FFT

Филипп Грибов

27.10.2018

1 Комплексные числа

1.1 Определение

Комплексное число — это пара из двух действительных чисел, вида (x, y), где x — *действительная часть*, а y — *мнимая часть*. Комплексные числа удобно представлять на плоскости в виде вектора, идущего к точчке с координатами (x, y). Ещё один популярные способ определять комплесные числа — записывать комплексное число (x, y) в виде $x + i \cdot y$, где i называется *мнимой еденицей*. Ообенность мнимой еденицы в том, что $i \cdot i = -1$

1.2 Операции

Из второго определения комплексных чисел можно легко вывести всевозможные арифметические операции с комплесными числами.

$(x_1 + i \cdot y_1) + (x_2 + i \cdot y_2) = (x_1 + x_2) + i \cdot (y_1 + y_2)$
$(x_1 + i \cdot y_1) - (x_2 + i \cdot y_2) = (x_1 - x_2) + i \cdot (y_1 - y_2)$
$(x_1 + i \cdot y_1) \cdot (x_2 + i \cdot y_2) = (x_1 \cdot x_2 - y_1 \cdot y_2) + i \cdot (x_1 \cdot y_2 + x_2 \cdot y_1)$

Число $\overline{z}$ называется сопряженным числом z и имеет формулу $x - i \cdot y$. Легко заметить, что $(x + i \cdot y) \cdot (x - i \cdot y) = x^2 + y^2$. Тогда из этого можно вывести формулу деления.

$$\frac{x_1 + i \cdot y_1}{x_2 + i \cdot y_2} = \frac{(x_1 + i \cdot y_1) \cdot (x_2 - i \cdot y_2)}{(x_2 + i \cdot y_2) \cdot (x_2 - i \cdot y_2)} = \frac{(x_1 + i \cdot y_1) \cdot (x_2 - i \cdot y_2)}{x_2^2 + y_2^2} = \frac{x_1 \cdot x_2 + y_1 \cdot y_2}{x_2^2 + y_2^2} + i \cdot \frac{y_1 \cdot x_2 - x_1 \cdot y_2}{x_2^2 + y_2^2}$$

Вернёмся к первому определению и посмотрим, что означают опрерации с комплексными числами в геомерическом смысле.

$(x_1, y_1) + (x_2, y_2) = (x_1 + x_2,\ y_1 + y_2)$
$(x_1, y_1) - (x_2, y_2) = (x_1 - x_2,\ y_1 - y_2)$

Тогда сумма и разность двух комплексных чисел соответствует сумме и разности векторов, соответствующих им.

$(x_1, y_1) \cdot (x_2, y_2) = (x_1 \cdot x_2 - y_1 \cdot y_2,\ x_1 \cdot y_2 + x_2 \cdot y_1)$

Замечаем, что x координата вектора это длина вектора умноженная на косинус угла вектора, а y координата вектора это длина вектора умноженная на синус угла вектора. Т.е. $x = L \cdot \cos\alpha$, $y = L \cdot \sin\alpha$. Тогда $x_1 \cdot x_2 - y_1 \cdot y_2 = L_1 \cdot L_2 \cdot (\cos\alpha_1 \cdot \cos\alpha_2 - \sin\alpha_2 \cdot \sin\alpha_2) = L_1 \cdot L_2 \cdot \cos(\alpha_1 + \alpha_2)$ (По формуле двойного угла). Аналогично $x_1 \cdot y_2 + x_2 \cdot y_1 = L_1 \cdot L_2 \cdot \cos(\alpha_1 + \alpha_2)$. Тогда при произведении двух комплексных чисел результат в векторном виде имеет длину, равную произведению двух векторов, соответсвующих множителям, а угол этого вектора равен сумме углов, соответвующих множителям.

Тоже самое можно вывести для деления, оставим это читателю в качестве упражнения.

1.3 Применения

У комплексных чисел есть много применений, их удобно использовать для геометрии и для многих других структур, но сегодня речь не об этом

2 Быстрое преобразование Фурье

Если здесь вы что-то не поймёте, почитайте на e-maxx, там про Фурье написано довольно неплохо.

2.1 Умножение многочленов

Умножать многочлены — это круто, но долго. Самый тривиальный алгоритм делает это за $O(n^2)$. Есть конечно Карацуба, который справляется с этим чуть быстрее, но тоже не идеально. Но есть алгоритм быстрого преобразования фурье, с помощью которого это достигается за $O(n \log n)$.

В классическом понимании для умножения многочленов надо перемножить их коэффиценты. Но можно подойти к этому с другой стороны. Посчитаем значения многочленов в $2n$ различных точках. Замечаем, что значение их произведения в каждой из этих точек будет равно произведению их значений в каждой из этих точек. А т.к. различные многочлены степени n имеют различные значения в $n+1$ точке, то если по значения многочлена в n точках мы сможем восстановить многочлен, то так мы найдём произведение.

2.2 Прямое преобразование Фурье

Надо найти значения многочлене степени n в n точках. Для удобства увеличим n так, чтобы n стал точной степенью 2.

Пусть есть многослен $A = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_{n-1} \cdot x^{n-1}$. Заметим, что

$$A = (a_0 + a_2 \cdot x^2 + x_4 \cdot x^4 + \ldots + a_{n-2} \cdot x^{n-2}) + x \cdot (a_1 + a_3 \cdot x^2 + \ldots + a_{n-1} \cdot x^{n-2})$$

Мы хотм посчитать его значения в $x_0, x_1, \ldots x_{n-1}$. Тогда заметим, что если бы множество из $x_0^2, x_1^2, \ldots x_{n-1}^2$ состоядо бы из $n/2$ чисел, то мы бы посчитали бы за $O(n/2 \log(n/2))$ значения $(a_0 + a_2 \cdot x^2 + x_4 \cdot x^4 + \ldots + a_{n-2} \cdot x^{n-2})$ в этих $n/2$ точках, далее за $O(n/2 \log(n/2))$ посчитали бы значения $(a_1 + a_3 \cdot x^2 + \ldots + a_{n-1} \cdot x^{n-2})$ в этих же $n/2$ точках и в конце за $O(n)$ построили бы нормальный ответ.

Такие $x_0, x_1, \ldots, x_{n-1}$ существуют. Для этого нам потребуется первообразный $\sqrt[n]{1}$. Первообразным корнем n-й степени из 1 называется такое число, что оно во всех степенях меньших или равных n принимает разные значения, а в степени n оно равно 1. Вспомниим комплексные числа. Возьмём число $z = (\cos(\frac{2 \cdot \pi}{n}), \sin(\frac{2 \cdot \pi}{n}))$. Замечаем что z соответствует вектору, имеющему длину 1 и угол $\frac{2 \cdot \pi}{n}$. Тогда z^k соответствует вектору, имеющему длину 1 и угол $k \cdot \frac{2 \cdot \pi}{n}$. Тогда z^n равен 1 и все числа $z^1, z^2, \ldots z^n$ различны.

Тогда возьмём $x_0 = z^0, x_1 = z^1, x_2 = z^2, \ldots, x_{n-1} = z^{n-1}$. Замечаем, что $x_i^2 = z^{2i}$. Тогда угол вектора, соответствующего x_i делится на $2 \cdot \frac{2 \cdot \pi}{n}$. А таких углов всего $n/2$.

Тогда быстрое преобразование устроенно так:

Сначала многочлен делится на два многочлена размера $n/2$.

$$A = (a_0 + a_2 \cdot x^2 + x_4 \cdot x^4 + \ldots + a_{n-2} \cdot x^{n-2}) + x \cdot (a_1 + a_3 \cdot x^2 + \ldots + a_{n-1} \cdot x^{n-2})$$

Для каждого из них считаются значения в разных **чётных** степенях $z = (\cos(\frac{2 \cdot \pi}{n}), \sin(\frac{2 \cdot \pi}{n}))$. После чего на основе этого считаются значения A во всез степенях z.

2.3 Обратное преобразование Фурье

Нам надо по значениям многочлена степени $n-1$ в n различных точках, равных степеням $z = (\cos(\frac{2 \cdot \pi}{n}), \sin(\frac{2 \cdot \pi}{n}))$, восстановить многочлен.

Будем использовать прямое FFT к значения многочлена, но вместо z возьмём $\overline{z} = (\cos(-\frac{2 \cdot \pi}{n}), \sin(-\frac{2 \cdot \pi}{n}))$. В конце каждый член надо будет поделить на n. Так мы получим ответ.

Доказательство правельности этого можете загуглить.

2.4 Итог

Тогда так мы можем считать произведение двух многочленов за $O(n \log n)$. Разумеется делить такм образом не получится. Ниже описан алгоритм быстрого деления.

3 Быстрое деление двух чисел

У нас будут 2 функции, рекурсивно вызывающие друг друга. n всегда будет являться точной степенью двух, $|a|$ будет обозначать длину числа а.

1. div21 — делит число a длиной не более $2n$ на число b длиной не более n, при условии, что ответ по длине не превосходит n.

2. div32 — делит число a длиной не более $3n$ на число b длиной не более $2n$, при условии, что ответ по длине не превосходит n.

Как работает div21: пусть $m = \frac{n}{2}$. Тогда a имеет длину не более $4m$, на b имеет длину не более $2m$, а ответ по длине не превосходит $2m$. Возьмём первые $|a| - m \le 3m$ цифр числа a и разделим на число b при помощи div32. Ответ по длине не превзойдёт m, так как иначе a/b было бы больше по длине чем $2m$. Далее возьмём остаток после этого деления, домножим его на 10^m, и припишем к нему последние m цифр числа a. После этого разделим это число на b. Так как $|b| \le 2m$, то длина остатка не превосходит $2m$, значит длина остатка, домноженного на 10^m на превосходит m, тогда для деления мы можем использовать функцию div32. далее умножим результат первого деления на m и прибавил результат второго деления. Так мы получим ответ.

Как работает div32: пусть $|b| \le n$. Тогда так как длина ответа не превосходит n, $|a| \le 2n$, тогда будем использовать $div21$. Иначе возьмём b_1, образованное первыми n цифрами из b. Возьмём a_1, образованное первыми $|a|-(|b|-n)$ цифрами из a. (Т.е. обрежем a и b на одинаковое число цифр в конце так, чтобы длина b стала равна n) Обозначим за $k = |b|-n$. (т.е. сколько цифр в конце мы отрезали от a и от b). Обозначим за $r_a = a-a_1 \cdot 10^k$, $r_b = b-b_1 \cdot 10^k$. (т.е. обозначим за r_a и r_b то, что мы отрезали от a и от b). Тогда $a/b = (a_1 \cdot 10^k + r_a)/(b_1 \cdot 10^k + r_b)$. Тогда $a_1/b_1 \ge a/b$ При этом замечаем, что $b < (b_1+1) \cdot 10^k$. Тогда $a/b > a/((b_1+1) \cdot 10^k)$. А т.к. $|a_r| \le n$, то $a_r/b < 1$. Тогда $a_1/b_1 \ge a/b \ge a_1/(b_1+1) - 1$. Возьмём $q = a_1/b_1$. Возьмём $(q-10) \cdot (b_1+1) = (a_1/b_1 - 10) \cdot (b_1+1) = a_1 - 10 \cdot b_1 + q - 10$. Т.к. $q = a_1/b_1$ по длине не превышает n, а b по длине ровно равно n, то $10b > q$. Тогда $(q-10) \cdot (b_1+1) < a_1$. Тогда $q - 10 < a_1/(b_1+1)$. Тогда $a_1/b_1 \ge a/b \ge a_1/b_1 - 10$. Тогда для div32 получается следующий алгоритм. Строим числа a_1 и b_1. С помощью div21 разделим a_1 на b_1. (Если $a_1 > b_1 \cdot 10^n$, будем считать что результат равен $10^n - 1$). Далее вычтем из a результат деления, домноженный на b и посчитаем остаток от текущего деления. Пока остаток меньше нуля, будем прибавлять к нему b и вычитать из частного 1. Таких действий мы сделаем не более n.

При совсем маленьких числах надо делить уже используя int-овое деление.

Можно увеличтить базу с 10 до 10^9, но делать это надо аккуратно.

Асимптотику доказать несложно. $div21(n)$ делает $O(n)$ действий и вызывает div32 два раза. $div32(n)$ делает $O(n \log n)$ действий (т.к. там есть операция умножения) и возможно вызывает $div21(n)$. Таким образом $div21(n)$ делает $O(n \log n)$ действий и два раза вызывает div21. Таким образом время работы $O(n \log^2 n)$.

Более подробное писание и доказательство этого алгоритма можно найти по ссылке
`http://cr.yp.to/bib/1998/burnikel.ps`

4 Быстрый корень

Для начала рассмотрим алгоритм поиска корня k степени за $O(n \log^3 n)$ с огромной константой. Будем использовать Ньютоновский метод поиска нуля функции $f(x)$. Он работает итерациями — есть начальное число x_0. Далее $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$. Каждая следующая итерация приближает x_n к нулю функции. Тогда если мы возьмём функцию $f(x) = a - x^k$, то максимальный целый x, при котором функция положительна и есть ответ. Тогда если за x_0 взять $10^{n/k}$, то кол-во итераций будет $O(\log n)$ с большой константой, а в каждой производится операция деления, т.е. асимптотика — $O(nlog^3n)$.

Это был общеизвестный алгоритм поиска корня который везде используется. Он не самый быстрый. А теперь будет описан другой алгоритм, который работает за $O(nlog^2n + klog^3k)$ с малой константой. Будем искать корень k степени из x рекурсивно. Но сначала чуть чуть математики.

Пусть $\sqrt[k]{x} = a10^m + b$, где $b < 10^m$. Тогда

$$x = a^k 10^{km} + ka^{k-1}b10^{(k-1)m} + \frac{k(k-1)}{2}a^{k-2}b^2 10^{(k-2)m} + ... + b^k$$

Тогда если $a > b$ и $10^m > k$ то

$$\frac{k(k-1)}{2}a^{k-2}b^2 10^{(k-2)m} > \frac{k(k-1)(k-2)}{1 \cdot 2 \cdot 3}a^{k-3}b^3 10^{(k-3)m} > ... > b^k$$

Тогда

$$k\left(\frac{k(k-1)}{2}a^{k-2}b^2 10^{(k-2)m}\right) > \frac{k(k-1)}{2}a^{k-2}b^2 10^{(k-2)m} + \frac{k(k-1)(k-2)}{1 \cdot 2 \cdot 3}a^{k-3}b^3 10^{(k-3)m} + ... + b^k$$

Тогда если $b < 10^m < a$ и $10^m > k^2$ то

$$a^{k-1} \cdot 10^{(k-1)m} > k\left(\frac{k(k-1)}{2}a^{k-2}b^2 10^{(k-2)m}\right) > \frac{k(k-1)}{2}a^{k-2}b^2 10^{(k-2)m} + \frac{k(k-1)(k-2)}{1 \cdot 2 \cdot 3}a^{k-3}b^3 10^{(k-3)m} + ... + b^k$$

А из этого следует уже не такой ужасный алгоритм.

Пусть n - длина числа x, из которого мы ищем корень. Пусть $m = \frac{n-1}{2k}$. Тогда если $10^m \le k^2$, то ищем корень методом Ньютона. Иначе возьмём число x без последних km цифр. Пусть a - корень из этого числа. Тогда замечаем, что $\sqrt[k]{x}a10^m + b$, где $b < 10^m$. При этом $b < 10^m < a$ и $10^m > k^2$. Тогда верно последнее большое неравенство, описанное выше. Тогда $b = (x - a^k 10^{km})/(a^{k-1}10^{(k-1)m})$ или на единицу меньше. Тогда совершив такое деление и проверив, надо ли вычесть 1 из b мы найдём $sqrt[k]x$. Замечаем, что самое большое по асимптотике действие, которое мы сделаем — деление, а далее вызовемся от числа, в 2 раза меньшего нас по длине. Тогда так как в конце нам ещё придётся искать корень методом Ньютона из числа, по длине примерно равному k, то всего асимптотика — $O(nlog^2n + klog^3k)$.