ПОЛИТЕХ
Санкт-Петербургский
политехнический университет
Петра Великого

Быстрое преобразование Фурье

Докладчик:
студент группы 3331506/70401
Паньков И.С.

Постановка задачи

Допустим, необходимо перемножить два многочлена $A(x)$ и $B(x)$

$$A(x) = a_0 x^0 + a_1 x^1 + \ldots + a_n x^n,$$

$$B(x) = b_0 x^0 + b_1 x^1 + \ldots + b_m x^m,$$

чтобы в результате получить многочлен $C(x)$

$$C(x) = A(x) \cdot B(x).$$

Как это сделать?

Поэлементное перемножение

$$C(x) = \left(c_0,\ c_1, \ldots,\ c_{m+n}\right):$$

$$\left.\begin{gathered}
c_0 = a_0 b_0, \\
c_1 = a_0 b_1 + a_1 b_0, \\
\vdots \\
c_k = a_u b_{k-v} + a_{u+1} b_{k-(u+1)} + \ldots + a_{k-v} b_v, \\
u = \max\{0,\ n-k\},\ v = \max\{0,\ m-k\}, \\
\vdots \\
c_{m+n} = a_n b_m.
\end{gathered}\right\}\ \begin{gathered} m \times n \\ \text{перемножений} \end{gathered}$$

При $m \approx n$ получаем временную сложность $O(n^2)$. А побыстрее?

Поэлементное перемножение

Давайте посмотрим внимательнее на все произведения:

$$c_0 = a_0 b_0 = \sum_{j=0}^{0} a_j b_{0-j},$$

$$c_1 = a_0 b_1 + a_1 b_0 = \sum_{j=0}^{1} a_j b_{1-j},$$

$$\vdots$$

$$c_k = a_u b_{k-v} + a_{u+1} b_{k-(u+1)} + \ldots + a_{k-v} b_v = \sum_{j=0}^{k} a_j b_{k-j}, \qquad \begin{array}{l} \{a_j\}_{j=n+1}^{m+n} = 0, \\ \{b_j\}_{j=m+1}^{m+n} = 0. \end{array}$$

$$u = \max\{0,\ n-k\},\ v = \max\{0,\ m-k\},$$

$$\vdots$$

$$c_{m+n} = a_n b_m = \sum_{j=0}^{m+n} a_j b_{m+n-j},$$

Это ничто иное как свёртка двух дискретных функций.

Свёртка

В частотной области свёртке двух дискретных функций соответствует произведение Фурье-образов функций в соответствующей точке.

Для нахождения $m+n$ свёрток требуется перемножить $m+n$ Фурье-образов функций, что требует временных затрат $O(n)$ при $m \approx n$.

Основная идея заключается в том, чтобы сначала применить к многочленам прямое преобразование Фурье, перемножить Фурье-образы, а затем применить обратное преобразование Фурье:

$$A(x) \cdot B(x) = \mathcal{F}^{-1}\left(\mathcal{F}(A(x)) \cdot \mathcal{F}(B(x))\right).$$

Но какие временные затраты требуют прямое и обратное преобразования Фурье?

Дискретное преобразование Фурье

Дискретное преобразование Фурье (англ. Discrete Fourier Transform, DFT) многочлена $A(x)$ — это вектор значений этого многочлена в точках

$$x = \omega_{n,k} : \omega_{n,k} = e^{i 2\pi k/n} = \omega_n^k, \ \omega_n = \omega_{n,1} = e^{-i 2\pi/n}.$$

Сравним различные представления прямого преобразования Фурье:

$$X_k = \mathcal{F}(\{x_k\}) = \sum_{j=0}^{n-1} x_j e^{i \frac{2\pi k}{n} j},$$

$$f_k = \mathcal{F}(A(x)) = A(e^{i 2\pi k/n}) =$$

$$= a_0 \left(e^{i 2\pi k/n}\right)^0 + a_1 \left(e^{i 2\pi k/n}\right)^1 + \ldots + a_n \left(e^{i 2\pi k/n}\right)^n = \sum_{j=0}^{n-1} a_j e^{i \frac{2\pi k}{n} j}.$$

Дискретное преобразование Фурье

Дискретное преобразование Фурье (ДПФ) можно также записать в матричном виде:

$$\begin{pmatrix} \omega_n^0 & \omega_n^0 & \omega_n^0 & \cdots & \omega_n^0 \\ \omega_n^0 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{(n-1)} \\ \omega_n^0 & \omega_n^2 & \omega_n^4 & \cdots & \omega_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_n^0 & \omega_n^{(n-1)} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{pmatrix}.$$

Необходимо определить элементы матрицы размерности $n \times n$, а затем скалярно умножить каждую строку матрицы на каждый столбец вектора, то есть имеем задачу с временной сложностью $O(n^2)$. А быстрее?

Быстрое преобразование Фурье

Будем считать, что имеем многочлен $A(x)$ порядка $n-1$, где $n=2^t,\ t\in\mathbb{N}$, а иначе дополним его нулевыми коэффициентами. Разложим многочлен на два других следующим образом:

$$A_0(x)=a_0x^0+a_2x^1+\ldots+a_{n-2}x^{n/2-1},$$

$$A_1(x)=a_1x^0+a_3x^1+\ldots+a_{n-1}x^{n/2-1}.$$

Тогда исходный многочлен может быть получен как

$$A(x)=A_0(x^2)+xA_1(x^2).$$

Быстрое преобразование Фурье

Допустим, известны Фурье-образы многочленов $A_0(x)$ и $A_1(x)$

$$\mathcal{F}(A_0(x)) = \{f_{0k}\}_{k=0}^{n/2-1},$$
$$\mathcal{F}(A_1(x)) = \{f_{1k}\}_{k=0}^{n/2-1}.$$

Для первой половины Фурье-образов получаем

$$f_k = A(\omega_n^k) = A_0(\omega_n^{2k}) + \omega_n^k A_1(\omega_n^{2k}) = f_{0k} + \omega_n^k f_{1k},\ k = \overline{0, n/2-1}.$$

Для второй половины Фурье-образов получаем

$$f_{k+n/2} = A(\omega_n^{k+n/2}) = A_0(\omega_n^{2k+n}) + \omega_n^{k+n/2} A_1(\omega_n^{2k+n}) =$$
$$= A_0(\omega_n^{2k}\omega_n^n) + \omega_n^k \omega_n^{n/2} A_1(\omega_n^{2k}\omega_n^n) = f_{0k} - \omega_n^k f_{1k},\ k = \overline{0, n/2-1}.$$

Быстрое преобразование Фурье

Итак, мы получили простые формулы:

$$\begin{aligned} f_k &= f_{0k} + \omega_n^k f_{1k} \\ f_{k+n/2} &= f_{0k} - \omega_n^k f_{1k} \end{aligned}, \quad k = \overline{0, n/2-1}.$$

Но что делать, если мы ещё не вычислили Фурье-образы для многочленов $A_0(x)$ и $A_1(x)$? – Вычислить!

Но как? – Разбить каждый многочлен на ещё два многочлена и использовать их Фурье-образы!

А если и они неизвестны? – Разбить снова! Получаем *рекурсию*.

Это известная парадигма «Разделяй и властвуй». Как правило, такие алгоритмы имеют временную сложность $O(n \log n)$.

Обратное преобразование Фурье

Снова запишем прямое преобразование Фурье в матричном виде:

$$\begin{pmatrix} \omega_n^{0\cdot 0} & \omega_n^{0\cdot 1} & \omega_n^{0\cdot 2} & \dots & \omega_n^{0\cdot (n-1)} \\ \omega_n^{1\cdot 0} & \omega_n^{1\cdot 1} & \omega_n^{1\cdot 2} & \dots & \omega_n^{1\cdot (n-1)} \\ \omega_n^{2\cdot 0} & \omega_n^{2\cdot 1} & \omega_n^{2\cdot 2} & \dots & \omega_n^{2\cdot (n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_n^{(n-1)\cdot 0} & \omega_n^{(n-1)\cdot 1} & \omega_n^{(n-1)\cdot 2} & \dots & \omega_n^{(n-1)\cdot (n-1)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{pmatrix}.$$

Можно домножить левую и правую части на обратную к стоящей слева матрицу и по Фурье-образу найти коэффициенты многочлена.

Обратное преобразование Фурье

Это матрица Вандермонда, которая задаётся следующим соотношением:

$$F = \left(F_{jk}\right) = \left(f_k(\omega_{n,\, j-1})\right): \quad f_k(\alpha_j) = \alpha_j^{k-1}, \quad j, k = \overline{1, n};$$

Для неё справедливо

$$F = \left(\omega_n^{jk}\right)_{j,\,k=0}^{n-1}, \quad \omega_n = e^{i\,2\pi/n};$$

$$F^{-1} = \frac{1}{n}\left(\bar{\omega}_n^{jk}\right)_{j,\,k=0}^{n-1} = \frac{1}{n}F^*, \quad \bar{\omega}_n = e^{-i\,2\pi/n},$$

где F^* – эрмитово-сопряжённая матрица.

Обратное преобразование Фурье

Получаем следующее выражение:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{pmatrix} = \frac{1}{n} \begin{pmatrix} \omega_n^0 & \omega_n^0 & \omega_n^0 & \cdots & \omega_n^0 \\ \omega_n^0 & \omega_n^{-1} & \omega_n^{-2} & \cdots & \omega_n^{-(n-1)} \\ \omega_n^0 & \omega_n^{-2} & \omega_n^{-4} & \cdots & \omega_n^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_n^0 & \omega_n^{-(n-1)} & \omega_n^{-2(n-1)} & \cdots & \omega_n^{-(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{pmatrix},$$

или

$$a_k = \frac{1}{n} \mathcal{F}^{-1}(F^*(x)\, f) = \frac{1}{n} A(e^{-i\, 2\pi k/n}) = \frac{1}{n} \sum_{j=0}^{n-1} f_j e^{-i\,\frac{2\pi k}{n} j}.$$

Задача свелась к ранее решённой с временной сложностью $O(n \log n)$.

Пример

Пусть дан многочлен третьего порядка:

$$A(x) = 5 + 2x + 4x^2 - x^3.$$

Разобьём его на два многочлена:

$$A_0(x) = 5 + 4x, \qquad A_1(x) = 2 - x,$$

$$A(x) = A_0(x^2) + xA_1(x^2) = 5 + 4x^2 + x\left(2 - x^2\right) = 5 + 2x + 4x^2 - x^3.$$

Вспомним формулу для быстрого преобразования Фурье и найдём Фурье-образы многочленов:

$$\begin{matrix} f_k = f_{0k} + \omega_n^k f_{1k} \\ f_{k+n/2} = f_{0k} - \omega_n^k f_{1k} \end{matrix}, \quad k = \overline{0, n/2-1}.$$

$$f_{00} = 5 + 4 = 9, \qquad f_{10} = 2 - 1 = 1,$$

$$f_{01} = 5 - 4 = 1, \qquad f_{11} = 2 + 1 = 3.$$

Пример

Вновь применим быстрое преобразование Фурье, но уже для исходного многочлена:

$$f_0 = f_{00} + \sqrt[4]{1}^{\,0} f_{10} = f_{00} + f_{10} = 9 + 1 = 10,$$

$$f_1 = f_{01} + \sqrt[4]{1}^{\,1} f_{11} = f_{01} + i\, f_{11} = 1 + 3i,$$

$$f_2 = f_{00} - \sqrt[4]{1}^{\,0} f_{10} = f_{00} - f_{10} = 9 - 1 = 8,$$

$$f_3 = f_{01} - \sqrt[4]{1}^{\,1} f_{11} = f_{01} - i\, f_{11} = 1 - 3i.$$

Итого:

$$\{f_k\} = \mathcal{F}(A(x)) = (10,\ 1 + 3i,\ 8,\ 1 - 3i).$$

Пример

Теперь восстановим коэффициенты многочлена по его Фурье-образу. Имеем многочлен с комплексными коэффициентами:

$$B(x)=10+\left(1+3i\right)x+8x^2+\left(1-3i\right)x^3.$$

Разобьём его на два многочлена:

$$B_0(x)=10+8x, \qquad B_1(x)=\left(1+3i\right)+\left(1-3i\right)x.$$

$$B(x)=B_0(x^2)+xB_1(x^2)=10+8x^2+x\left(\left(1+3i\right)+\left(1-3i\right)x^2\right)=$$
$$=10+\left(1+3i\right)x+8x^2+\left(1-3i\right)x^3.$$

Найдём коэффициенты многочленов:

$$\begin{matrix} a_k = a_{0k}+\omega_n^{-k}a_{1k} \\ a_{k+n/2}=a_{0k}-\omega_n^{-k}a_{1k} \end{matrix}, \ k=\overline{0, n/2-1}. \qquad \begin{matrix} a_{00}=10+8=18, & a_{10}=1+3i+1-3i=2, \\ a_{01}=10-8=2, & a_{11}=1+3i-1+3i=6i. \end{matrix}$$

Пример

А теперь найдём коэффициенты исходного многочлена:

$$a_0 = a_{00} + \sqrt[4]{1}^{-0} a_{10} = a_{00} + a_{10} = 18 + 2 = 20,$$

$$a_1 = a_{10} + \sqrt[4]{1}^{-1} a_{11} = a_{10} - i\, a_{11} = 2 - i \cdot 6i = 8,$$

$$a_2 = a_{00} - \sqrt[4]{1}^{-0} a_{11} = a_{00} - a_{10} = 18 - 2 = 16,$$

$$a_3 = a_{10} - \sqrt[4]{1}^{-1} a_{11} = a_{10} + i\, a_{11} = 2 + i \cdot 6i = -4.$$

Итого:

$$A(x) = \frac{1}{4}\mathcal{F}^{-1}(\{ f_k \}) = \frac{1}{4}\mathcal{F}^{-1}(B(x)) = (5,\ 2,\ 4,\ -1).$$

Реализация

```
typedef complex<double> base;

void fft(vector<base> &a, bool isInvert)
{
    int n = a.size();
    if (n == 1) return;

    vector<base> a0(n / 2), a1(n / 2);                          // Дополнительные многочлены A0 и A1
    for (int i = 0; i < n / 2; i++)
    {
        a0[i] = a[2 * i];                                       // Заполняем многочлен A0 чётными коэффициентами
        a1[i] = a[2 * i + 1];                                   // Заполняем многочлен A1 нечётными коэффициентами
    }
    fft(a0, invert);                                            // Рекурсивно выполняем алгоритм для многочлена A0
    fft(a1, invert);                                            // Рекурсивно выполняем алгоритм для многочлена A1

    double arg = 2 * M_PI / n * (isInvert ? -1 : 1);            // Вычисляем аргумент главного значения корня n-ой степени
    base w(1), wn(cos(arg), sin(arg));                          // Вычисляем главный корень n-ой степени
    for (int i = 0; i < n / 2; i++)                             // Вычисляем Фурье-образ/коэффициенты исходного многочлена
    {
        a[i]         = (a0[i] + w * a1[i]) / (isInvert ? 2 : 1);
        a[i + n / 2] = (a0[i] - w * a1[i]) / (isInvert ? 2 : 1);
        w *= wn;
    }
}
```

Список использованных источников

1. MAXimal::algo::Быстрое преобразование Фурье за *O*(*N log N*): https://e-maxx.ru/algo/fft_multiply
2. Discrete Fourier transform: https://en.wikipedia.org/wiki/Discrete_Fourier_transform
3. Матрица и определитель Вандермонда: http://pmpu.ru/vf4/algebra2/vander

Спасибо за внимание!