} удовлетворяет требованию $V_n^{-1} V_n = I_n$, где I_n — единичная матрица размера $n \times n$. По определению, (j, j')-й элемент произведения $V_n^{-1} V_n$ есть

$$\left[V_n^{-1} V_n\right]_{jj'} = \sum_{k=0}^{n-1} \left(\frac{\omega_n^{-kj}}{n}\right) (\omega_n^{kj'}) = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^{k(j'-j)}.$$

По лемме о сложении последняя сумма равна 1, если $j = j'$, и равна 0 при $j \neq j'$. Действительно, $-(n-1) \leqslant j' - j \leqslant n-1$, так что $j - j'$ не делится на n при $j \neq j'$. □

Утверждение выше можно записать в более компактной форме. Обозначим через A^* комплексно сопряжённую матрицу к A: $(A^*)_{i,j} = \overline{A_{j,i}}$. Тогда верно

Следствие 2. $V_n^{-1} = \frac{1}{n} V_n^*$.

Зная обратную матрицу V_n^{-1}, мы можем найти $a = \mathrm{DFT}_n^{-1}(y)$ по формуле

$$a_j = \frac{1}{n} \sum_{k=0}^{n-1} y_k \omega_n^{-kj}$$

для $j = 0, 1, \dots, n-1$. Мы видим, что для вычисления обратного преобразования Фурье можно применить тот же алгоритм БПФ, поменяв в нём a и y местами, заменив ω_n на ω_n^{-1} и разделив каждый элемент результата на n. Отсюда следует, что DFT_n^{-1} также можно вычислить за $O(n \log n)$.

Перемножение многочленов при помощи БПФ

Теперь мы можем описать последовательность действий для перемножения многочленов, заданных своими коэффициентами. Для этого мы сначала перейдём к значениям в комплексных корнях из единицы за время $\Theta(n \log n)$; затем перемножим значения, получив значения многочлена-произведения в комплексных корнях из единицы за время $\Theta(n)$; после этого применим к полученным значениям обратное преобразование Фурье за время $\Theta(n \log n)$. Общее время работы составит $\Theta(n \log n)$.

Отметим ещё одну важную деталь. Существует класс задач, которые каким-то образом (бывает, что очень хитро, а бывает и прямолинейно) сводятся к вычислению *свёрток* двух последовательностей. *Свёрткой* двух

последовательностей $\{a_n\}_{n\geqslant 0}$ и $\{b_n\}_{n\geqslant 0}$ называется такая последовательность $\{c_n\}_{n\geqslant 0}$, что $c_k = \sum\limits_{j=0}^{k} a_j b_{k-j}$. Для конечных последовательностей можно считать, что если индекс больше, чем количество элементов в последовательности, то соответствующий элемент равен нулю. Заметим, что если перемножить 2 многочлена $f(x) = a_0 + a_1 x + \dots a_{n-1}x^{n-1}$ и $g(x) = b_0 + b_1 x + \dots b_{n-1}x^{n-1}$, то получится многочлен $h(x) = f(x)g(x)$, коэффициенты которого — это свёртка последовательностей $\{a_k\}$ и $\{b_k\}$: $h(x) = f(x)g(x) = c_0 + c_1 x + \dots + c_{2n-2}x^{2n-2}$, $c_k = \sum\limits_{j=0}^{k} a_j b_{k-j}$. Следовательно, мы научились вычислять свёртки двух последовательностей длины n за $O(n \log n)$. Рассмотрим теперь некоторые примеры, когда задачу можно преобразовать к вычислению свёртки двух последовательностей.

Поиск подстрок при помощи быстрого преобразования Фурье

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 научились решать 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) $p_0 \dots p_{m-1}$ в строке (тексте) $t_0 \dots t_{n-1}$, здесь p_i, t_j — это символы некоторого алфавита. Говорят, что подстрока входит с i-й позиции, если $p_j = t_{i+j}$, $j = 0, \dots, m-1$. 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 $\{B_i,\, i = 0, \dots, n-m\}$ позволяет определить все места вхождения подстроки в текст.

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

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

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

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 $O(m)$ шагов вычисляем сумму квадратов S и за $O(m)$ шагов считаем сумму первых m квадратов t_i, т. е. $H = \sum\limits_{j=0}^{m-1} t_j^2$. Далее вычисляем $B_0 = S - 2 \cdot c_{m-1} + H$ и $B_1 = S - 2 \cdot c_m + \sum\limits_{j=0}^{m-1} t_{j+1}^2 = B_1 + 2 \cdot c_{m-1} - 2 \cdot c_m - t_1^2 + t_m^2$.

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 $A_i = \sum\limits_{j=0}^{m-1} p_j t_{i+j}(t_{i+j} - p_j)^2 = \sum\limits_{j=0}^{m-1} (p_j^3 t_{i+j} - 2p_j^2 t_{i+j}^2 + p_j t_{i+j}^3)$. При этом символу «?» соответствует число 0, а все остальные символы кодируются ненулевыми числами. Тогда нетрудно заметить, что обнуляться полученная сумма будет тогда и только тогда, когда подстрока входит в слово с данной позиции. Вычисление отдельных частей суммы можно произвести при помощи БПФ, если правильно подобрать многочлены (по аналогии с предыдущим случаем, то есть так, чтобы коэффициенты произведения полученных многочленов выражались через отдельные части суммы A_i). Заметим, что для поиска подстроки с джокерами КМП-алгоритм не годился. Подобрать конкретные многочлены и обосновать корректность процедуры поиска подстроки с джокерами за $O(n \log n)$ вам предлагается в домашнем задании.

БПФ в $\mathbb{Z}_p$

Заметим, что вместо комплексных корней из единицы можно было бы рассматривать корни из единицы в некотором поле, например, в поле остатков по простому модулю p (если простой модуль достаточно большой, то лемма о сложении выполнена; остальные леммы, которыми мы пользовались при построении БПФ, очвеидно, тоже будут выполнены). Как мы знаем, в таком поле всегда есть образующий элемент (первообразный корень). Поэтому для любого простого p в поле остатков по модулю p есть корень g степени $p-1$ из единицы. Кроме того, если $p - 1 = m \cdot 2^k$, то g^m является корнем степени 2^k из единицы, а значит, можно применять точно тот же алгоритм БПФ[7] для многочленов степени 2^k. Более того, можно показать, что для любого натурального k такое простое число p найдётся. Это следует из так называемой теоремы Дирихле.

Теорема 3 (Теорема Дирихле)**.** Пусть a и b — два целых взаимно простых простых числа, т.е. НОД $(a, b) = 1$. Тогда существует бесконечно много простых чисел таких, что $p \equiv a \pmod b$.

Данный факт очень мощный, его доказательство выходит очень-очень далеко из нашей программы. Однако отсюда следует (если взять $a = 1, b = 2^k$), что для любого k существует бесконечно много простых чисел вида $p = m \cdot 2^k + 1$.

Рассмотрим следующее

Упражнение 1. Пусть есть n корзин. Часть из них содержит белые шары, а часть — чёрные шары. Про каждую корзину нам известно, какого цвета шар в ней лежит. Требуется за время $o(n^2)$ найти число троек $i < k < j$, что корзины с номерами i, j, k содержат белые шары, причём $j - k = k - i$, то есть они образуют арифметическую прогрессию длины 3, если дополнительно известно, что таких прогрессий не больше n.

Решение. Прямой перебор всех возможных троек, которые могут образовывать арифметическую прогрессию требует $\Omega(n^2)$ времени (подумайте, почему так; оцените число возможных арифметических прогрессий длины 3).

Рассмотрим последовательность $\{a_i\}$ такую, что $a_i = 1$, если в $i-$ корзине белый шар, и $a_i = 0$ — иначе. Рассмотрим индикаторную функцию

$$\delta(x) = \begin{cases} 1, x \equiv 0 \pmod p, \\ 0, x \not\equiv 0 \pmod p, \end{cases}$$

где p выбирается как $m \cdot 2^k + 1$, где k достаточно большое ($2^k \geq n$). Тогда нам нужно подсчитать $\sum\limits_{i=0}^{n-1} \sum\limits_{j=0}^{n-1} \sum\limits_{k=0}^{n-1} a_i a_j a_k \delta(i + j - 2k)$ по модулю p (здесь мы пользуемся, что $p > n$, что больше числа нужных нам прогрессий по условию). Будем считать, что $n = 2^k$ (если это не так, то дополним последовательность $\{a_k\}$ нулями). Тогда индикаторную функцию можно выразить в $\mathbb{Z}_p$ через корни из единицы: $\delta(x) = \frac{1}{n} \sum\limits_{k=0}^{n-1} \omega_n^{kx}$, где

[7] Или же, если есть 2 многочлена степени n, где $n \neq p - 1$ ни для какого простого числа p, то можно дополнить многочлены нулевыми старшими коэффициентами и получить многочлены степени $p - 1$. Постулат Бертрана гарантирует, что такое p будет не больше, чем $2n$. Однако, в таком случае уже нельзя будет применить тот же самый трюк, который мы использовали в БПФ. Поэтому мы сразу отбросили этот путь.

$\omega_n = g^m$, где g — первообразный корень по модулю p. Отсюда следует, что искомая сумма равна

$$\frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1}\sum_{t=0}^{n-1} a_i a_j a_k \omega_n^{t(i+j-2k)} = \frac{1}{n}\sum_{t=0}^{n-1}\left(\sum_{i=0}^{n-1} a_i \omega_n^{(-2t)i}\right)\left(\sum_{i=0}^{n-1} a_i \omega_n^{ti}\right)^2 = \frac{1}{n}\sum_{t=0}^{n-1} \left[\mathrm{DFT}_n(a)\right]_{(-2t)} \left[\mathrm{DFT}_n(a)\right]_t^2,$$

где нижний индекс означает в какой степени нужно взять ω_n. Таким образом, мы свели задачу к вычислению дискретного преобразования Фурье. Заметим, что все равенства в предыдущей формуле берутся по модулю p, именно поэтому важно, чтобы p был достаточно большим, чтобы ответ совпадал и с результатом, который бы мы получили, если бы проводили вычисления в $\mathbb{Z}$, а не в $\mathbb{Z}_p$. □

Циркулянтные матрицы. Системы линейных уравнений с циркулянтными матрицами.

Рассмотрим систему линейных уравнений с рациональными коэффициентами

$$Ax = b, \qquad A \in \mathbb{Q}^{n\times n}, b \in \mathbb{Q}^n.$$

Как мы знаем, если решать эту систему методом Гаусса, то потребуется $O(n^3)$ арифметических операций. Однако в некоторых случаях эту систему можно решить быстрее, то есть в тех случаях, когда матрицу A можно быстро обратить. Рассмотрим следующий красивый пример такой ситуации.

Будем называть матрицу A циркулянтной, если она имеет вид

$$A = \begin{pmatrix} a_0 & a_{n-1} & a_{n-2} & \dots & a_2 & a_1 \\ a_1 & a_0 & a_{n-1} & \dots & a_3 & a_2 \\ a_2 & a_1 & a_0 & \dots & a_4 & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-2} & a_{n-3} & a_{n-4} & \dots & a_0 & a_{n-1} \\ a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_1 & a_0 \end{pmatrix},$$

то есть A получается из столбца $a^\top = (a_0, a_1, \dots, a_{n_1})^\top$ следующим образом: первый столбец A — это a, второй столбец A — это циклический сдвиг a на 1 позицию (на 1 позицию вниз, если записать a как столбец) и так далее. Нетрудно показать (см., например, лекции Тыртышникова Е.Е., лекция 34), что собственными числами матрицы $A^\top$ (а значит, и мтарицы A) являются $\lambda_k = a_0 + \omega_n^k a_1 + \left(\omega_n^k\right)^2 a_2 + \ldots + \left(\omega_n^k\right)^{n-1} a_{n-1}$, при $k = 0, 1, 2, \dots, n-1$ с соответствующими собственными векторами $\left(1, \omega_n^k, \left(\omega_n^k\right)^2, \dots, \left(\omega_n^k\right)^{n-1}\right)^\top$. Тогда матрицу $A^\top$ можно разложить по базису из собственных векторов: $A^\top = V_n \Lambda V_n^{-1} = \frac{1}{n} V_n \Lambda V_n^*$, где $\Lambda = \mathrm{diag}(\lambda_0, \lambda_1, \dots, \lambda_{n-1})$ — диагональная матрица с элементами $\lambda_0, \lambda_1, \dots, \lambda_{n-1}$. Так как $V_n^\top = V_n$, то $A = V_n^{-1} \Lambda V_n = \frac{1}{n} V_n^* \Lambda V_n$. Поэтому

$$x = A^{-1} b = \left(V_n^{-1} \Lambda V_n\right)^{-1} b = V_n^{-1}\left(\Lambda^{-1}\left(V_n b\right)\right),$$

что можно посчитать за $O(n \log n)$, используя БПФ! Действительно, при помощи БПФ можно сначала найти $V_n b$ за $O(n \log n)$, затем за время $O(n)$ производится вычисление $\Lambda^{-1}\left(V_n b\right)$, т.к. перемножаются диагональная матрица и вектор длины n, а затем опять-таки при помощи БПФ вычисляется $V_n^{-1}\left(\Lambda^{-1}\left(V_n^{-1} b\right)\right)$ за $O(n \log n)$. Итоговая сложность — $O(n \log n)$ арифметических операций.