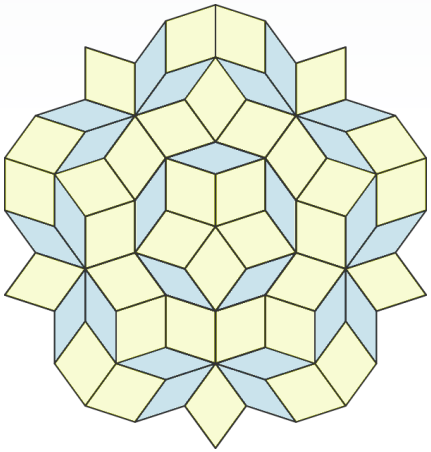
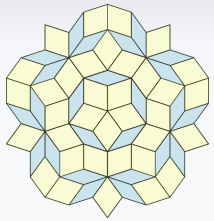


Рекурсивные алгоритмы и производящие функции

Дополнительная лекция по ДМ

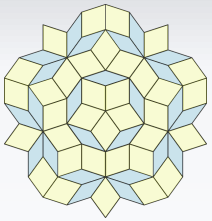
4 мая, 2023 г.



Воспоминание I: Сложность в худшем случае

- $f(x) = O(g(x)) \Leftrightarrow \exists C(f(x) < C \cdot g(x))$ для достаточно больших x;
- $f(x) = \Omega(g(x)) \Leftrightarrow \exists c(f(x) > c \cdot g(x))$ для достаточно больших x;
- $f(x) = \Theta(g(x)) \Leftrightarrow \exists c, C(C \cdot g(x) > f(x) > c \cdot g(x))$ для достаточно больших x.

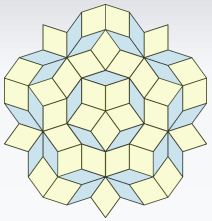
Для теоретически обоснованных оценок важно выбрать дискретный параметр x (размер структуры данных, количество определённого рода операций, и т.д.). Параметр должен быть абстрактным (в терминах алгоритма), а не физическим.



Почему не время / память?

- Колебания значений из-за посторонних процессов (вмешательство фоновых программ, неявное распараллеливание). Можно отчасти решить включением специального тестирующего скрипта.
- Зависимость от компилятора и набора оптимизаций в нём. Эта зависимость делает ненадёжным также подсчёт конкретных низкоуровневых инструкций.

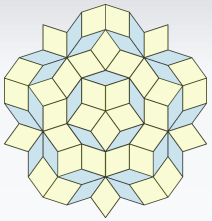
Колебания, вносимые в пункте 1, можно сгладить применением аппроксимационных оценок, т.к. распараллеливание в общем случае даёт сублинейный выигрыш. Колебания, вносимые в пункте 2, могут привести к изменению асимптотики.



Техника оценки сложности

- Вариант 1 (индуктивный): найти какую-нибудь меру сложности и оценить, как она уменьшается пошагово;

Сортировка пузырьком чисел от 1 до N. Сопоставим каждому массиву A размера N меру неупорядоченности: кортеж, где $N - i$-ый элемент — это число элементов $A[j]$, таких что $|A[j] - j| = i$. Например, массиву $(3\ 4\ 2\ 1)$ соответствует кортеж $\langle 1,\ 2,\ 1,\ 0\rangle$; отсортированному массиву — кортеж $\langle 0,\ \ldots,\ N\rangle$. Тогда мера неупорядоченности будет убывать лексикографически с каждым шагом сортировки. Это грубая оценка, т.к. допускает перепрыгивания типа $\langle 1,\ \ldots,\ N-1\rangle \to \langle 0,\ N,\ \ldots,\ 0\rangle$, каких не может быть при реальной сортировке. Но если заметить, что уменьшение каждого элемента кортежа не может быть сделано более, чем N раз, то получается уже адекватная оценка $O(N^2)$.

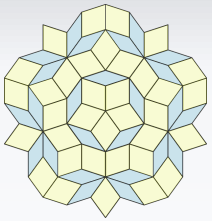


Техника оценки сложности

- Вариант 1 (индуктивный): найти какую-нибудь меру сложности и оценить, как она уменьшается пошагово;
- Вариант 2 (конструктивный): найти худший случай и оценить количество шагов для него.

Та же самая мера неупорядоченности подсказывает, как найти худший случай. Для любого массива размера N не более, чем 2 элемента имеют отличие в позициях от значений на $N - 1$. Далее по индукции. После чего достаточно посчитать количество шагов сортировки на построенном массиве.

Данное построение не обосновывает, почему хуже, чем этот случай, не бывает, однако позволяет построить полноценный бенчмарк для «достаточно плохих» случаев.

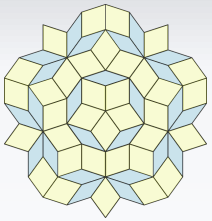


Техника оценки сложности

- Вариант 1 (индуктивный): найти какую-нибудь меру сложности и оценить, как она уменьшается пошагово;
- Вариант 2 (конструктивный): найти худший случай и оценить количество шагов для него.

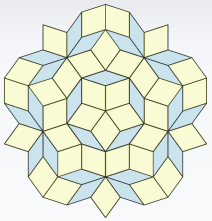
На практике при работе со сложными структурами данных точные «худшие случаи» почти никогда не ищут. Намного проще работать с «достаточно плохими» относительно выбранной меры сложности[a].

[a] Тем более, что точные худшие случаи часто являются патологиями, которые никогда не встречаются при реальном применении алгоритма.



Сложность «в среднем»

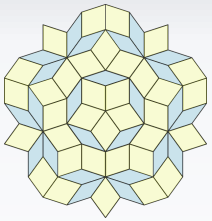
- Требует анализа распределения данных и адекватно работает для простых структур (массивы, произвольные графы).
- Очень плохо работает для сложных структур: например, «случайная» (т.е. случайно порождённая) программа содержит грубые семантические ошибки с вероятностью 1. Случайно порождённое уравнение в словах не имеет решений, причём это устанавливается тривиальными алгоритмами с вероятностью 1.
- На практике сложность в среднем оценивается на фрагментах реальных программ, либо требует порождения специальных бенчмарков по нетривиальным алгоритмам.



Воспоминание II: рекуррентные соотношения

Если функция $f(x)$ определяется через равенство $f(x) = G(f(h_1(x)), \ldots, f(h_k(x)), x)$, то говорят, что f задаётся рекуррентным соотношением. Если $f(x) = G\left(f\left(h_1(x)\right), \ldots, f\left(h_k(x)\right)\right)$, то рекуррентность называется однородной. Если $\forall i(h_i(x) = x - s_i)$, а G определяет линейную комбинацию вызовов f, то рекуррентность называется линейной.

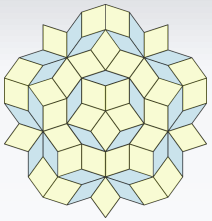
- Возникают при оценках сложности по индукции.
- Описывают сложность рекурсивных алгоритмов.



Воспоминание II: рекуррентные соотношения

Если функция $f(x)$ определяется через равенство $f(x) = G(f(h_1(x)), \ldots, f(h_k(x)), x)$, то говорят, что f задаётся рекуррентным соотношением. Если $f(x) = G\left(f\left(h_1(x)\right), \ldots, f\left(h_k(x)\right)\right)$, то рекуррентность называется однородной. Если $\forall i(h_i(x) = x - s_i)$, а G определяет линейную комбинацию вызовов f, то рекуррентность называется линейной.

- Переборные EXPTIME-алгоритмы: линейные рекуррентности по размеру входных данных. Например, $f(n) = 2 \cdot f(n-1) + n^2$.
- Субэкспоненциальные алгоритмы: как линейные (например, $f(n) = 2 \cdot f(n-1) - f(n-2)$), так и нелинейные (например, $f(n) = 2 \cdot f\left(\frac{n}{2}\right) + n$).



Мастер–теорема («разделяй и властвуй»)

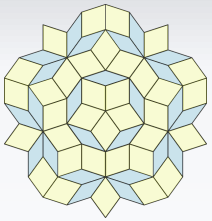
Пусть функция $f(n)$ описывается следующим рекуррентным соотношением:

$$f(n) = a \cdot f\left(\frac{n}{b}\right) + g(n)$$

Тогда сложность $f(n)$ оценивается так:

- если $g(n) = O(n^c)$, где $c < \log_b a$, тогда $f(n) = \Theta(n^{\log_b a})$;
- если $g(n) = \Theta(n^{\log_b a})$, тогда $f(n) = \Theta(n^{\log_b a} \cdot \log n)$;
- если $g(n) = \Omega(n^c)$, где $c > \log_b a$, причём асимптотически верно, что для некоторой величины $t < 1$ $a \cdot g\left(\frac{n}{b}\right) \leqslant t \cdot g(n)$, тогда $f(n) = \Theta\left(g(n)\right)$.

Сочетание условий $c > \log_b a$ и $a \cdot g\left(\frac{n}{b}\right) \leqslant t \cdot g(n)$ выполняется почти всегда.



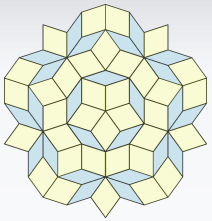
Экспоненциальные рекуррентности

Классика жанра анализа программ: алгоритм полиномиален по размеру графа, а граф экспоненциален от входных данных. Переход к оценке по размеру входа порождает рекуррентные соотношения уже другой формы, асимптотику которых, однако, также можно оценить с помощью мастер-теоремы.

- Пример оценки:

$$f(n) = 3 \cdot f(n-1) + 2^n$$

Как поступить с двумя рекурсивными вызовами:
$f(n) = a_1 \cdot f(n-k_1) + a_2 \cdot f(n-k_2) + g(n)$?



Экспоненциальные рекуррентности

- Частный случай:

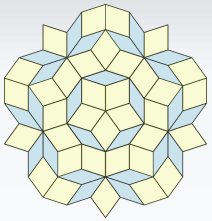
$$f(n) = 3 \cdot f(n-1) + 2^n$$

- Общий случай:

$$f(n) = a \cdot f(n-1) + b^n$$

Метод замены переменных: вводим функцию g такую, что $g(b^n) = f(n)$, и затем осуществляем замену b^n на n'.

Как поступить с двумя рекурсивными вызовами:
$f(n) = a_1 \cdot f(n-k_1) + a_2 \cdot f(n-k_2) + g(n)$?



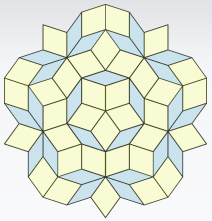
Пример рекуррентности

Даны слова $S_1 = a_1 \dots a_n$, $S_2 = b_1 \dots b_m$. Найти их максимальную общую подпоследовательность.

Это NP-трудная задача. Будем анализировать очень наивный алгоритм её решения, связанный с полным перебором всех возможных подпоследовательностей:

- Находясь на i-й позиции в S_1, ищем все подпоследовательности суффикса $a_i \dots a_n$ и слова S_2, включающие a_i. То есть мы либо успешно сопоставляем a_i с b_1 и решаем задачу для строк $a_{i+1} \dots a_n$ и $b_2 \dots b_m$, либо решаем задачу для строк $a_i \dots a_n$ с $b_2 \dots b_m$ так, что a_i сопоставляется с каким-нибудь b_k.
- Сдвигаем i на 1 вправо.

Рекуррентное соотношение, дающее оценку количества перебираемых случаев, выглядит следующим образом: $g(n+1, m+1) = g(n, m+1) + g(n+1, m)$.



Пример рекуррентности

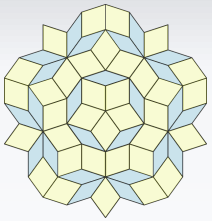
Даны слова $S_1 = a_1 \dots a_n$, $S_2 = b_1 \dots b_m$. Найти их максимальную общую подпоследовательность.

Рекуррентное соотношение, дающее оценку количества перебираемых случаев, выглядит следующим образом: $g(n+1, m+1) = g(n, m+1) + g(n+1, m)$.

Положим $g(n, 1) = g(1, n) = n$ (побуквенный перебор), и оценим для начала $h_2(n) = g(2, n)$. Получаем соотношение: $h_2(n+1) = n + 1 + h_2(n)$. То есть $h_2(n) = O(n^2)$.

Очевидно, $h_{i+1}(n) = h_i(n) + h_{i+1}(n-1)$, из чего можно вывести $h_i(n) = O(n^i)$. На практике эта оценка несколько лучше: $O\left(\frac{n^i}{i!}\right)$.

Неудобство заключается в том, что для построения этой оценки мы вынуждены пользоваться знаниями о сумме ряда $1^i + 2^i + \dots n^i$, а хотелось бы уметь выводить их в общем виде.

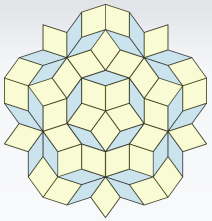


Пример рекуррентности

Даны слова $S_1 = a_1 \dots a_n$, $S_2 = b_1 \dots b_m$. Найти их максимальную общую подпоследовательность.

Рекуррентное соотношение, дающее оценку количества перебираемых случаев, выглядит следующим образом: $g(n+1,\, m+1) = g(n,\, m+1) + g(n+1,\, m)$.

Заметим, что если положить $g(n,\, 1) = g(1,\, n) = q$ (проверка на вхождение буквы в конечный словарь стоимостью в константу q), то все оценочные полиномы сразу станут на 1 степень меньше. И хотя диагональная функция $h_n(n) = g(n, n)$ всё равно растёт экспоненциально быстро, использование словаря даже только для базисных случаев даёт заметный выигрыш в скорости.

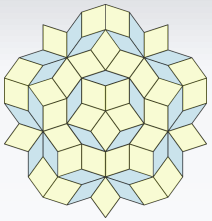


Пример комбинаторной задачи

Дан недетерминированный конечный автомат $\mathscr{A}$ без ε-переходов над $\{\mathfrak{a}, \mathfrak{b}\}$. Известно, что максимальное количество дуг, выходящих из состояния $\mathscr{A}$, равно $\mathfrak{k}$ ($\mathfrak{k}$ может быть больше 2, т.к. $\mathscr{A}$ — НКА).

- Найти множество слов длины, меньшей или равной $\mathfrak{n}$;
- Найти количество слов длины, меньшей или равной $\mathfrak{n}$.

- Тупой рекурсивный алгоритм по длине слова – $\mathrm{O}(???)$;
- Разбиение на две подзадачи — $\mathrm{O}(???)$;
- «Разделяй и властвуй» — $\mathrm{O}(???)$.



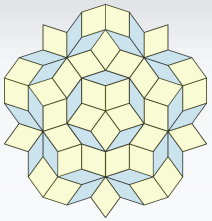
Пример комбинаторной задачи

Дан недетерминированный конечный автомат $\mathscr{A}$ без ε-переходов над $\{a, b\}$. Известно, что максимальное количество дуг, выходящих из состояния $\mathscr{A}$, равно k (k может быть больше 2, т.к. $\mathscr{A}$ — НКА).

- Найти множество слов длины, меньшей или равной n;
- Найти количество слов длины, меньшей или равной n.

- Тупой рекурсивный алгоритм по длине слова – $O(k^n)$;
- Разбиение на две подзадачи — $O(max(2, \sqrt{k})^n)$;
- «Разделяй и властвуй» — $O(2^n)$. Не зависит от k.

Оценка $O(2^n)$ грубая (предполагает, что $\mathscr{A}$ порождает почти все возможные слова). Более точная оценка связана с решением задачи номер 2.

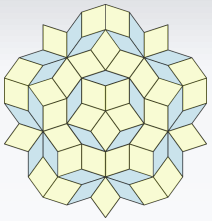


Ещё комбинаторные задачи

- Сколько существует слов длины 5 в алфавите $\{a, b\}$, не содержащих подстроку ab?
- Сколько существует слов длины n в алфавите $\{a, b\}$, не содержащих подстроку abb? Построить соответствующее рекуррентное соотношение.

Подсказка: рекурсия по количеству слов с соответствующими суффиксами.

Упражнение: слова в алфавите $\{a, b\}$, не содержащие подстроки $aabb$.



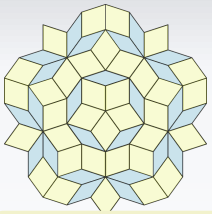
Ещё комбинаторные задачи

- Сколько существует слов длины 5 в алфавите $\{a, b\}$, не содержащих подстроку ab?
- Сколько существует слов длины n в алфавите $\{a, b\}$, не содержащих подстроку abb? Построить соответствующее рекуррентное соотношение.

Подсказка: рекурсия по количеству слов с соответствующими суффиксами.

Построенная рекуррентность для задачи 2:
$g(n+1) = g(n) + g(n-1) + 1$.

Упражнение: слова в алфавите $\{a, b\}$, не содержащие подстроки $aabb$.



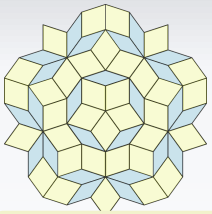
Формальные степенные ряды

$\mathbb{K}$ — какое-нибудь (числовое) поле. Существует взаимно-однозначное соответствие между списками элементов из $\mathbb{K}$ и многочленами с коэффициентами из $\mathbb{K}$. Т.е. представим списки чисел как «многочлены» (с бесконечным числом коэффициентов):

$$\sum_{n=0}^{\infty} a_n x^n$$

Приятная неожиданность

- Покоэффициентное сложение, коммутативное и ассоциативное.
- Умножение, коммутативное и ассоциативное + дистрибутивность.



Формальные степенные ряды

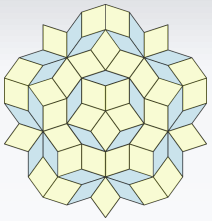
Производящая функция последовательности $\{a_n\}$ — это формальный степенной ряд:

$$\sum_{n=0}^{\infty} a_n x^n$$

Приятная неожиданность

- Покоэффициентное сложение, коммутативное и ассоциативное.
- Умножение, коммутативное и ассоциативное + дистрибутивность.
- Расширение алгебры многочленов! Частное и производная.

Значения, принимаемые x — не численные, а комбинаторные. (Полу)кольцо формальных рядов обычно обозначается $\mathbb{K}[[x]]$.

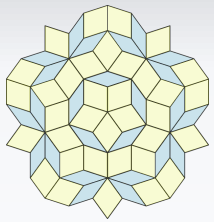


Произведение и частное

Произведение ПФ $\sum_{n=0}^{\infty} a_n x^n$ и $\sum_{n=0}^{\infty} b_n x^n$ есть ПФ $\sum_{n=0}^{\infty} c_n x^n$, где

$$c_n = \sum_{k=0}^{n} a_k \cdot b_{n-k}$$

Единицей для умножения, очевидно, является ряд с коэффициентами $(1, 0, 0, \dots)$ (единица в обычном смысле, так и обозначаем её : 1). Тогда имея $A[x] = \sum_{n=0}^{\infty} a_n x^n$, как найти такой ряд $B[x] = \sum_{n=0}^{\infty} b_n x^n$, что $A[x] \cdot B[x] = 1$?



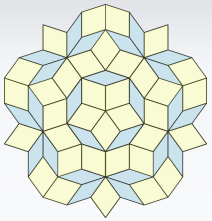
Произведение и частное

Произведение ПФ $\sum_{n=0}^{\infty} a_n x^n$ и $\sum_{n=0}^{\infty} b_n x^n$ есть ПФ $\sum_{n=0}^{\infty} c_n x^n$, где

$$c_n = \sum_{k=0}^{n} a_k \cdot b_{n-k}$$

Если $a_0 \neq 0$, то ряд $B[x]$ существует и определяется рекурсивно:

- $b_0 = \frac{1}{a_0}$
- $b_n \cdot a_0 = -a_1 \cdot b_{n-1} - \cdots - a_n \cdot b_0$

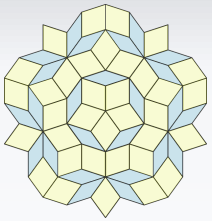


Краткое представление ряда

Пусть $P(x)$ — полином степени n. Рассмотрим «обращённый» полином $P^R(x) = P(\frac{1}{x}) \cdot x^n$ (тот же самый, но упорядоченный в перевёрнутом порядке).

Что получится, если формально разделить в столбик 1 на $(-k \cdot x + 1)^R$?

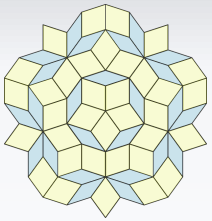
Это в точности наш рекурсивный алгоритм деления формальных рядов. Получаем способ краткой записи формального ряда (списка) в виде частного обратных полиномов.



Способы краткого представления

- ПФ для частного $\frac{1}{1-k\cdot x}$ — это $\sum_{i=0}^{\infty} k^i x^i$.
- Умножение на n-ую степень x — это сдвиг на n.
- ПФ для частного $\frac{P(x)}{Q(x)}$, где $Q(x)$ — многочлен без кратных и мнимых корней, получается с помощью разложения в сумму простых дробей со знаменателями вида $1 - k \cdot x$.

Упражнение: превратить рекуррентность для чисел Фибоначчи в частное вида $\frac{1}{P(x)}$ (или подсмотреть в интернете), а потом разделить 1^R на $P(x)^R$ в столбик.

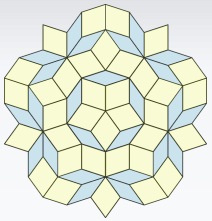


Алгоритм приведения к дроби

Ищем краткое представление для $G = \sum_{j \geqslant 0} a_j x^j$.

Строим уравнения на коэффициенты производящей функции и умножаем на подходящие степени x:

$$a_k = c_1 \cdot a_{k-1} + c_2 \cdot a_{k-2} + \cdots + c_k \cdot a_0 \qquad (\cdot x^0)$$
$$a_{k+1} = c_1 \cdot a_k + c_2 \cdot a_{k-1} + \cdots + c_k \cdot a_1 \qquad (\cdot x^1)$$
$$\ldots$$
$$a_n = c_1 \cdot a_{n-1} + c_2 \cdot a_{n-2} + \cdots + c_k \cdot a_{n-k} \qquad (\cdot x^{n-k})$$
$$\ldots$$



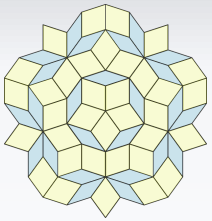
Алгоритм приведения к дроби

Ищем краткое представление для $G = \sum_{j \geqslant 0} a_j x^j$.

$$
\begin{aligned}
a_k &= c_1 \cdot a_{k-1} + c_2 \cdot a_{k-2} + \cdots + c_k \cdot a_0 && (\cdot x^0) \\
a_{k+1} &= c_1 \cdot a_k + c_2 \cdot a_{k-1} + \cdots + c_k \cdot a_1 && (\cdot x^1) \\
&\ldots \\
a_n &= c_1 \cdot a_{n-1} + c_2 \cdot a_{n-2} + \cdots + c_k \cdot a_{n-k} && (\cdot x^{n-k}) \\
&\ldots
\end{aligned}
$$

Теперь складываем все эти уравнения:

$$\sum_{j \geqslant k} a_j x^{j-k} = c_1 \cdot \sum_{j \geqslant k-1} a_j x^{j-k+1} + \cdots + c_k \cdot \sum_{j \geqslant 0} a_j x^j$$



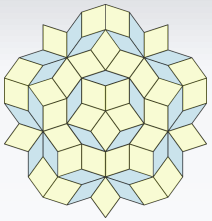
Алгоритм приведения к дроби

Ищем краткое представление для $G = \sum_{j \geqslant 0} a_j x^j$.

$$\sum_{j \geqslant k} a_j x^{j-k} = c_1 \cdot \sum_{j \geqslant k-1} a_j x^{j-k+1} + \cdots + c_k \cdot \sum_{j \geqslant 0} a_j x^j$$

Чтобы стало возможным подставить G, умножаем сумму на x^k:

$$\sum_{j \geqslant k} a_j x^j = c_1 x \cdot \sum_{j \geqslant k-1} a_j x^j + \cdots + c_k x^k \cdot \sum_{j \geqslant 0} a_j x^j$$



Алгоритм приведения к дроби

Ищем краткое представление для $G = \sum_{j \geqslant 0} a_j x^j$.

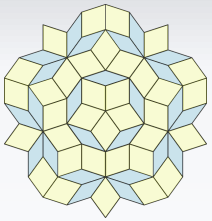
$$\sum_{j \geqslant k} a_j x^j = c_1 x \cdot \sum_{j \geqslant k-1} a_j x^j + \cdots + c_k x^k \cdot \sum_{j \geqslant 0} a_j x^j$$

Выражаем результат в терминах G и нескольких первых значений a_i:

$$G - \left(\sum_{j < k} a_j x^j \right) = c_1 x \cdot \left(G - \sum_{j < k-1} a_j x^j \right) + \cdots + c_k x^k \cdot G$$

Итог:

$$G = \frac{\sum_{j<k} a_j x^j - c_1 x \cdot \sum_{j<k-1} a_j x^j - \cdots + c_{k-1} x^{k-1} a_0}{1 - c_1 x - \cdots - c_k x^k}$$



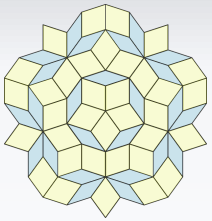
Расширенный бином Ньютона

Напомним, что

$$(x+y)^n = \sum_{i=0}^{n} \frac{n!}{k!(n-k)!} x^i y^{n-i}$$

Число сочетаний из n по k обычно записывается кратко: $\binom{n}{k}$.
Бином Ньютона можно расширить на отрицательные значения n:

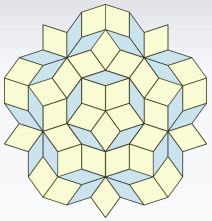
$$\binom{-n}{k} = \frac{(-1)^k \cdot (n+k-1)!}{k!(n-1)!} = \binom{n+k-1}{k} \cdot (-1)^k$$



Извлечение коэффициентов из ПФ

Если дана ПФ $F(x)$ в краткой форме (частное), то как посчитать коэффициент при n-ой степени x?

- Вычисление n-ого коэффициента в ряду Маклорена для $F(x)$.
- Или использование Бинома Ньютона и разложение в сумму элементарных дробей.



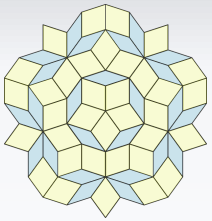
Теорема о рациональных ПФ

Квазиполином — это сумма вида $\sum \beta_i \cdot x^{q_i} \cdot k_i{}^{\alpha_i \cdot x}$.

Следующие утверждения относительно последовательности $\{a_n\}$ эквивалентны:

- ПФ $\{a_n\}$ является рациональной (может быть представлена в виде частного полиномов $P(x)/Q(x)$), где $Q(x) = 1 - c_1 \cdot x - \cdots - c_k \cdot x^k$;
- $f(i) = a_i$ описывается квазиполиномом, основания экспонент которого определяются корнями $Q(x)$;
- $\{a_n\}$ описывается линейной рекуррентностью $a_n = c_1 \cdot a_{n-1} + c_2 \cdot a_{n-2} + \cdots + c_k \cdot a_{n-k}$.

Более точно, если $Q(x) = (1 - q_1 \cdot x)^{t_1} \cdot \ldots (1 - q_k \cdot x)^{t_k}$, то $a_n = O(q_1^n \cdot n^{t_1 - 1} + \cdots + q_k^n \cdot n^{t_k - 1})$. Случай мнимых корней $Q(x)$ в лекции не рассматривается.

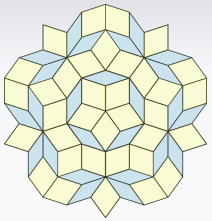


Пример оценки

Применим формальные ряды для оценки скорости роста $a_n = h_2(n) = n + h_2(n-1)$.

Берём начальное значение h_2: $h_2(0) = 1$, чтобы получить $a_1 = h_2(1) = g(2,1) = 2 = a_0 + 1$.
Соотношение имеет дополнительное слагаемое (n), поэтому придётся повторять конструкцию суммы по шагам.

$$\sum_{j \geqslant 1} a_j x^j = x \cdot \sum_{j \geqslant 0} a_j x^j + x \cdot \sum_{j \geqslant 0} (j+1) \cdot x^j.$$



Пример оценки

Применим формальные ряды для оценки скорости роста $a_n = h_2(n) = n + h_2(n-1)$.

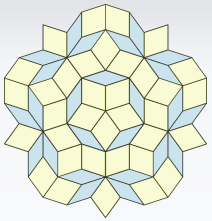
Берём начальное значение h_2: $h_2(0) = 1$, чтобы получить $a_1 = h_2(1) = g(2,1) = 2 = a_0 + 1$.

$$\sum_{j \geqslant 1} a_j x^j = x \cdot \sum_{j \geqslant 0} a_j x^j + x \cdot \sum_{j \geqslant 0} (j+1) \cdot x^j.$$

Ряд $\sum_{j \geqslant 0} (j+1) \cdot x^j = 1 + 2 \cdot x + 3 \cdot x^2 + \ldots$ — можно заметить, что это производная знакомого нам ряда $1 + x + x^2 + \cdots + x^n + \ldots$.

В терминах частных, $\left(\frac{1}{1-x}\right)' = \frac{1}{(1-x)^2}$. Дальше всё просто.

$$H_2 - 1 = x \cdot H_2 + \frac{x}{(1-x)^2}$$



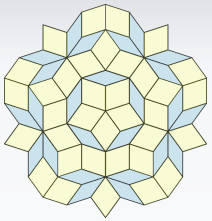
Пример оценки

Применим формальные ряды для оценки скорости роста $a_n = h_2(n) = n + h_2(n-1)$.

Берём начальное значение h_2: $h_2(0) = 1$, чтобы получить $a_1 = h_2(1) = g(2,1) = 2 = a_0 + 1$.

$$H_2 - 1 = x \cdot H_2 + \frac{x}{(1-x)^2}$$

$$H_2 = \frac{1 - x + x^2}{(1-x)^3}$$



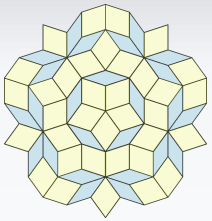
Пример оценки

Применим формальные ряды для оценки скорости роста $a_n = h_2(n) = n + h_2(n-1)$.

Берём начальное значение h_2: $h_2(0) = 1$, чтобы получить $a_1 = h_2(1) = g(2,1) = 2 = a_0 + 1$.

$$H_2 = \frac{1 - x + x^2}{(1-x)^3}$$

Разделив $1 - x + x^2$ на $1 - 3 \cdot x + 3 \cdot x^2 - x^3$ в столбик, мы можем убедиться, что получаются нужные оценки. Заодно из неоднородной рекуррентности $a_n = a_{n-1} + n$ построена однородная: $a_n = 3 \cdot a_{n-1} - 3 \cdot a_{n-2} + a_{n-3}$.
Чтобы оценить рост функции h_2, воспользуемся теоремой о рекуррентных соотношениях. Корень у знаменателя единственный, это 1, его кратность равна 3, поэтому $h_2(n) = O(1^n \cdot n^2) = O(n^2)$. Что и требовалось доказать.

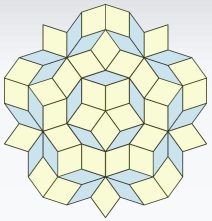


Пример оценки-2

Оценить скорость роста количества слов длины n, не содержащих abb.

Берём уже известное соотношение и начальные значения g:
$g(0) = 1$; $g(1) = 2$; $g(n) = g(n-1) + g(n-2) + 1$.
Соотношение имеет дополнительное слагаемое (1), поэтому придётся повторять конструкцию суммы по шагам.

$$\sum_{j \geqslant 2} a_j x^j = x \cdot \sum_{j \geqslant 1} a_j x^j + x^2 \cdot \sum_{j \geqslant 0} a_j x^j + x^2 \cdot \sum_{j \geqslant 0} x^j.$$



Пример оценки-2

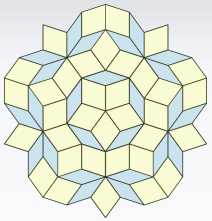
Оценить скорость роста количества слов длины n, не содержащих abb.

Берём уже известное соотношение и начальные значения g: $g(0) = 1$; $g(1) = 2$; $g(n) = g(n-1) + g(n-2) + 1$.

$$\sum_{j \geqslant 2} a_j x^j = x \cdot \sum_{j \geqslant 1} a_j x^j + x^2 \cdot \sum_{j \geqslant 0} a_j x^j + x^2 \cdot \sum_{j \geqslant 0} x^j.$$

Как свернуть последнюю сумму, мы уже знаем, поэтому остальные построения не представляют труда:

$$G - 1 - 2x = x \cdot (G - 1) + x^2 \cdot G + \frac{x^2}{1 - x}$$



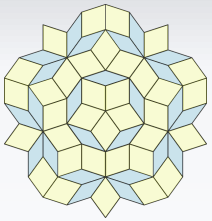
Пример оценки-2

Оценить скорость роста количества слов длины n, не содержащих abb.

Берём уже известное соотношение и начальные значения g: $g(0) = 1$; $g(1) = 2$; $g(n) = g(n-1) + g(n-2) + 1$.

$$G - 1 - 2x = x \cdot (G - 1) + x^2 \cdot G + \frac{x^2}{1-x}$$

$$G = \frac{1+x}{1-x-x^2} + \frac{x^2}{(1-x)(1-x-x^2)} = \frac{1}{1-2x+x^3}$$



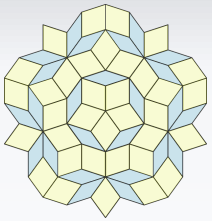
Пример оценки-2

Оценить скорость роста количества слов длины n, не содержащих abb.

Берём уже известное соотношение и начальные значения g:
$g(0) = 1$; $g(1) = 2$; $g(n) = g(n-1) + g(n-2) + 1$.

$$G = \frac{1+x}{1-x-x^2} + \frac{x^2}{(1-x)(1-x-x^2)} = \frac{1}{1-2x+x^3}$$

Разделив 1 на $1 - 2x + x^3$ в столбик, мы действительно можем убедиться, что получаются нужные оценки. Заодно из неоднородной рекуррентности $a_n = a_{n-1} + a_{n-2} + 1$ мы построили однородную: $a_n = 2 \cdot a_{n-1} - a_{n-3}$.
Чтобы оценить рост функции g, нужно найти разложение полинома $1 - 2x + x^3 = (1 - q_1 \cdot x)(1 - q_2 \cdot x)(1 - q_3 \cdot x)$.
Получаем $q_1 = 1$, $q_2 = \frac{\sqrt{5}+1}{2}$, $q_3 = \frac{1-\sqrt{5}}{2}$.

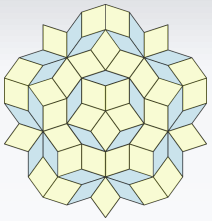


Пример оценки-2

Оценить скорость роста количества слов длины n, не содержащих abb.

Чтобы оценить рост функции g, нужно найти разложение полинома $1 - 2x + x^3 = (1 - q_1 \cdot x)(1 - q_2 \cdot x)(1 - q_3 \cdot x)$.
Получаем $q_1 = 1$, $q_2 = \frac{\sqrt{5}+1}{2}$, $q_3 = \frac{1-\sqrt{5}}{2}$.
Если стоит задача определить *точное* количество слов длины n, тогда придётся разложить G в сумму простых дробей (см. ниже) и найти коэффициенты A_1, A_2, A_3. Понятно, что для оценки асимптотики g это делать не обязательно.

$$\frac{1}{1 - 2x + x^3} = \frac{A_1}{1 - q_1 \cdot x} + \frac{A_2}{1 - q_2 \cdot x} + \frac{A_3}{1 - q_3 \cdot x}.$$

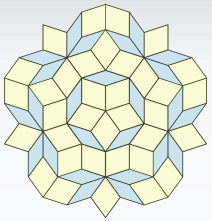


Связь ПФ и М.- Т.

Оценить асимптотику роста n-ого коэффициента производящей функции, заданной следующей рекуррентностью:

$$f(n) = a \cdot f(n-1) + b^{c \cdot n}$$

Вывести из этой оценки хотя бы один частный случай Мастер–Теоремы.



Теорема о регулярных языках

Определение

ПФ языка L — это формальный ряд вида $\sum_{i=1}^{\infty} n_i x^i$, где n_i — это количество слов языка длины ровно i.

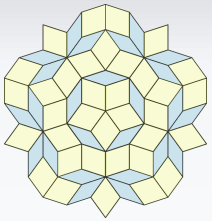
Теорема

Если язык L регулярен, тогда его ПФ является рациональной.

Показать, что язык $\{a^{2^n} \mid n \in \mathbb{N}\}$ не является регулярным.

Рабочий приём: стабилизация рекуррентного отрезка ряда.

Показать, что язык $\{a^{[\log_2 n]+3n} \mid n \in \mathbb{N}\}$ нерегулярен.



ПФ для регулярки

Только для однозначных регулярных выражений!

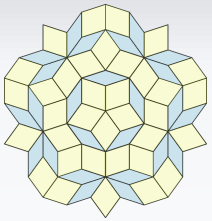
Алгоритм построения ПФ

Строим свёртку ПФ по следующему алгоритму:

1. Любой букве соответствует x;
2. Альтернативе соответствует сложение, конкатенации — умножение;
3. Если s является ПФ для выражения Φ, тогда $\frac{1}{1-s}$ является ПФ для выражения Φ^*.

Далее разворачиваем свёртку в ряд по вышеописанному алгоритму.

Физический смысл: комбинаторные объекты + однозначные сочетания этих объектов.



Пример применения

Оценить асимптотику роста количества слов в $\{a, b\}$, не содержащих подслова abb.

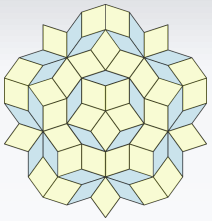
Строим однозначное регулярное выражение для искомых слов:

$$b^*(aa^*b)^*a^*$$

Здесь ещё нужно обосновать, почему это выражение однозначное (это почти очевидно) и почему оно описывает требуемый язык (например, по индукции). Дальше останется тупо применить алгоритм порождения ПФ:

$$G(b^*(aa^*b)^*a^*) = \frac{1}{1-x} \cdot \frac{1}{1 - x \cdot \frac{1}{1-x} \cdot x} \cdot \frac{1}{1-x} = \frac{1}{1-x} \cdot \frac{1}{1-x-x^2}.$$

Этот путь намного короче первого, если хорошо владеть техникой построения регулярок.



А если есть неоднозначность?

- Техника ПФ остаётся применимой для оценки асимптотики числа путей разбора регулярки. Можно сказать, что при этом мы оцениваем число слов в её линеаризованной версии (различающей разные вхождения букв).
- Однако для корректного определения ПФ для неоднозначной регулярки придётся озаботиться, чтобы разбор пустого слова был единствен (причём в любом подвыражении). Иначе деление формальных рядов может быть не определено, см. регулярка $(\mathfrak{a}^*)^*$.