$[1, n]$ в $[1, n]$, при этом коэффициент μ_α определяется суммой

$$\mu_\alpha = \sum_{\epsilon_i \in \{0,1\}} (-1)^{\epsilon_1 + \ldots + \epsilon_n} \epsilon_{\alpha(1)} \ldots \epsilon_{\alpha(n)}.$$

Если α — перестановка, коэффициент μ_α равен $(-1)^n$, поскольку единственный ненулевой член суммы — это тот, для которого все ϵ_i, равны 1. В противном случае можно разложить μ_α следующим образом:

$$\sum_{\epsilon_i, i \neq k} (-1)^{\epsilon_1 + \ldots + \epsilon_n} \sum_{\epsilon_k = 0,1} (-1)^{\epsilon_k} \epsilon_{\alpha(1)} \ldots \epsilon_{\alpha(n)},$$

где k — элемент из $[1, n]$, который не принадлежит образу α, и хорошо видно, что внутренний член (сумма по ϵ_k) равен нулю; это доказывает, что μ_α в этом случае равно нулю. Второе искомое выражение (суммы которого записываются на множествах) просто выводится из первого, исключением ненулевых ϵ_i, появляющихся в сумме.

b. Возвратный код Грея из n разрядов является выражением линейного порядка на множестве всех частей E из $[1, n]$, наименьший элемент которого есть $\emptyset$ и наибольший, E_{max}, исходное множество. Если E' является последователем для E, то симметрическая разность $E \Delta E'$ сводится к одному элементу, а это означает, что E' получается, исходя из E, добавлением или удалением одного элемента. Введем несколько обозначений: $E' = \mathrm{succ}(E)$, $\sum_E = \sum_{F \leqslant E} (-1)^{|F|} \prod_{1 \leqslant i \leqslant n} S_i(F)$, где $S_i(E) = \sum_{j \in E} a_{ij}$. Вычисляемый перманент является, значит, суммой $(-1)^n \sum_{E_{max}}$. Эта сумма получается вычислением последовательных значений $\sum_E$ и, тогда, итерируя, имеем:

// here goes some code

$$\sum_{E'} = \sum_E + (-1)^{|E'|} \prod_{1 \leqslant i \leqslant n} S_i(E'). \tag{7}$$

Вклад наименьшего элемента $\emptyset$ в эту сумму — нулевой; значит, начнем с его последователя, который может быть $\{1\}$, $\{n\}$ или какое-нибудь одноэлементное множество в соответствии с порядком, заданным на $[1, n]$. Можно заметить, что $S_i(E') = S_i(E) \pm a_{ij}$, где $\{j\} = E \triangle E'$.

В этой записи, как в алгоритме 12, $\pm$ должен пониматься как $+$, если $E \subset E'$ и $-$, если $E' \subset E$.

Чтобы оценить $\sum_{E'}$, исходя из $\sum_E$ с использованием соотношения (7), нужно осуществить n сложений (для подсчета каждого $S_i(E')$), затем $n-1$ перемножений: $S_1(E') \times \cdots \times S_n(E')$, и, наконец, 1 сложение. Имеем $2^n - 2$ операций для осуществления (7), при этом первый член $\sum = -\prod_{1 \leqslant i \leqslant n} \alpha_{in}$ требует $n-1$ перемножений; это доказывает сформулированный результат о сложности. Сложность $O(n2^n)$ значительна, но остается того же порядка, что и сложность, индуцированная определением ($n!(n-1)$ перемножений и $n!-1$ сложений). Формула Стирлинга позволяет сравнить эти два значения сложности: $n \cdot n!/(n \cdot 2^n) \approx (n/2e)^n \sqrt{2\pi n}$.

22. Перманент матрицы (продолжение)

a. Правая часть может рассматриваться как многочлен (от переменных a_{ij}), равный $\sum_\alpha \mu_\alpha a_{1\alpha(1)} \ldots a_{n\alpha(n)}$, где сумма распространяется на все отображения $[1, n]$ в $[1, n]$. Коэффициент μ_α задан формулой

$$\mu_\alpha = \sum_\omega \mu_\alpha(\omega) \quad \text{с} \quad \mu_\alpha(\omega) = \omega_1 \ldots \omega_{n-1}\omega_{\alpha(1)} \ldots \omega_{\alpha(n)},$$

в которой полагаем $\omega_n = 1$. Если α — перестановка, то каждое $\mu_\alpha(\omega)$, присутствующее в сумме μ_α, равно 1 и, следовательно, $\mu_\alpha = 2^{n-1}$. Напротив, если α не является перестановкой, то сумма μ_α — нулевая. Действительно, образ α отличен от $[1, n]$, и различаем два случая:

(i) $\exists\, k < n$, не принадлежащий образу α,

(ii) $\exists\, k < n$, дважды полученный из α.

В обоих случаях члены $\mu_\alpha(\omega)$, присутствующие в сумме, группируются попарно, один соответствуя $\omega_k = 1$, другой — $\omega_k = -1$, и взаимно уничтожаются (в случае (ii) $\mu_\alpha(\omega) = \omega_k$).

b. Формула пункта **a** может быть записана в следующем виде:

$$\frac{per A}{2} = \sum \omega_1 \ldots \omega_{n-1} \prod_{1 \leqslant i \leqslant n} (a_{in} + \omega_1 a_{i1} + \cdots + \omega_{n-1} a_{in-1})/2.$$

Как и в предыдущем упражнении, вычисление перманента получается генерированием перебора при линейной упорядоченности на $\{-1, 1\}^{n-1}$, в которой два последовательных элемента отличаются только одной компонентой. Если для $\omega \in \{-1, 1\}^{n-1}$ и $i \leqslant n$ положить

$$S_i(\omega) = (a_{in} + \omega_1 a_{i1} + \cdots + \omega_{n-1} a_{in-1})/2,$$

то можно, благодаря перебору на $\{-1,1\}^{n-1}$, вычислить последовательно $\sum_{\omega} = \sum_{\rho \leqslant \omega} \ldots$, используя формулу

$$\sum_{\omega'} = \sum_{\omega} + \omega'_1 \ldots \omega'_{n-1} \prod_{1 \leqslant i \leqslant n} S_i(\omega'), \tag{8}$$

где ω' — последователь для ω в $\{-1,1\}^{n-1}$.

Если j является индексом, по которому различаются два слова ω и ω', то сумма $S_i(\omega')$ вычисляется, исходя из $S_i(\omega)$, через $S_i(\omega') = S_i(\omega) - a_{ij}$, если $\omega_j = 1$, и через $S_i(\omega') = S_i(\omega) + a_{ij}$, если $\omega_j = -1$.

// here goes some code

Алгоритм 13. Вычисление перманента в кольце, где 2 обратимо

С практической точки зрения для генерации адекватного перебора $\{-1,1\}^{n-1}$, выбираем соответствие между $\{0,1\}$ и $\{-1,1\}$ вида $\epsilon \mapsto (-1)^{\epsilon}$ и классическую генерацию кода Грея на $\{0,1\}^{n-1}$, что достигается с помощью алгоритма 13. Мультипликативная сложность получается, если заметить, что нужно вычислить $2^n - 1$ членов $\sum_{\omega}$, каждый из которых требует $n-1$ перемножений (см. формулу (8)), значит, всего $2^{n-1}(n-1)$ произведений, к которым нужно добавить последнее умножение на 2. С точки зрения сложений первоначальный член $\sum_{(1,\ldots,1)}$ требует $n(n-1)$ сложений и n делений на 2, тогда как общий член $\sum_{\omega}$ вычисляется, исходя из предыдущего, с помощью n сложений; наконец, нужно сложить все эти члены, что требует в целом $n(n-1) + (2^{n-1} - 1)(n+1)$ сложений и n делений на 2. Заметим относительно предыдущего упражнения, что сложность была приблизительно разделена на 2.

c. Рассмотрения полностью аналогичны предыдущему пункту, если только невозможно деление на 2; деление (точное) на 2^{n-1} будет иметь место уже в конце. Результатом является алгоритм 14.

// here goes some code

23. Массив инверсий подстановки

d. Массив инверсий перестановки α имеет вид (0, 0, 0, 1, 4, 2, 1, 5, 7). Свойство $0 \leqslant \alpha_k < k$ легко получается из того, что имеется точно $k-1$ целых чисел, заключенных строго между 0 и k. Массив инверсий возрастающей перестановки интервала [1, n] есть, очевидно, (0, 0, ... , 0), и таблица для единственной убывающей перестановки — (0, 1, 2, ... , n).

e. Можно использовать тот факт, что $a_{\alpha(j)}$ есть число индексов таких, что $i > j$ и $\alpha(i) < \alpha(j)$, что приводит к нижеследующему алгоритму:

// here goes some code

Сложность полученного способа, конечно, имеет порядок квадрата длины перестановки.

f. Пусть a — элемент из $[0,1[\times[0,2[\times\cdots\times[0,n[$. Построим перестановку α, для которой a является массивом инверсий, следующим способом:

• элемент n помещаем в массив, индексированный с помощью [1, n], представляющий α, оставляя a_n **пустых ячеек** справа от n; это означает в точности, что $\alpha^{-1}(n) = n - a_n + 1$;

• затем помещаем $n-1$ в массив α, оставляя α_{n-1} **пустых ячеек** справа от $n-1$;

• продолжаем, зная, что на k-м этапе этого процесса $k-1$ величин уже размещены в массиве, следовательно, в массиве α остается $n-k$ свободных мест, и, с другой стороны, величина α_{n-k} строго меньше, чем $n-k$.

// here goes some code

В алгоритме 15 использована оптимизация: свободные места в массиве, представляющем перестановку α, отмечены числами 1, что позволяет не помещать это последнее значение в массив, представляющий α, в конце алгоритма.

24. Перебор перестановок транспозициями $(i, i+1)$

a. Рассуждаем индукцией по n, при этом случаи $n=1$ и $n=2$ очевидны. С помощью перестановки σ интервала [1, n] можно построить $n+1$ перестановок $\sigma^1, \sigma^2, \ \ldots\ , \sigma^{n+1}$ интервала [1, $n+1$], где перестановка σ^i получается включением в σ элемента $n+1$ на i-ое место; например, если $\sigma = (52413)$, то

$$\sigma^1 = (\underline{6}\ 5\ 2\ 4\ 1\ 3),\quad \sigma^2 = (5\ \underline{6}\ 2\ 4\ 1\ 3),\ \dots,$$
$$\sigma^5 = (5\ 2\ 4\ 1\ \underline{6}\ 3),\quad \sigma^6 = (5\ 2\ 4\ 1\ 3\ \underline{6}).$$

Если $\sigma_1, \sigma_2, \dots, \sigma_{n!}$ и есть такая последовательность перестановок на [1, n], то соответствующую последовательность перестановок интервала [1, n+1] получаем следующим образом:

$$\sigma_1^1, \sigma_1^2, \dots, \sigma_1^{n+1},\quad \sigma_2^{n+1}, \sigma_2^n, \dots, \sigma_2^1$$
$$\sigma_3^1, \sigma_3^2, \dots, \sigma_3^{n+1},\quad \sigma_4^{n+1}, \sigma_4^n, \dots, \sigma_4^1 \quad \text{и т.д.}$$

b. Действуем индукцией по n; знаем, что *b* — последующий элемент для a в знакопеременном лексикографическом порядке — получается изменением *одной* компоненты . Предположим сначала, что эта компонента — последняя; тогда имеем $b_n = a_n \pm 1$ и $b_j = a_j$ для $1 \leqslant j \leqslant n-1$. Если i — индекс n в α (т.е. $\alpha(i) = n$), то имеем $\beta = \alpha \circ (i, i-1)$ в случае $b_n = a_n + 1$, и $\beta = \alpha \circ (i, i+1)$ в случае $b_n = a_n - 1$, при этом запись (j, k) означает транспозицию индексов j и k.

Теперь предположим, что компонента, по которой различаются a и b, не является последней. Поскольку b — последующий элемент для a в знакопеременном лексикографическом порядке, имеем $a_n = b_n = 0$ или $a_n = b_n = n$ и $b_{[1..n-1]}$ есть последующий элемент для $a_{[1..n-1]}$ в лексикографическом знакопеременном произведении $[0,1[\times[0,2[\times\cdots\times[0,n-1[$, причем компонентой с самым большим индексом массива инверсий является та, которая меняется быстрее всех во время перебора в лексикографическом знакопеременном порядке. Если α' (соответственно, β') означает перестановку [1, $n-1$], для которой $a_{[1..n-1]}$ (соответственно, $b_{[1..n-1]}$) массив инверсий, то β' получается из α' транспозицией двух последовательных элементов (гипотеза индукции). Тогда утверждение верно также и для α и β, поскольку элемент n находится в этих перестановках либо на месте n (случай $a_n = b_n = 0$), либо на месте 1 (случай $a_n = b_n = n$).

c. Пусть α — перестановка интервала [1, n] и $a = (a_1, a_2, \dots, a_n)$ — ее массив инверсий. По предыдущему сигнатура перестановки α является также сигнатурой a в знакопеременном лексикографическом произведении. Алгоритм вычисления последующего элемента в знакопеременном лексикографическом произведении приводит к алгоритму 16-A.

В начале тела основного цикла этого алгоритма делается попытка опустить элемент q (применить транспозицию $(\alpha^{-1}(q)-1, a^{-1}(q))$ к перестановке α) или же поднять его.

Фактически, бесполезно приниматься за предварительное вычисление массива инверсий a. Достаточно вычислить при необходимости элемент a_q, что может быть реализовано одновременно с вычислением $\alpha^{-1}(q)$ благодаря алгоритму 16-B. В этом втором алгоритме результирующим значением i является $\alpha^{-1}(q)$.

// here goes some code

25. Принцип включения-исключения или формула решета

a. Первая формула удобно получается индукцией по $|I|$, числу элементов I, с использованием хорошо известной формулы $|A \cup B| = |A| + |B| - |A \cap B|$. Вторая получается переходом к дополнениям. Чтобы получить формулу Сильвестра, достаточно записать:

$$\bigcap_{i \in I} \overline{X_i} = \overline{\bigcup_{i \in I} X_i} = X - \bigcup_{i \in I} X_i,$$

затем применить первую формулу.

b. Пусть X — множество всех перестановок на $[1, n]$ и X_i — множество перестановок, имеющих i фиксированной точкой, $1 \leqslant i \leqslant n$. Искомое число σ_n:

$$\sigma_n = \Big|\bigcup_{i \in [1,n]} \overline{X_i}\Big| = \sum_{J \subset [1,n]} (-1)^{|J|} \Big|\bigcap_{i \in J} X_i\Big|.$$

Множество $\bigcap X_i$ есть множество J перестановок на $[1, n]$ и, следовательно, содержит $(n - |J|)!$ элементов, откуда:

$$\sigma_n = \sum_{J \subset [1,n]} (-1)^{|J|} (n - |J|)! = \sum_{k=0}^{n} (-1)^k C_n^k (n-k)! = \sum_{k=0}^{n} (-1)^k \frac{n!}{k!}$$

c. Положим здесь X равным интервалу [1, n] и для $1 \leqslant i \leqslant k$ пусть X_i — множество элементов из X, которые кратны p_i. Тогда имеем:

$$\phi(n) = \Big|\bigcap_{i\in[1,k]} \overline{X_i}\Big| = \sum_{J\subset[1,k]} (-1)^{|J|}\Big|\bigcap_{i\in J} X_i\Big|$$

Множество $\bigcap X_i$ здесь является множеством элементов из X, кратных $\prod_{i\in J} p_i$; но если d является делителем n, имеется точно n/d элементов из X, кратных d, откуда:

$$\phi(n) = n\sum_{J\subset[1,k]} \frac{(-1)^{|J|}}{\prod_{i\in J} p_i} = n(1-\frac{1}{p_1})(1-\frac{1}{p_2})\ \ldots\ (1-\frac{1}{p_k}).$$

d. Обозначим через X множество всех отображений из [1, n] в [1, n] и для $1 \leqslant i \leqslant n$ через X_i — множество отображений из X, не имеющих i в их образе; выберем в качестве весовой функции на X функцию $p(\alpha) = a_{1\alpha(1)}a_{2\alpha(2)}\ \ldots\ a_{n\alpha(n)}$: тогда нужно вычислить вес множества S_n всех перестановок на [1, n]. Заметим, что $S_n = \bigcap_{i\in[1,n]}\overline{X_i}$ и, значит, $perA = \sum_{J\subset[1,n]}(-1)^{|J|}p(\bigcap_{i\in J} X_i)$. Но $\bigcap_{i\in J} X_i$ есть множество всех отображений, образы которых не встречаются в J, следовательно:

$$p(\bigcap_{i\in J} X_i) = \sum_{\alpha:[1,n]\to\overline{J}} a_{1\alpha(1)}\ \ldots\ a_{n\alpha(n)} = \sum_{j\notin J} a_{1j}\sum_{j\notin J} a_{2j}\ \ldots\ \sum_{j\notin J} a_{nj}$$

отсюда получаем формулу $perA = \sum_{J\subset[1,n]}(-1)^{|J|}\prod_{i=1}^{n}\sum_{j\notin J} a_{ij}$, которая при замене J его дополнением дает формулу Райзера.

26. Произведение многочленов, заданных массивами

a. Алгоритм справа дает функцию умножения двух многочленов и Q, где многочлен R степени deg + deg Q (который дает результат в конце алгоритма) должен быть предварительно инициализирован нулем.

// here goes some code

b. Изучая предыдущий алгоритм, устанавливаем, что его сложность, как по числу перемножений, так и сложений, равна произведению высот двух многочленов: (deg P + 1) $\times$ (deg Q + 1) — обычно высотой многочлена называют число его ненулевых коэффициентов, но в этом алгоритме, который не учитывает случай нулевых коэффициентов, можно рассматривать высоту многочлена как число всех коэффициентов. Значит, возможно улучшить предыдущий алгоритм, исключив все ненужные перемножения: это сделано в алгоритме 17. В противовес тому, что можно было бы подумать, эта оптимизация вовсе не смехотворная и активно применяется при умножении разреженных многочленов.

27. Возведение в степень многочленов, заданных массивами

a. Очень просто вычислить сложность алгоритма возведения в степень последовательными умножениями, если заметить, что когда P — многочлен степени d, то P^i — многочлен степени id. Если обозначить $C_{mul}(n)$ сложность вычисления P^n, то рекуррентное соотношение $C_{mul}(i+1) = C_{mul}(i) + (d+1) \times (id+1)$ дает нам:

$$C_{mul}(n) = (d+1)\sum_{i=1}^{n-1} id + 1 = \frac{n^2 d(d+1)}{2} + \frac{n(d+1)(2-d)}{2} - (d+1).$$

b. Что касается возведения в степень с помощью дихотомии (т.е. повторяющимся возведением в квадрат), вычисления несколько сложнее: зная P^{2^i}, вычисляем $P^{2^{i+1}}$ с мультипликативной сложностью $(2^i d+1)^2$. Как следствие имеем:

$$C_{sqr}(2^l) = \sum_{i=0}^{l-1}(2^i d+1)^2 = \frac{d^2(4^l-1)}{3} + 2d(2^l-1) + l =$$
$$= \frac{d^2n^2}{3} + 2nd + \log_2 n - 2d - \frac{d^2}{3}$$

Предварительное заключение, которое можно вывести из предыдущих вычислений, складывается в пользу дихотомического возведения в степень: если n есть степень двойки (гипотеза ad hoc), этот алгоритм еще выдерживает конкуренцию, даже если эта победа гораздо скромнее в данном контексте ($n^2d^2/3$ против $n^2d^2/2$), чем когда работаем в $\mathbb{Z}/p\mathbb{Z}$ ($2\log_2 n$ против n).

Но мы не учли корректирующие перемножения, которые должны быть выполнены, когда показатель не является степенью 2-х. Если $2^{l+1}-1$, нужно добавить к последовательным возведениям в квадрат перемножения всех полученных многочленов. Умножение многочлена $P^{(2^i-1)d}$ степени $(2^i-1)d$ на многочлен P^{2^id} степени 2^id вносит свой вклад из $((2^i-1)d+1)\times(2^id+1)$ умножений, которые, будучи собранными по всем корректирующим вычислениям, дают дополнительную сложность:

$$
\begin{aligned}
C'_{sqr}(2^{l+1}-1) &= \sum_{i=1}^{l}((2^i-1)d+1)\times(2^id+1) = \\
&= \frac{d^2n^2}{3} - \frac{4d^2n}{3} + 2nd - 2d^2 + d(1-\lfloor\log_2 n\rfloor).
\end{aligned}
$$

Теперь можно заключить, что дихотомическое возведение в степень не всегда является лучшим способом для вычисления степени многочлена с помощью перемножений многочленов. Число перемножений базисного кольца, которые необходимы, $C_{sqr}(n)$ — в действительности заключено между $C_{sqr}(2^{\lfloor log_2 n\rfloor})$ и $C_{sqr}(2^{\lfloor log_2 n\rfloor})+C'(2^{\lfloor log_2 n\rfloor+1}-1)$, т.е. между $n^2d^2/2$ и $2n^2d^2/3$, тогда как простой алгоритм требует всегда $n^2d^2/2$ перемножений. В частности, если исходный многочлен имеет степень, большую или равную 4, возведение в степень наивным методом требует меньше перемножений в базисном кольце, чем бинарное возведение в степень, когда n имеет форму 2^l-1.

Можно пойти еще дальше без лишних вычислений: можно довольно просто доказать, что если n имеет вид $2^l+2^{l-1}+c$ (выражения, представляющие двоичное разложение n), то метод вычисления последовательными перемножениями лучше метода, использующего возведение в квадрат (этот последний метод требует корректирующего счета ценой, по крайней мере, $n^2d^2/9$). Все это доказывает, что наивный способ является лучшим для этого класса алгоритмов, по крайней мере в половине случаев.

Действительно, МакКарти [124] доказал, что дихотомический алгоритм возведения в степень оптимален среди алгоритмов, оперирующих повторными умножениями, если действуют с плотными многочленами (антоним к разреженным) по модулю m, или с целыми и при условии оптимизации возведения в квадрат для сокращения его сложности наполовину (в этом случае сложность действительно падает приблизительно до $n^2d^2/6+n^2d^2/3 = n^2d^2/2$).

28. Небольшие оптимизации для произведений многочленов

a. Алгоритм состоит просто в применении формулы квадрата суммы:

$$\left(\sum_{0\leqslant i\leqslant n} a_iX^i\right)^2 = \sum_{0\leqslant i\leqslant n} a_i^2X^{2i} + \sum_{0\leqslant i<j\leqslant n} 2a_ia_jX^{i+j},$$

что дает $n+1$ умножений для первого члена и $n(n+1)/2$ — для второго, или в целом $(n+1)(n+2)/2$ умножений, что близко к половине предусмотренных умножений, когда n большое.

b. Достаточно легко для вычисления произведения двух многочленов $P = aX + b$ и $Q = cX + d$ находим формулы $U = ac$, $W = bd$, $V = (a + b)(c + d)$ и $PQ = UX^2 + (V - U - W)X + W$, в которых появляются только три элементарных умножения, но четыре сложения (само же существование этих формул связано с записью тензорного ранга произведения многочленов, приведенной в главе IV). Можно рекурсивно применить этот процесс для умножения двух многочленов P и Q степени $2^l - 1$, представляя их в виде $P = AX^{2^l-1} + B$, $Q = CX^{2^l-1} + D$ и применяя предыдущие формулы для вычисления PQ в зависимости от A, B, C и D, где каждое произведение AB, CD и $(A + B)(C + D)$ вычисляется с помощью рекурсивного применения данного метода (это метод Карацубы). Все это дает мультипликативную сложность $\mathcal{M}(2^l)$ и аддитивную сложность $\mathcal{A}(2^l)$ такие, что:

$$\mathcal{M}(2^l) = 3\mathcal{M}(2^{l-1}),\ \ \ldots,\ \ \mathcal{M}(2) = 3\mathcal{M}(1),\quad \mathcal{M}(1) = 1,$$
$$\mathcal{A}(2^l) = 3\mathcal{A}(2^{l-1}) + 3\cdot 2^l,\ \ \ldots,\ \ \mathcal{A}(2) = 3\mathcal{A}(1) + 6,\quad \mathcal{A}(1) = 1.$$

В этой последней формуле член $3\cdot 2^l$ представляет собой число элементарных сложений, необходимых, чтобы сделать два сложения многочленов степени $2^{l-1} - 1$ ($a + b$ и $c + d$) и два вычитания многочленов степени $2^l - 1(U - V - W)$. Суммируя каждое из этих выражений, находим для n, являющегося степенью двойки:

$$\mathcal{M}(n) = n^{\frac{\log 3}{\log 2}} \approx n^{1,585} \quad \text{и} \quad \mathcal{A}(n) = 7n^{\frac{\log 3}{\log 2}} - 6n.$$

К сожалению, этот принцип остается теоретическим, и на его основе нужно построить итерационный алгоритм, чтобы получить разумную эффективность (цена управления рекурсией очень велика).

29. Высота произведения двух многочленов

a. Не говоря даже об оценке стоимости умножения двух разреженных многочленов, уже очень трудно — и даже невозможно, потому что эта высота зависит только от высоты каждого из множителей, — выразить высоту произведения двух многочленов как функцию высот исходных многочленов. Однако можно довольно легко ограничить ее произведением высот исходных многочленов. Два многочлена $Q = b_{q-1}X^{q-1} + \ \ldots\ + b_0$ и $P = a_{p-1}X^{(p-1)q} + a_{p-2}X^{(p-2)q} + \ \ldots\ + a_0$ показывают, что эта граница может быть достигнута.

b. Положим $P = \sum_{i=1}^{d} a_i X^{\alpha_i}$. Вычисление P^n дает нам выражение $\sum a_{i_1} \ldots a_{i_n} X^{\alpha_{i_1} + \ \cdots\ + \alpha_{i_n}}$. Эта формула дает такое представление для P^n, в котором не сделано никакого упрощения и никакой группировки членов. Высота P^n дается числом членов, имеющих разные степени, и чтобы оценить ее сверху, нужно сгруппировать члены, которые очевидным образом имеют одинаковые степени. Поскольку сложение показателей неизвестного коммутативно, можно переписать P^n как сумму мономов степеней $\alpha_{i_1} + \alpha_{i_2} + \ \ldots\ + \alpha_{i_n}$, где последовательность $(\alpha_{i_k})_k$ — возрастающая. Если все эти суммы различны, высота многочлена P^n будет максимальной и равной числу возрастающих отображений (в широком смысле) интервала $[1,\ n]$ в интервал $[1,\ d]$. Это число отображений хорошо известно (Берж [19]) и равно $\begin{pmatrix} d+n-1 \\ n \end{pmatrix}$.

Этот факт можно легко доказать. Действительно, рассмотрим возрастающее отображение f интервала $[1,\ n]$ в $[1,\ d]$; множество $\{f(1), 1+f(2), \ldots, n-1+f(n)\}$ является частью из n элементов множества из $n+d-1$ элементов. Обратно, если рассмотреть такую часть $\{x_1 < x_2 \ldots\ < x_{n-1} < x_n\}$, то последовательность $x_1, x_2-1, \ldots, x_n-n+1$ — возрастающая со значениями в $[1,\ d]$.

c. Прежде чем доказать требуемый результат, сначала сформулируем небольшую лемму (которая легко доказывается индукцией): для $k \geqslant 1$ и $p \geqslant 2$ имеем $kF_p < F_{p+k}$. Доказательство максимального характера (в смысле, данном в формулировке условия) многочлена P состоит просто в том, чтобы показать, если положить $S_i = \sum F_{1+ni_k}$, что для различных $i = \{i_1, \ldots, i_n\}$ и $j = \{j_1, \ldots, j_n\}$ суммы S_i и S_j различны. Предположим, что множества i и j упорядочены в возрастающем порядке; мы покажем, что если $F_{1+ni_n} \neq F_{1+nj_n}$, то две суммы различны. Предположим, что $i_n > j_n$, и пусть $i_n \geqslant j_n + 1$. Можно применить лемму, сформулированную в начале этого вопроса, и вывести отсюда, что $F_{1+ni_n} > nF_{1+nj_n}$; числа Фибоначчи входят в суммы, расположенные в порядке возрастания, и суммы S_i и S_j содержат только n членов; отсюда непосредственно выводится, что $S_i > S_j$.