

Я бы хотел обсудить связи с перечислением типов данных и смежные вопросы: типы данных (например,... Да, это хорошая тема, и её лучше сначала спроектировать, иначе она быстро расползётся в species, типы,...

1. Я бы предпочёл рабочий мини-курс: простой пример, пример с деревьями Каталана, более сложный... Отлично. Тогда я бы с самого начала выбрал язык комбинаторных видов (species), но использовал его очен... То есть, производная списков - это декартово произведение списков?

Да — в точности, если под L понимаем species линейных списков. $L' \cong L \times L$ Комбинаторный смысл такой:... Хорошо. Давай перейдём к деревьям Каталана.

Отлично. Здесь схема уже становится намного содержательнее. Возьмём полные плоские бинарные... Какие алгоритмические задачи решает tree zipper? То есть, какие конкретно алгоритмы [для каких задач]...

Да. Самая практическая идея zipper'a очень проста: он превращает дерево + выбранную вершину в... Хорошо. Давай, как ты предложил, посмотрим на pointing и на связь с Seven Trees in One.

Да. И здесь полезно разделить две близкие, но всё-таки разные линии: pointing живёт внутри... Да. Переходим к деревьям?

Да. И теперь я бы выбрал произвольные помеченные деревья, но начал с корневых - это самый естественн... Хорошо. Пойдём по пунктам. Я бы сказал, что некоторые пункты я понимаю (1,4,5), но всё равно хотелось б... Тогда действительно по одному пункту. Начнём с 1. Почему $T = X \cdot \text{SET}(T)$ для корневых помеченных деревьев...

Да, пункт 1 я понимаю. А вот пункт 2 нет.

Именно здесь скрывается главный трюк помеченной комбинаторики. Начнём с самого простого случая, а... Очень красиво. Это было 2 и 3. Теперь 4.

Да. Пункт 4 — это как раз место, где появляется формула Лагранжа для обращения рядов. У нас $T(x) = xeT(x)$... Нет, давай перейдем к пункту 5 [хотя сам по себе он довольно очевиден, но давай проговорим]. Меня...

Да. Пункт 5 сам по себе прост, а формула $U = T - 21T^2$ действительно гораздо интереснее. Здесь есть два... Да. Здорово! И пункт 6: что такое T' ?

Да. Здесь T' уже очень наглядно интерпретируется. Напомню: $T = X\text{SET}(T)$ где T — корневые помеченные... Извини, не понял этот пассаж: "ниже корня. Тогда корень обычный, рядом с ветвью, содержащей дырку,... Ты прав — моя формулировка была слишком сжатой. В выражении $X\text{SET}(T)T'$ только один множитель T' ...

Да, я понял. Замечательно! А есть ещё интересные [в этом смысле] типы данных [помимо списка, дерева... Да, и есть несколько очень удачных примеров. Причём некоторые дают даже более красивые сюрпризы, чем... Хорошо. Давай разберем set partitions. [А потом, чуть позже, и endofunctions].

Да. Здесь всё получается особенно чисто, потому что сама структура разбиения почти буквально... Хорошо. Продолжаем.

Да. Здесь как раз появляются две разные операции выделения, и их полезно развести. 1. Pointing: выделяе... Хорошо. Переходим к endofunctions.

Да. Это, пожалуй, лучший «составной» пример: здесь одновременно работают деревья, циклы, множества... Это совершенно грандиозно! Спасибо за лекции! У меня есть только одна проблема: я понимаю, что ты... Да, знаю именно этот подход: материал не сообщается сначала в готовом виде, а раскладывается на... Отлично! Поехали!

Поехали. Листок 1: алгебра структур и списки. Пока без лекции; только задачи. Отвечать можно словами,...

Задача 1. L - это либо пустой список $()$, как в Haskell [то есть, 1] или пара атом $\times$ список. То есть, формально,... Да, задача 1 решена точно: $L=1+XL$. И твой вопрос — как раз ключевой: почему из равенства типов можно...

Задача 2. Нам нужно знать список слева и, независимо от него [произведение], список справа. То есть, $L' = \dots$ Да, именно так: $L' = L \times L = L^2$. Смысл совершенно точный: список с одной дыркой однозначно задаётся парой... Задача 3. $L = 1/(1-X)$, $L' = 1/(1-X)^2$, то есть, $L' = L^2$. Или, $L' = L + XL'$, $L'(1-X) = L$ и опять: $L' = L^2$. Есть ли... Да — и есть очень красивый вариант, вообще без деления. После дифференцирования: $L' = L + XL'$. Теперь...

Задача 4. Если мы говорим о EGF, то коэффициент $1 = n!/n!$. А для n элементов у нас как раз $n!$ вариантов. ... Да — именно здесь стоит остановиться и развести понятия. Ты правильно чувствуешь: OGF не означает... Ага. Вот это, пожалуй, самое важное для меня на текущий момент (понимание OGF vs EGF). Замечательно!...

Да. Тогда Листок 2 я бы специально построил вокруг того различия, которое сейчас стало центральным:...

Задача 1. $B = 1 + X \times B^2$ [вроде так, но ... почему 1 перед + и X после? Если мы стираем метки? $C \times B^2$ всё... Да — тут ты поймал важную развилку. Я в условии зафиксировал, что атомами считаются листья, поэтому... Секундочку: то есть, размер - это количество атомов? [В первом варианте листьев, во-втором: "не листьев"]? Да, именно. В нашей алгебре X означает один атом размера 1, а 1 — структуру размера 0. Поэтому «размер... А! О сколько нам открытий чудных готовит сын ошибок трудных! Вот поэтому активное владение так и... Да! И твой способ совершенно законный. Более того, это хороший конструктивный способ решать уравнен...

Задача 3. Не совсем понял, но попробую. Если форм b_n (для n листьев/атомов), то различных деревьев $(n+... Да — ты правильно почувствовал, что что-то съехало. Ошибка именно в том ключевом месте, которое мы... "Метки ставятся на атоме" - теперь понятно. Да, тот же ряд. Задача 4. Если у нас только один лист (X), то... Да — ты почти всё схватил, и здесь есть две небольшие, но важные поправки. Во-первых, да: дыркой може... Ага, не совсем решил, но идеи правильные. Что-то понял. Вопрос о zippers. Почему мы собираем контекст?... Да. Самая короткая формулировка такая:...$

Да, но в нашем случае, наши атомы только листья, поэтому мы поддеревья не редактируем. Фактически, м... Да — и здесь ты поймал ещё одну важную тонкость. Я раньше немного смешал две разные производные/д... Я примерно понимаю, что ты хочешь сказать, но для активного понимания это слишком сложно. Мы можем... Да, это разумная коррекция курса. И я согласен с твоей самооценкой: во втором листке направление мысл... 1,2,3; X означает один лист (простейшее дерево, состоящее из одного листа) - база рекурсии.

Да, точно. 1, 2, 3 и X =один лист, то есть один атом размера 1. Причём ты хорошо добавил: это ещё и база... Два независимых друг от друга поддерева, причём корень мы не считаем.

Да. Точнее: $B^2=B \times B$ — это упорядоченная пара независимых поддеревьев: левое и правое. Корень здесь... $k+m$ листьев. В OGF: член $x^{(k+m)}$ отвечает за деревья с $k+m$ листьями и получить его можно так: слева 0... Да, идея правильная: размеры складываются, а коэффициенты при одинаковой общей степени...

Ага. То есть, если есть правое поддерево, то есть и левое, и наоборот? Например, линейная конструкция... Да, всё верно. И на твой вопрос: в нашей модели линейная конструкция невозможна. Мы рассматриваем... "потому что при произведении двух классов размеры просто складываются"

Да. И добавим вторую половину мысли: мы считаем только формы, без распределения различных меток...
 $n! \times b_n$ - произвольная нумерация листьев

Да, точно: $n!b_n$ потому что на каждой из b_n форм можно расставить n различных меток по n листьям $n!$...
 $n!$ сокращаются и получаем тот же ряд, что и для OGF.

Да, именно: $n \geq 1 \sum n!b_{n-1} \times n! \times n! = n \geq 1 \sum b_n \times n!$. То есть EGF помеченных деревьев совпала как формальный ряд с OG...
Только лист. Удалять можно только атом.

Да. Именно: $B' =$ дерево с одной дыркой вместо листа потому что производная берётся по X , а X у нас...
Звучит двусмысленно. Чтобы сделать шаг вверх, ничего запоминать не надо. Нужно запомнить, чтобы име...

Да, согласен: формулировка была плохая. Переформулирую точнее. Задача 9. Пусть у нас есть дерево с...
Вот из-за таких вопросов ученики и путаются. Давай я сформулирую за тебя (ибо твоя формулировка мне в...

Да. Твоя формулировка значительно лучше и педагогически чище. И ответ именно такой:...

Тип $\text{Seq}[2B]$, то есть $[\]$ или $[2B]$ или $[2B, 2B]$ или ...

Да, точно. $B' = \text{SEQ}(2B)$ то есть $B' = 1 + 2B + (2B)^2 + (2B)^3 + \dots$ Смысл: $[\]$ — между дыркой и корнем нет ни одного...
 $B' = 1 + 2B \times B'$

Да, ровно: $B' = 1 + 2BB'$ Пустой контекст даёт 1, непустой — первый слой $2B$ и оставшийся контекст B' . И...
 $B' = 1/(1-2B)$, что, собственно, и есть уравнение для $\text{Seq}(2B)$

Да. Именно: $B' = 1 - 2B1$ и это ровно $\text{SEQ}(2B) = 1 + 2B + (2B)^2 + \dots$. То есть мы замкнули круг тремя способами:...

Да. Темп я бы чуть-чуть ускорил. Что-то среднее между 5 задачами (раньше, чересчур трудно) и 12 (сейчас...)

Да, 9 - похоже на хороший темп. Сделаем так: около 9 задач на листок, каждая - не микро-вопрос, но и не...
Встречный вопрос: а что такое корневые помеченные деревья? Что в них особенного? Какие у нас атомы?...

Да — это как раз тот вопрос, который надо было поставить до формулы. Здесь модель заметно отличается...
Ага. Причём одна вершина/атом ещё и дополнительно выделена как корень. И это X в нашей формуле....

Да — формула правильная: $T = X \cdot \text{SET}(T)$. И ты очень правильно заметил место, которое кажется...

Ага пара (дерево, вершина) всё объясняет. Задача 2. У нас не просто форма. У нас нумерация. Другая...

Да — основная причина выбрана правильно: при произведении помеченных структур надо распределить...
Верно. Я забыл про автоморфизмы. Задача 3.1: упорядоченный список - это A^k . 3.2: чтобы забыть порядок...

Да, именно так. $\text{SET}_k(A) \leftrightarrow k! A(x)^k$ и затем $\text{SET}(A) \leftrightarrow \sum_{k \geq 0} k! A(x)^k = e^{A(x)}$. Маленькая техническая оговорка:...
 $T = x \cdot \exp(T)$, то есть, X переходит в x , а Set в $\exp$.

Да: $T(x) = x e^{T(x)}$. И объяснение ровно такое: $X \rightarrow x$, потому что корень — один атом размера 1; $\text{SET}(T) \rightarrow e^T(x)$,...
 t_n - это количество пар (дерево, вершина) с учётом маркировок. Хмм, мы так формулу построили - не...

Да, ты прав: вопрос был плохо поставлен и почти тавтологичен. Твой ответ полный: $t_n = \# \dots$

$\phi(u)$ это $\exp(u)$, $\phi^n(u)$, это $\exp(nu)$. Коэффициент: $n^{n-1}/(n-1)!$ То есть, коэффициент при x^n - это $n^{n-1}/(n-1)!$...

Да — и ты за один раз фактически решил задачи 6 и 7. Из $T = x e^T$ по Лагранжу: $[x^n]T = n! [u^{n-1}] e^u = n! \dots$

U' - у нас есть дырка (может быть в любой вершине). XU' - пара: атом (вершина) и помеченное дерево с...

Да — это как раз важная тонкость species, и твое сомнение очень уместно. Ключевой момент: в...

Вот это-то меня и смущает: XU' предполагает независимость X и U' . В то же время у нас X однозначно...

Да! Именно слово «независимы» здесь и вводит в заблуждение. Для произведения species это не означает,...
А на комбинаторику это не влияет? В смысле $|X \times Y|$ должно быть равно $|X| \times |Y|$

Да — и здесь как раз надо очень строго развести обычное декартово произведение множеств и...
"Вот почему для EGF произведение species соответствует обычному произведению функций." А, ключевой...

Да - именно. Это действительно ключевой момент №3: species-product $\leftrightarrow$ ordinary product of EGF не потому,...
Хорошо. Продолжаем. Уравнение есть.

Да. У нас уже есть два ключевых уравнения: $T = x e^T$ и $xU' = T$. Значит $U' = xT$. А из первого уравнения $xT = e^T$...
 $T' = \exp(T) + x \cdot \exp(T) \cdot T'$, то есть, $\exp(T) = T'/(1+xT')$. Что-то не вижу ничего знакомого..

Да, твоя алгебра верна, но ты пошёл в менее удобную сторону. Вместо того чтобы решать относительно e^T ...
 $T - T^2/2$

Да: $dxd(T - 21T^2) = T' - 2T' = e^T$. А у нас уже было $U' = e^T$. Значит $U' = dxd(T - 21T^2)$. Следовательно, $U = T - 21T^2 + C$...

Если меток нет, то и деревьев нет, то есть ноль и тут и там и константа равна 0

Да. Поэтому $U = T - 21T^2$. И теперь это уже получено тобой по цепочке: $xU' = T$, $T = xe^T$, $U' = e^T$, $T' = e^T + TT'$, $e^T = T'$... Хорошо. $U = T - T^2/2$. Это что-то значит геометрически/топологически?

Да — и здесь есть очень красивый смысл, почти буквально эйлера характеристика дерева. Т считает...

Да, $V - E = 1$, это очевидно, и связь с T и $T^2/2$ тоже очень ясная. Ну что ж, следующий листок.

Отлично. Листок 4. Разбиения множества: Bell, Stirling и производная Здесь будет примерно 9 задач. Главн... Атомом является элемент/метка. Один блок имеет тип $\exp(X)-1$ - множество непустых атомов. Разбиение -...

Да — по существу ты сразу решил несколько шагов вперёд. Но здесь важно снова развести тип и его EGF. Н... У нас есть пара: любое множество меток (в том числе пустое, если был один элемент и теперь на его месте...

Да, именно. $P' = \text{SET}(X) \cdot P$ Комбинаторно: бывший блок с дыркой после удаления дырки превращается в...

$P' = \exp(x) \cdot P$. С другой стороны, $P' = \sum (B_{n+1}/n! \cdot x^n)$. То есть, $B_{n+1}/n! = \sum (1/k! \cdot B_{n-k}/(n-k)!)$,...

Да, всё правильно. Из $P'(x) = \exp(x)P(x)$ получаем $n \geq 0 \sum B_{n+1} n! x^n = (k \geq 0 \sum k! x^k) (m \geq 0 \sum B_m m! x^m)$. Поэтому B_{n+1} ...

Было $\exp(\exp(x)-1)$, где $\exp(x)-1$ отвечает за число элементов. Мне кажется, $\exp(u \cdot (\exp(x)-1))$, но я не уверен.

Да, именно: $P(x, u) = \exp(u(\exp(x)-1))$. Почему: один непустой блок по-прежнему даёт $\exp(x)-1$, а если каждый блок...

Комбинаторно мы выкидываем целый блок и заменяем его дыркой [знаю, но не уверен почему]. Если после...

Да — здесь как раз важно остановиться: ты смешал две разные производные. Производная по X у species...

Пассивно понятно, активно не очень. Возможно, у нас ещё будут примеры частного дифференцирования в...

Да, именно так и сделаем. Сейчас достаточно удерживать только правило на уровне узнавания:...

Хорошо. Продолжим.

Продолжим с более устойчивого места. Задача 6 Пусть $S(n, k)$ — число разбиений n -элементного множества...

1. Не считая блока с $\setminus^*$ у нас n элементов в $k-1$ блоке (взаимно-однозначное соответствие). 2. Не считая $\setminus^*$ у...

Да, точно: $S(n+1, k) = S(n, k-1) + kS(n, k)$. И ты очень хорошо сформулировал второй случай: после удаления $\setminus^*$...

$dP/dx = \exp(u \cdot (\exp(x)-1)) \cdot u \cdot \exp(x) = P \cdot u \cdot \exp(x)$; $dP/du = \exp(u \cdot (\exp(x)-1)) \cdot (\exp(x)-1) = P \cdot (\exp(x)-1) = \dots$

Да, всё точно: $\partial \partial P = u(\partial u \partial P + P)$. И это очень приятное уравнение: слева мы «добавляем один атом», а справ...

dP/dx : $S(n+1, k)$, uP : $S(n, k-1)$, udP/du : $kS(n, k)$. Да, работает.

Да. И это как раз хороший активный пример того, что раньше было только «пассивно понятно»: $\partial u \partial \dots$

$P(x, 1) = P(x)$, то есть сумма $S(n, k)$ $k \geq 0 = B_n$.

Да, именно: $P(x, 1) = P(x)$ и потому для каждого n $B_n = k=0 \sum S(n, k)$. То есть число Белла просто суммирует...

Хорошо. Попробуем Endofunctions.

Хорошо. Листок 5. Endofunctions Пусть E - класс отображений $f: S \rightarrow S$ конечного помеченного множества S в...

1. Потому что множество конечное - несколько странный вопрос. Если бы все вершины были бы разные, то ...

Да. Первый подпункт действительно был слишком тривиальным; можно было сразу спросить...

1. В последовательности может быть префикс, затем первое появление повторяющейся вершины, затем...

Да, именно. И это уже правильное структурное доказательство. У каждой вершины есть единственный...

Сус(T) - цикл, в котором каждый элемент представляет корневое дерево.

Да, точно: $C = \text{CYC}(T)$ где C — тип одной связной компоненты endofunction. Почему не SEQ и не SET: не SEQ,...

$\text{Set}(\text{Cyc}(T))$

Да: $E = \text{SET}(\text{CYC}(T))$. Это главный структурный результат всего листка. Теперь Задача 5. Если $A(x)$ — EGF...

- A^k - Цикл длины k можно повернуть k раз [то есть, он посчитан k раз], A^k/k , $k \geq 1 - \sum (A^k/k)$, $k \geq 1 \dots$

Да. Всё правильно, только маленькая терминологическая поправка: $k \geq 1 \sum k A^k = -\log(1-a)$ обычно называют...

$\text{Set} \rightarrow \exp$; $\exp(-\ln(1-T)) = 1/\exp(\ln(1-T)) = 1/(1-T)$, то есть $\text{Seq}(T)$

Да — вычисление совершенно правильное: $E(x) = 1 - T(x)$. Но вот последний шаг «то есть $\text{SEQ}(T)$ » нужно...

Вопрос: я понимаю, что одинаковые EGF не означают одинаковые структуры. С другой стороны, можно...

Да — здесь очень красивый ответ: как species — нет; как классы на $[n]$ после выбора порядка меток — да. Т...

Задача 7. Таких функций, очевидно, n^n (любое в любое), и это коэффициент.

Да, именно. $|E[n]| = n^n$ потому что каждый из n элементов независимо выбирает один из n образов. А...

То есть, есть список корневых деревьев и общее количество вершин равно n . Добавим объединяющий...

Да — и здесь твоя попытка очень полезна, потому что она показывает, где я в задаче 8 сформулировал...

Хорошо, но как это естественно объяснить через коэффициенты для T ? Мне казалось, что должен быть пут...

Да! Твоя интуиция была правильной: есть очень естественный путь через объединение списка корневых...

Несколько тяжеловесно, но красиво. Формула с двойной суммой по тяжести и внешнему виду напоминает...

Да — сходство не случайное. Но точнее сказать так:...

Хорошо. Продолжим?

Да. Осталась как раз хорошая заключительная задача листка про endofunctions — она соединит...

1. E , то есть $SET(CYC(T))$ 2. T' (?) 3. $SEQ(T)$ 4. $E' = E \times SEQ(T) \times T'$ (?)

Да — всё четыре пункта правильно. Получилось $E' = E \cdot SEQ(T) \cdot T'$. И это очень хороший пример цепного прави...

Похоже, мы закончили с эндофункциями. Какие у тебя планы на следующие листки?

Да. Я бы теперь не расширял тему бесконечно, а сделал несколько целевых листков, каждый с одной ново...

Хорошо. Продолжаем. Листок 6.

Хорошо. Листок 6. Повторение и перенос Здесь новых понятий почти не будет. Цель - посмотреть, что уже...

1. A : лист, B : вершина, 2. $A: T = X + T^2$ [единственный лист или вершина и два поддерева.] $B: T = \text{пары}...$

Да, почти всё правильно. Две небольшие поправки — обе важные. Для A структурно лучше сказать $T = X + T^2...$

1 SET, 2 SEQ, 3 CYC, 4 произведение.

Да, все четыре правильно. Для 4-го пункта точнее: $SET \geq 1(X) \cdot P$ если речь о разбиении с одним выделенным...

1 $\exp(A)$: суммируем кортежи A длиной $k = 0$ $[A^k]$, $\inf$ и удаляем порядок, то есть делим на $k!$, $2 \frac{1}{(1-A)} - ...$

Да, всё три формулы и объяснения правильные: $SET(A) \leftrightarrow e^A$ $SEQ(A) \leftrightarrow 1 - A$ $CYC(A) \leftrightarrow -\log(1-A)$ И да, для...

Произведение species это не просто произведение обычных рядом. Сначала нам нужно распределить метк...

Да, именно. Я бы только уточнил одну фразу: обычное произведение EGF как раз и кодирует этот...

Давай не будем торопиться. Если я могу что-то сказать по поводу EGF, это не значит, что я всё понимаю...

Да. Ты сейчас подошёл к очень важному месту, и почти всё у тебя правильно. Но в рассуждении про EGF ес...

Честно говоря, не вижу разницу между твоим подходом и моим. В обоих случаях у нас есть $n!$ перестаново...

Да. И сначала про $n!$: ты прав — между твоим подходом и моим по существу разницы нет. Я слишком...

Ага. А я недоумевал: сначала говорим о дырке в структуре, а потом получаем новую структуру, причём она...

Да, именно. Это очень важное прояснение. Есть жёстко определённый объект $F'[U] = F[U \sqcup \{*\}]$, то есть $F...$

Да, снимает, спасибо! Продолжаем?

Да. Задача 6. Производная vs pointing Пусть T — наши корневые помеченные деревья на n вершинах. Разбе...

1. T структура с одним специальным атомом $*$ (или её эквивалентное представление).• 2. Формальн...

Да, всё правильно. Особенно важно последнее: $T = \text{дерево с выделенным корнем}$ а...

1. Число объектов одно и то же. 2. Естественного изоморфизма между species может не существовать (хот...

Да, ровно так. Можно заключить, что для каждого n число объектов одинаково: $|E[n]| = |SEQ(T)[n]|$. Нельзя...

3 $\Rightarrow$ 2 $\Rightarrow$ 1. Причём, равенства нет. Но, что конкретно добавляется сказать трудно, не повторяя...

Да. И тут есть важная поправка к самой лестнице: $3 \Rightarrow 2 \Rightarrow 1$ но если в пункте 2 понимать просто «существует...

Ага, вот и теория категорий.

Да — буквально. Здесь теория категорий появляется не как внешняя абстракция, а потому что мы уже...

Замечательно. Продолжаем? Листок 7 или всё ещё Листок 6?

Всё ещё Листок 6: у нас осталась одна заключительная задача. После неё — Листок 7 про маркеры и частн...

По порядку: лес, возможно пустой; цикл, где у каждой вершины своё дерево; дерево с дополнительно...

Да, всё четыре правильно. Чуть точнее второй пункт: $CYC(T)$ — это цикл корневых деревьев: сами элемент...

То есть, в структуре у нас есть n атомов, которые разбиты на k компонент. Мы выделили одну из компонен...

Да — здесь есть глубокий смысл, и это не просто случайное совпадение формул. На самом деле одна из...

Ощущение такое, что мы просто выделяем (выдаём флажок на помахать), но для атомов, выделенный ато...

Да — твоя интуиция здесь очень точная. И по x , и по u мы в сущности делаем одно и то же: одну из...

"выделить, но продолжать считать её обычной" - неудачная формулировка, мне кажется. "выделить, но...

Да, согласен. Лучше именно так: ∂x — выделить один атом и перестать учитывать его переменной x ; $x \partial x$ —...

Да! Продолжаем?

Да. Продолжаем Листок 7. Задача 2. Разбиения с маркером числа блоков U нас $P(x,u)=\exp(u(\exp(x)-1))$

Только что обсудили [если я правильно понял вопрос]: один блок выделяем и его не считаем с остальными....

Да, именно так. Для разбиения с k блоками: $u \partial^k$ даёт $k u k-1$: один блок выделен, остальные $k-1$ продолжа...

А есть разница между этой задачей и предыдущей? Один атом выделяем (заменяем дыркой/*) и этот атом ...

Да — по существу это та же самая схема, только u считает блоки, а x — атомы. Твоя формулировка точная:...

Я бы сказал неупорядочено. Выделяем два блока (красный флажок и синий флажок, то есть разные...

Да — флажки разные. Это как раз важная тонкость. Если у объекта k блоков, то $u \partial^2 u k = k(k-1) u k-2$. Числ...

Да, верно, $k(k-1)$ объясняет упорядоченность пары. Иначе бы было $C(k, 2) = k(k-1)/2$.

Именно. $u \partial^2$ даёт упорядоченную пару различных блоков: $k(k-1)$. А $2! u \partial^2$ даёт неупорядоченную пару: $(2k...$

Выделяем один цикл, но считаем всё равно. И, выделяем два цикла, но неупорядоченно, и считаем всё...

Да. Возьмём совсем маленький пример: перестановки трёх элементов. По числу циклов они распределяются...

Ага, теперь понятно. Ассоциация: а нельзя ли как нибудь из этого получить теорию Burnside-Pólya? Мы, в...

Да — и не просто «можно»: это один из естественных выходов из species прямо в Burnside-Pólya. До сих пор...

Да, я как раз хотел попросить добавить листок в конце на эту тему. Хорошо, продолжаем.

Да. Burnside-Pólya добавляем отдельным поздним листком. Продолжаем Листок 7. Задача 6. Перестановки ...

$\exp(-\log(1-x)) = 1/(1-x) = \sum (x^n)$, а нам надо $\sum (u^k x^n)$, $k, n \geq 0$

Да — ты почти дошёл, но u надо вставить до внешней экспоненты, потому что вес u получает каждый цикл...

$-u \times \ln(1-x) \times P$, считает количество перестановок на n элементов, причём каждый цикл имеет вес u , причём...

Да — из закрытой формулы это действительно не видно, пока не прочитать множители комбинаторно. Ест...

Если так посмотреть, то понятно. Хорошо, продолжим.

Хорошо. Тогда следующая задача — уже с содержательным результатом. Задача 8. Сколько всего циклов в...

Да. Перемножаем $\sum (x^m/m)$ и $\sum (x^r)$. Коэффициент при x^n это сумма коэффициентов при $x^{1/1} \times x^{(...$

Да, именно. $u \partial^2 P u = 1 = 1 - x - \log(1-x) = n \geq 1 \sum n x^n$. Но это EGF, значит...

Комбинаторно мы фиксируем два цикла (если они есть), то есть после подстановки мы получим при x^n не...

Да, именно. И здесь появляется очень полезное понятие: факториальные моменты. Если K — число циклов...

Хорошо. Эту тему мы ранее [практически] не разбирали и она мне не очень знакома, может перед листком...

Я бы сделал гибрид: сначала очень короткая обзорная лекция на 5-10 минут, чтобы появилась карта...

Ты это говоришь уже не в первый раз: "при обычном условии $\backslash(G(0)=0)$: внутренняя структура не должна...

Да — лучше всего увидеть это на совсем конкретном контрпримере. Возьмём $F=SEQ, G=1+X$. $G=1+X...$

Понятно. Хорошо, можно продолжать.

Тогда начинаем Листок 8: композиция species. Будем идти постепенно, с пояснениями по мере надобности....

У меня есть довольно фундаментальное непонимание (надо все мои текущие непонимания как-то собрать...

Да — здесь действительно смешались два разных понятия: атом как объект с идентичностью и значение,...

Хорошо, посмотрим на новую структуру: у нас есть одно разбиение с тремя блоками $A1:\{1\}$, $A2:\{2\}$, $A3:\{3\}$...

Ты понял структуру почти совершенно правильно; ошибка только в последнем шаге — в том, что считает...

Приблизительно понял: строим структуру на $3x$ уникальных атомах (можно считать, что значение атома - э...

Да — почти. Единственное, что я бы поправил: фраза «с точностью до автоморфизмов» здесь немного...

$SEQ(SET())$ использует автоморфизмы на уровне SET $\{[1,2] = \{2,1\}\}$. У SEQ автоморфизмов нет. Вот если бы...

Да — именно так, с одной важной оговоркой: лучше писать $SET(SET \geq 1(X))$, чтобы внутренние блоки не был...

Хорошо. Я не уверен, что всё понимаю, но продолжим. Для этого листка есть смысл идти очень медленно и...

Да. Для этого листка лучше сильно снизить темп и не вводить Bell polynomials, пока сама композиция speci...

$G = x + x^2/2 + x^3/6 + \dots$ $G^2 = e^{2x} - 2e^x + 1 = 1 + 2x + 4x^2/2 + 8x^3/6 - 2x - 2x^2/2 - 2x^3/6 + 1 [+...$

Да, всё правильно. Только в строке для G^2 , конечно, подразумевалось $G^2=(\exp(x)-1)^2=\exp(2x)-2\exp(x)+1$. И ты получ...

$1/6+1/2+1/6 = 1/6+3/6+1/6 = 5/6$, то есть 5 неизоморфных структур. Делители появляются потому, что...

Почти всё правильно. Арифметика: $6!+2!+6!=65$, поэтому число структур на фиксированном множестве...

Давай обсудим терминологию. Для меня множество - это кортеж с забытым порядком, то есть, добавляем...

Да, так гораздо лучше сформулировать. В нашем контексте твой взгляд $SET_k = SEQ_k / S_k$ вполне рабочий:...

Хорошо. Продолжаем.

Тогда следующий шаг - один конкретный коэффициент, без общей формулы. Задача 3 Оставим $G(x) = e^x - 1$...

Разбиений типа $1+3$: $4 = C(4, 1)$; типа $2+2$: $6 = C(4, 2)$, итого: 10. Коэффициент при x^4 $7/24$. Не сходится.

Вот как раз хорошая ошибка: она показывает именно ту симметрию, которую мы хотели поймать. Для типа...

A, верно. Достаточно выбрать пару к 1 (2, 3, или 4).

Да, именно. Для разбиений типа $2+2$ можно избежать двойного счёта так: зафиксировать атом 1 и выбрать...

Если 1 слева, то 4 варианта. Если 1 справа, то $C(4, 2) = 6$. Итого: 10. Или, просто выбираем 2 из 5: $C(5, 2) = 10$.

Да, именно: $(25)=10$. И здесь важно, почему не делим ещё на 2!: блоки размеров 2 и 3 различимы уже по...

Выбрать одноэлементный блок: 5. Умножить на 3 (это мы уже проходили). Итого: 15.

Да: $5 \cdot 3 = 15$. Теперь посмотрим, как точно то же самое возникает внутри EGF. Это полезно, потому что здесь...

Просто из соображений симметрии: $6!/(2!^3 \cdot 3!) = 15$. Комбинаторно: выбираем пару для 1 (5 вариантов); 2...

Да, всё три способа совпали — и это как раз знак, что механизм уже начинает становиться естественным....

1. Формула: $7!/(1!^2 \cdot 2!)/(2!^1 \cdot 1!)/(3!^1 \cdot 1!) = 7!/(2 \times 2 \times 6) = 210$ 2. $j!^{m_j}$ - это внутренние симметрии блок...

Да, всё верно: $(1!)^{22}(2!)^{11}(3!)^{11}7! = 245040 = 210$. И объяснение знаменателя тоже правильное. Только од...

3,1,1 и 2,2,1

Да, ровно эти два типа: $3+1+1$ и $2+2+1$. Теперь посчитаем их по нашей формуле. Для $3+1+1$: $m_1=2, m_3=1, \dots$

Каждой двухэлементной структуре соответствует множитель 2, то есть вес разбиения $2+2+1$ увеличится в...

Да, именно. Тип $3+1+1$ вообще не меняется, потому что двухэлементных блоков там нет: 10. А в типе $2+2...$

Хорошо.

Тогда теперь соединяем внутренний и внешний уровни. Пусть $F(y) = k \geq 0 \sum f_k k! y^k$, $G(x) = j \geq 1 \sum g_j j! x^j$. Мы уже...

$f_k = k!$, $a_3 = 1 \times 1 + 2 \times 3 + 6 \times 1 = 13$

Да, совершенно верно: $f_k = k!$, потому что на фиксированном множестве из k блоков SEQ задаёт один из $k!$...

Один и тот же блок: $F'(G)G''$. В F только один блок затронут, зато дважды. В G выделены два элемента...

Да, именно так: $H''(x) = F'(G(x))G''(x) + F''(G(x))(G'(x))^2$ И твоя комбинаторная расшифровка правильная. В...

Хорошо. Поехали.

Тогда Задача 9: третья производная как разбиение трёх флажков. Пусть снова $H(x) = F(G(x))$. Теперь у нас т...

Три компонента: $F'''(G)G'^3$ - затронуты три разных блока, по одной дырке в каждом. Всё симметрично....

Да, формула получилась совершенно правильно: $H''' = F'''(G)(G')^3 + 3F''(G)G'G'' + F'(G)G'''$. Только я бы чуть...

Для меня это был довольно сложный листок, но, думаю, это просто моя особенность - в принципе, я...

Да, и я бы не списывал это на твою «особенность». Листок 8 действительно был концептуально плотным:...

Не уверен, что я понял твои вопросы. Давай попробуем разобраться. Обычная производная по атому: прост...

Да. И здесь твоё выражение показывает, что я слишком резко ввёл новую терминологию. Производную по...

Контекст на атоме с это, например, два списка $[a, b]$ и $[d]$. Контекст для подсписка $[c, d]$ - это два фрейма а...

Да, именно. Только я бы чуть уточнил представление второго контекста. Для атома c : $[a, b, c, d] \rightarrow ([a, b], [d])$

Да, я понимаю.

Тогда переходим к дереву Каталана. Задача 2. Один frame бинарного дерева Пусть $B = X + B^2$, то есть дерев...

L: (мы пришли слева, R), для R - наоборот.

Да, ровно так. Для узла $Node(L, R)$ один frame хранит: если фокус был в L: «пришли слева» + сохранить R;...

З, как последовательность фреймов. Я думаю, это всё довольно ясно (в отличие от предыдущего листка), и...

Да. Тогда оставшуюся часть листка 9 можно сильно ускорить и сразу выйти на главное. Для рекурсивного...

$\Phi_Y = 2XY$. $A = 2XD$. $C = SEQ(2XD)$. $\Phi_X = 1 + Y^2 = 1 + D^2$. $D' = C \times \Phi_X = SEQ(2XD) \times (1 + D^2)$. В отличие от...

Да, всё правильно. И здесь как раз хорошо видно различие между локальной частью вокруг выбранного...

Очень хороший листок (9)! Крайне эстетично и проясняет мозги! ОК, листок 10.

Тогда Листок 10: производные Бржозовского. Здесь действительно будет другая производная, но...

b и b

Да, верно: $Da((a+b)b) = b, Db((a+b)b) = b$. Это уже хороший пример того, что разные прочитанные префиксы...

$D_a(R) = a*b$, $D_b(R) = \epsilon$, $nullable(R) = false$

Да, всё верно: $Da(a*b) = a*b$, потому что после одной прочитанной a задача остаётся той же самой. $Db...$

$D_a(ast\ b) = D_a(ast)b + D_a(b) = ast\ b + \text{пусто} = ast\ b$

Да, именно: $Da(a*b) = Da(a*)b + Da(b)$. Поскольку $Da(a*) = a*$, $Da(b) = \emptyset$, получаем $Da(a*b) = a*b$. Теперь очень...

$Da(ast(a+b)) = Da(ast)(a+b) + Da(a+b) = ast(a+b) + \epsilon = ast + ast\ b$

Да, всё верно. $Da(a*(a+b)) = a*(a+b) + \epsilon$. И твое упрощение тоже правильное:...

$b(ab)^*$ и Пустое множество

Да, оба ответа верны: $Da((ab)^*) = b(ab)^*$, потому что $Da(ab) = b$, а значит по правилу звезды $Da((ab)^*) = Da(ab...$

$b(ab)^*$, $(ab)^*$ - принимающее Φ (пустое множество)

Да. Это ровно три состояния: $A = (ab)^*$, $B = b(ab)^*$, $\emptyset$. Переходы: AaB , $Ab\emptyset$, BbA , $Ba\emptyset$, $\emptyset a\emptyset$, $\emptyset b\emptyset$. Принимающее...

$Da = R + b$, $Db = R$

Да, оба ответа правильны: $Da(R) = R + b$, $Db(R) = R$, где $R = (a+b)^*ab$. Разбор первого: $Da((a+b)^*ab) = Da...$

$Da(Q) = Q$, $Db(Q) = R + \epsilon = R$

Почти: первая производная верна, $Da(Q) = Da(R + b) = Da(R) + Da(b) = Q + \emptyset = Q$. А вот во второй есть важная...

$Da(P) = Q$, $Db(P) = R$

Да, верно: $Da(P) = Q$, $Db(P) = R$, потому что $Da(\epsilon) = Db(\epsilon) = \emptyset$. И теперь весь автомат замкнулся на трёх...

Q , Q , Q , P . Состояние соответствующее w (?)

Да: $a \rightarrow Q$, $aa \rightarrow Q$, $aba \rightarrow Q$, $abab \rightarrow P$. И формулировку стоит сделать чуть точнее: $Dw(R) = \{z: wz \in L(R)\}$. То есть $Dw(...$

Может не стоит уходить в теорию автоматов. Это отдельная тема. Мой вопрос про производные регулярны...

Да, я согласен. Здесь связь оказалась гораздо слабее, чем я первоначально надеялся использовать...

То есть, осталось два листка? Можешь дать общий список листков (пройденных и оставшихся)? Просто...

Да, по текущему плану осталось два листка. Списки: рекурсивные типы, производящие функции и...

Хорошо. Тогда, если можешь, собери те ключевые моменты, которые я не понимал в силу того, что явно он...

Да. Я бы выделил шесть таких моментов. Это именно не «трудные теоремы», а скрытые соглашения языка...

7. Суть разницы между OGF и EGF. Как работает счёт и EGF на метках. Собственно, ты говоришь об этом в 4...

Да. Это действительно отдельный пункт, и, пожалуй, один из самых важных. OGF и EGF отличаются не "те...

Хорошо. Продолжим. Листок 11.

Итак, Листок 11. Синтез. Здесь идея другая: я не буду заранее говорить, какой именно инструмент нужен...

1. $SET(SEQ(X))$, 2 $\exp(1/(1-x))$, 3. EGF естественна, поскольку наши атомы помечены, и, при подсчёте, важны...

Почти всё верно; одна важная поправка в пп. 1-2. Поскольку блоки непустые, надо писать $SET(SEQ \geq 1(X))$ а...

Да, я пропустил слово непустые. 1 $SET(CYC(X), \geq 1)$, 2 $\exp(-\log(1-x)) = 1/(1-x)$

Да. И здесь даже проще: $CYC(X)$ уже по определению непустой, так что достаточно $SET(CYC(X))$. EGF:...

1. Как я понимаю, имеется в виду множество множеств. $SET(X, \geq 1) \times SET(SET(X, \geq 1), \geq 0)$, 2...

Да, всё правильно. В species-нотации я бы записал чуть компактнее: $SET \geq 1(X) SET(SET \geq 1(X))$ — первый...

1. Исходно у нас $SET(SET(X, \geq 1))$. Мы выделяем атом, то есть, мы выделяем блок, а в нём атом, который...

Почти, но здесь как раз важная ошибка: блок, содержащий выделенный атом, после выделения распадается...

Да, и я неправильно посчитал производную, иначе бы поправил.

Да, именно. Если бы сразу продифференцировать $P(x) = \exp(ex-1)$, то $P'(x) = ex \exp(ex-1)$ сразу подсказало б...

1. Выделенная вершина является корневой (n вариантов): $X \times SET(T)$. Плюс, выделенная вершина где-то в...

Да, по существу всё правильно. Особенно хорошо, что ты не просто написал pointing XT , а разложил его по...

1. $CYC(X) \times SET(CYC(X))$ 2. Формально: $-\log(1-x)/(1-x)$, 3. dP/du , $u=1$: $-\log(1-x)/(1-x)$ - то же самое.

Да, всё правильно: $CYC(X) \cdot SET(CYC(X))$ — один выделенный цикл плюс неупорядоченное множество...

1 EGF, 2 то же самое, 3 формально да, но теперь все коэффициенты ряда умножаются и делятся на $n!$, то...

Да, всё верно. Для помеченных листьев естественна EGF. Структурное уравнение species остаётся тем же:...

1. Исходная структура (без выделения): $SET(SEQ(X, \geq 1))$. С выделением (выбираем список, место и атом):...

Здесь как раз обнаружилась важная неоднозначность, и мой текст задачи её спровоцировал. Я имел в виду...

Да, это я промахнулся- читать надо внимательней: ((

Да, тут именно чтение условия, не математика. Причём твоя интерпретация была вполне содержательной ... Хорошо. Продолжим.

Тогда начинаем Листок 12. Burnside-Pólya и cycle index series. Здесь я бы снова дал короткое предисловие,...
 $1/6 \times (p_1^3 + 3 \times p_1 \times p_2 + 2 \times p_3)$

Да, совершенно верно: $ZSET,3=61(p_1^3+3p_1p_2+2p_3)$. Теперь сразу увидим, зачем эта штука нужна. Задача 2...

а) $1/6 \times x^3$, б) x^3 ; в первом случае нетождественные перестановки обнуляются, поскольку содержат хотя...

Да, всё именно так. Особенно хороша последняя формулировка: $p_1c_1p_2c_2 \dots p_j = x_j x_{c_1} + 2c_2 + 3c_3 + \dots = x_n$, пото...
 $1/P(j^m_j \times m_j!)$, не вижу без записи, а записать здесь сложно :((

Да, первый шаг правильный: $j!j^m_j m_j!$. С учётом монома это вклад $j!j^m_j m_j!$. А дальше как раз можно...

Мне переход от суммы произведений к произведению сумм очевидным не кажется.

Да, тут я слишком быстро проскочил. Это не специфическое свойство cycle index — это просто многократно...

Хорошо, давай сделаем подстановки и попробуем получить из Z_set что-нибудь интересное. Те две...

Да — только одно важное уточнение: $ZSET(x,0,0,\dots)=ex$ — это EGF помеченного SET, а $ZSET...$

$$(r+b)(r^2+b^2) = r^3 + r^2b + rb^2 + b^3$$

Да. Тогда весь числитель: $(r+b)^3=r^3+3r^2b+3rb^2+b^3$, $3(r+b)(r^2+b^2)=3r^3+3r^2b+3rb^2+3b^3$, $2(r^3+b^3)=2r^3+2b^3...$

И что получается после подстановки?

Получается: $41[(r+b)^4+(r^2+b^2)^2+2(r^4+b^4)]$. Раскрываем: $(r+b)^4=r^4+4r^3b+6r^2b^2+4rb^3+b^4,...$

У меня сложилось впечатление, что Z_set появилось несколько немотивировано (просто догадка), а потом...

Да — согласен. Я слишком быстро сделал главный концептуальный переход. В результате $ZSET$ выглядел...

Очень хорошо. Спасибо!! Если мы закончили с листками я постараюсь всё реорганизовать с твоей помощью.

Да, листки на этом закончены. Дальше действительно имеет смысл перейти от последовательности...



Я бы хотел обсудить связи с перечислением типов данных и смежные вопросы: типы данных (например, списки, деревья двоичные или нет, с метками или нет, и т.п.) - сериализация (например, деревья Каталана в скобочные структуры - рекуррентные соотношения - уравнения для порождающих функций [и интересные тождества, например Seven Trees in One] - порождающие функции - дифференцирование порождающих функций- дифференцирование типов данных [связаны ли эти две операции?] -что ещё я забыл?. У нас уже был пример для деревьев Каталана. Можно ли его [кратко!] переписать, показывая эти связи. А потом, мы можем попробовать другой пример [деревья?]. Или может стоит начать с карманного примера [списки?]. Пока не надо ничего делать. Просто задай вопросы, что выработать общую точку зрения на то, что я бы хотел получить.

10/1/2026, 2:57:57 PM



Да, это хорошая тема, и её лучше сначала спроектировать, иначе она быстро расплзётся в species, типы, производящие функции, теорию категорий и программирование одновременно.

Я бы уточнил несколько вещей.

1. Тебе нужна “карта понятий” или рабочий мини-курс?

То есть мы хотим получить обзор вроде:

тип данных → рекурсивное уравнение → порождающая функция → коэффициенты → биекции/сериализация

или хотим на каждом примере реально проходить весь цикл и считать?

2. На каком языке удобнее держать типы?

Например:

$$L = 1 + XL$$

для списков,

$$B = 1 + B^2$$

для бинарных деревьев,

$$T = X \cdot \text{SET}(T)$$

для помеченных деревьев.

Это ближе к combinatorial species. Но можно говорить менее формально: “пустой список или голова + хвост”, “лист или пара деревьев”, “корень + множество поддеревьев”.

3. Хотим ли различать ordinary generating functions и exponential generating functions сразу?

Это важно, потому что:

- непомеченные линейно/плоско организованные объекты часто ведут к обычным ПФ;
- помеченные структуры естественно ведут к экспоненциальным ПФ;
- Cayley trees n^{n-2} и Joyal species лучше объяснять через EGF/species, а Catalan - через OGF.

Можно сначала дать интуитивное различие, а технические детали добавлять позже.

4. Насколько глубоко идти в “дифференцирование типов данных”?

Есть красивая линия:

$$\frac{d}{dX} \text{ типа данных}$$

означает “структура с дыркой” или “one-hole context”.

Например, производная списка даёт контекст списка с выделенным местом; производная дерева даёт zipper. Это напрямую связано с McBride, Huet zipper, species derivative. Вопрос: хочешь ли ты это как центральную тему или как красивое ответвление?

5. Начать лучше со списков или сразу с деревьев?

Списки - идеальный карманный пример:

$$L = 1 + XL, \quad L(x) = 1 + xL(x), \quad L(x) = \frac{1}{1-x}.$$

Потом производная:

$$L'(x) = \frac{1}{(1-x)^2}$$

и типово это “список слева + список справа”, то есть контекст с одной дыркой.

Но деревья Каталана богаче:

$$B = 1 + B^2$$

или, если листья считаются атомами,

$$B = X + B^2.$$

Там появляются скобки, сериализация, Catalan numbers, ассоциативность, Seven Trees in One.

6. Нужно ли включать программирование явно?

Например, Haskell-ish псевдокод:

```
data List a = Nil | Cons a (List a)

data Tree = Leaf | Node Tree Tree
```

И рядом математическое уравнение типа:

$$L = 1 + XL, \quad T = 1 + T^2.$$

Или лучше держаться чисто математического языка?

7. Хочешь ли ты, чтобы мы связали это с задачами Арнольда 46-48?

Тогда естественный маршрут:

46 : Catalan / binary trees / triangulations

47 : labeled leaves, unordered binary trees

48 : Cayley trees / Prüfer / arbitrary labeled trees

И дальше поверх этого - типы данных, serialization, generating functions, species.

Мой предварительный совет: начать со **списков** как игрушечной модели, затем переписать **Catalan trees** в той же схеме, а потом взять **Cayley trees** как пример, где обычная схема ломается и нужны помеченные структуры / species / Prüfer или Joyal.

10/1/2026, 2:58:02 PM · gpt-5-5-thinking



1. Я бы предпочёл рабочий мини-курс: простой пример, пример с деревьями Каталана, более сложный пример [если получится]. 2. Алгебраический [более формальный, чем неформальное описание], но с неформальными пояснениями. 3. Какие функции [естественно] работают в каждом конкретном примере, те и надо использовать. 4. Да, я знаю, потому и спросил. Как красивое ответвление. Собственно, меня интересует [кроме общей темы перечисления] конкретный вопрос: нельзя ли получить перечисление более сложных структур из перечисления более простых за счёт использования дифференцирования? 5. Сначала списки как карманный пример, потом деревья Каталана, как известный пример, потом что-то посложнее. 6. Нет программирование не надо - только математика, 7. Если получится связать естественным образом, то да. Но это необязательно. И согласен с твоим советом. Давай начнём со списков и посмотрим, насколько плавно [без шороховатостей] всё вытекает друг из друга. Если что, поправим.

10/1/2026, 3:10:53 PM



Отлично. Тогда я бы с самого начала выбрал язык **комбинаторных видов (species)**, но использовал его очень экономно. Он как раз позволяет без швов пройти

структура $\longrightarrow$ алгебраическое уравнение $\longrightarrow$ порождающая функция $\longrightarrow$ дифференцирование структуры.

Для списков всё получается почти идеально гладко.

1. Минимальный словарь

Обозначим через

X

один атом, а через

1

пустую структуру.

Операции имеют буквальный комбинаторный смысл:

$A + B$

- структура либо типа A , либо типа B ;

$A \cdot B$

- упорядоченная пара: одна структура A и одна структура B , построенные на непересекающихся наборах атомов.

Пока этого достаточно.

2. Список

Пусть L - тип конечных линейных списков.

У списка есть два случая:

- он пуст;
- он состоит из первого элемента и оставшегося списка.

Поэтому неформальное описание

«пустой список или атом + список»

становится настоящим алгебраическим уравнением:

$$L = 1 + XL.$$

Это уже первый важный момент: **рекурсивное определение типа данных становится алгебраическим уравнением.**

Отсюда формально

$$L = \frac{1}{1 - X}.$$

И можно даже разложить:

$$L = 1 + X + X^2 + X^3 + \dots.$$

Комбинаторно это означает именно то, что должно означать:

- один пустой список;
- список из одного атома;
- список из двух;
- список из трёх;
- ...

3. Перечисление

Теперь заменяем формальный атом X переменной x .

Если элементы **помечены**, на n различных элементах имеется

$$n!$$

линейных порядков.

Поэтому естественна экспоненциальная производящая функция:

$$L(x) = \sum_{n \geq 0} n! \frac{x^n}{n!} = \sum_{n \geq 0} x^n = \frac{1}{1 - x}.$$

И алгебраическое уравнение типа

$$L = 1 + XL$$

буквально переходит в

$$L(x) = 1 + xL(x).$$

То есть уже здесь виден общий принцип:

$$\text{комбинаторные операции} \longrightarrow \text{алгебраические операции над ПФ.}$$

Для списков случай слегка обманчиво прост: OGF для одной непомеченной формы каждого размера дала бы ту же функцию $1/(1 - x)$. Позже это совпадение исчезнет.

4. Где здесь сериализация?

Список уже практически является своей собственной сериализацией.

Структура на наборе

$$\{a, b, c, d\}$$

есть, например,

$$(c, a, d, b).$$



То есть комбинаторный объект кодируется словом, содержащим каждый ярлык ровно один раз.

Но гораздо интереснее следующий вопрос:

Что получится, если в списке выделить одно место?

И вот здесь появляется дифференцирование.

5. Производная типа

Определим L' как:

список, в котором один специальный атом заменён «дыркой».

Например,

$$a\ b\ \square\ c\ d.$$

Что представляет собой такая структура?

Ровно **два списка**:

(всё слева от дырки, всё справа от дырки).

Поэтому чисто комбинаторно

$$L' = L^2.$$

Это уже замечательно само по себе.

Но теперь продифференцируем исходное уравнение как обычное алгебраическое:

$$L = 1 + XL.$$

Получаем

$$L' = L + XL'.$$

Отсюда

$$L'(1 - X) = L,$$

а поскольку

$$L = \frac{1}{1 - X},$$

то

$$L' = L^2.$$

То есть **формальное дифференцирование алгебраического уравнения дало ровно ту операцию над структурами, которую мы получили, вырезав один элемент и оставив дырку.**

Это не совпадение.

6. То же происходит с производящей функцией

У нас

$$L(x) = \frac{1}{1 - x}.$$

Дифференцируем:

$$L'(x) = \frac{1}{(1 - x)^2} = L(x)^2.$$



Три утверждения совпали:



Вот это, думаю, и есть первый ответ на твой конкретный вопрос:

Да - дифференцирование типов данных и дифференцирование производящих функций связаны буквально.

В языке species это одна из основных конструкций.

7. Но надо различать «дырку» и «выделенный элемент»

Есть две очень близкие операции.

Производная L'

Мы добавляем специальный элемент $*$, строим структуру, а затем считаем $*$ дыркой.

Для списка:

$$a, c, *, b \rightsquigarrow (a, c) \mid (b).$$

Pointing

Берём обычный объект и **выделяем один из уже существующих атомов**.

Алгебраически это

$$\boxed{XL'}.$$

Почему дополнительный X ? Потому что дырку снова заполняем настоящим атомом.

Поэтому для любой структуры размера n pointing создаёт n вариантов.

На уровне ПФ:

$$\boxed{x F'(x)}$$

- производящая функция объектов с выделенным атомом.

Это различие потом будет полезно.

8. Можно ли производной получать более сложные структуры?

Да - и список уже даёт минимальный пример:

$$\boxed{L' = L^2}.$$

«Более сложная» структура - список с контекстом/дыркой - получена дифференцированием исходного типа.

Но ещё интереснее бывает наоборот: производная сложного рекурсивного типа неожиданно раскладывается в произведение более простых типов. Именно отсюда возникают **zippers** и всевозможные контексты деревьев.

Например, для дерева производная будет описывать:

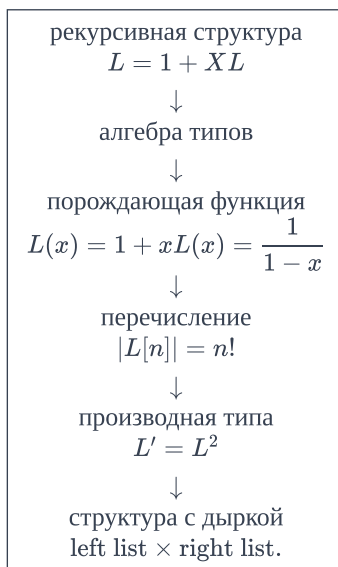
дерево с одной дыркой,

а это можно интерпретировать как **путь от дырки к корню плюс данные, висящие сбоку от этого пути**.

Вот там дифференцирование уже начинает что-то действительно открывать, а не просто подтверждать очевидное.

9. Маленькая схема, которую стоит сохранить

Для списка весь наш будущий мини-курс уже виден в микромасштабе:



Это получилось, по-моему, достаточно гладко.

И следующая проверка для нашей схемы должна быть гораздо содержательнее: **бинарные деревья Каталана**. Там

$$T = X + T^2$$

даст числа Каталана, скобочную сериализацию и производящую функцию; а когда мы продифференцируем это уравнение, должна появиться очень конкретная комбинаторная структура - **бинарное дерево с одной дыркой**, то есть древовидный контекст. Именно там мы впервые увидим, приносит ли дифференцирование реально новую информацию, а не только красивую аналогию.

10/1/2026, 3:11:05 PM · gpt-5-6-thinking



То есть, производная списков - это декартово произведение списков?

10/1/2026, 3:21:18 PM



Да — **в точности**, если под L понимаем species линейных списков.

$$L' \cong L \times L$$

Комбинаторный смысл такой: производная L' — это список с одной **дыркой**. А список с дыркой однозначно распадается на:

(список слева от дырки, список справа от дырки).

Например,

$$[a, b, \square, c, d, e] \longleftrightarrow ([a, b], [c, d, e]).$$

И обратно: из любой пары списков (u, v) получаем

$$u \square v.$$

Поэтому это не просто совпадение чисел, а настоящая **биекция типов**.

Важно только не спутать с множеством всех пар списков на произвольных независимых множествах элементов. В species-произведении $L \times L$ исходное множество меток разбивается на две непересекающиеся части: одна идёт в левый список, другая — в правый. Именно это соответствует одному исходному списку с удалённым элементом.

И производящая функция честно повторяет эту биекцию:

$$L'(x) = L(x)^2.$$

Так что здесь слово «декартово произведение» вполне правильно — **произведение двух комбинаторных видов списков**.

10/1/2026, 3:21:22 PM · gpt-5-6-thinking





Хорошо. Давай перейдём к деревьям Каталана.

10/1/2026, 3:22:01 PM



Отлично. Здесь схема уже становится намного содержательнее.

Возьмём **полные плоские бинарные деревья**: вершина либо лист, либо имеет ровно двух упорядоченных потомков. Будем считать листья атомами X . Тогда сам тип B задаётся рекурсивно:

$$B = X + B^2.$$

Неформально: дерево - либо один лист, либо упорядоченная пара деревьев.

1. Сериализация: скобки

Это рекурсивное определение сразу даёт каноническую запись. Лист a записываем как a , а пару деревьев u, v как

$$(u\ v).$$

Например,

$$((ab)(cd))$$

есть бинарное дерево с четырьмя листьями. Поэтому те же объекты можно читать как **полные расстановки скобок**:

$$((ab)c)d, \quad (ab)(cd), \quad a(b(cd)), \dots$$

То есть уже имеем биекцию

$$\text{бинарные деревья} \longleftrightarrow \text{полные скобочные структуры}.$$

Для n листьев таких форм C_{n-1} .

2. Рекурсия Каталана возникает прямо из типа

Если в корне внутренний узел, его n листьев разбиваются: k слева и $n - k$ справа. Поэтому

$$b_n = \sum_{k=1}^{n-1} b_k b_{n-k}, \quad b_1 = 1.$$

Это та же Catalan convolution.

На уровне производящей функции просто заменяем X на x :

$$B(x) = x + B(x)^2.$$

Отсюда

$$B(x) = \frac{1 - \sqrt{1 - 4x}}{2}$$

и

$$B(x) = x + x^2 + 2x^3 + 5x^4 + 14x^5 + \dots$$

То есть коэффициент при x^n есть

$$C_{n-1}.$$

Если листья снабжены различными метками, формально мы работаем со species и EGF; число структур на n метках равно $n!C_{n-1}$, а EGF всё равно получается именно $B(x)$. Это приятное совпадение между геометрической формой дерева и маркировкой.

3. Теперь дифференцируем тип

Из

$$B = X + B^2$$

формально:

$$B' = 1 + 2BB'.$$

Что это означает комбинаторно?

B' - бинарное дерево, у которого **один лист вынут и оставлена дырка**:



Если дырка - всё дерево, получаем член 1.

Иначе в корне есть два поддерева. Дырка находится:

- либо в левом, а справа висит обычное дерево B ;
- либо в правом, а слева висит обычное дерево B .

Отсюда буквально

$$B' = 1 + BB' + B'B = 1 + 2BB'.$$

То есть обычное правило производной произведения здесь имеет настоящий структурный смысл: дырка должна попасть **в один из двух множителей**.

4. А что представляет собой B' целиком?

Решим:

$$B' = \frac{1}{1 - 2B}.$$

То есть

$$B' = 1 + (2B) + (2B)^2 + (2B)^3 + \dots$$

И это удивительно наглядно.

Чтобы описать дерево с дыркой, идём **от дырки вверх к корню**. На каждом шаге надо запомнить две вещи:

1. было ли наше поддерево слева или справа - 2 варианта;
2. какое обычное дерево висело с другой стороны - B .

Поэтому контекст есть просто последовательность:

$$B' \cong \text{Seq}(2B).$$

Это и есть знаменитый **tree zipper** в чисто математическом виде: путь от выделенного места к корню плюс соседние поддерева.

Здесь производная уже действительно **обнаружила новую структуру**.

5. Производящая функция говорит то же самое

Дифференцируем

$$B(x) = \frac{1 - \sqrt{1 - 4x}}{2}.$$

Получаем

$$B'(x) = \frac{1}{\sqrt{1-4x}}.$$

Но

$$\frac{1}{\sqrt{1-4x}} = \sum_{n \geq 0} \binom{2n}{n} x^n.$$

Значит деревья Каталана **с одной дыркой** перечисляются центральными биномиальными коэффициентами:

$$1, 2, 6, 20, 70, 252, \dots$$

И это можно увидеть без анализа:

дерево с $n + 1$ листьями можно взять C_n способами и выбрать один из $n + 1$ листьев как специальный. Поэтому

$$(n + 1)C_n = \binom{2n}{n}.$$

Вот уже очень хороший пример ответа на твой исходный вопрос:

может ли дифференцирование типа дать естественное перечисление более сложной структуры?

Да.

$$\text{Catalan tree} \xrightarrow{d/dX} \text{tree context} / \text{tree with a hole}$$

и одновременно

$$C_n \xrightarrow{\text{pointing/differentiation}} (n + 1)C_n = \binom{2n}{n}.$$

6. Что произошло концептуально

У нас теперь получилась совершенно гладкая цепочка:

$$\begin{array}{c} B = X + B^2 \\ \downarrow \\ \text{бинарные деревья / скобки} \\ \downarrow \\ B(x) = x + B(x)^2 \\ \downarrow \\ C_n \\ \downarrow d/dX \\ B' = 1 + 2BB' \\ \downarrow \\ \text{дерево с дыркой} \\ \downarrow \\ \text{Seq}(2B) \\ \downarrow \\ B'(x) = \frac{1}{\sqrt{1-4x}} \\ \downarrow \\ \binom{2n}{n}. \end{array}$$

И здесь уже видно, что «дифференцирование типа» - не метафора. **Алгебра типов, комбинаторная операция с дыркой и обычное дифференцирование производящих функций - три проекции одной операции.**

Следующим естественным шагом я бы ещё немного задержался на этом примере и посмотрел на **pointing** XB' и на связь с *Seven Trees in One*. После этого уже можно переходить к более сложному типу деревьев, где появятся EGF и, вероятно, уравнение вида $T = Xe^T$.

10/1/2026, 3:22:12 PM · gpt-5-6-thinking



Какие алгоритмические задачи решает tree zipper? То есть, какие конкретно алгоритмы [для каких задач] его используют?

10/1/2026, 3:29:12 PM



Да. Самая практическая идея zipper'a очень проста:

он превращает дерево + выбранную вершину в структуру, по которой можно ходить локально и менять выбранное место, не теряя остального дерева.

То есть вместо

$$T$$

мы храним

$$(\text{focus}, \text{context}),$$

где `focus` — текущее поддереву, а `context` — путь к корню вместе с «боковыми» поддеревьями.

Для бинарного дерева context как раз имеет вид последовательности элементов «я пришёл слева/справа + соседнее поддереву».

Это решает несколько вполне конкретных алгоритмических задач.

- **Редактирование дерева.** Найти узел, заменить его поддереву, затем восстановить всё дерево. В неизменяемой структуре данных без zipper приходится хранить рекурсивный стек или заново строить путь к корню. С zipper путь уже материализован как данные.
- **Навигация по AST** в компиляторах и системах переписывания: перейти к дочернему узлу, к родителю, заменить выражение, сделать локальное преобразование и подняться обратно. Например, было

$$(a + (b \cdot c))$$

и мы сфокусировались на $b \cdot c$; zipper хранит контекст $a + \square$.

- **Term rewriting / symbolic algebra.** Правило применяется к подтерму:

$$f(g(x), y) \rightsquigarrow f(\square, y)[g(x)].$$

Можно переписать $g(x)$, а затем автоматически вставить результат обратно в исходный контекст.

- **Proof assistants / логические термы.** Очень похожая задача: выбрать подформулу или подтерм, применить к нему преобразование, сохранив окружающий контекст.
- **Итератор по дереву.** Zipper фактически содержит явный стек обхода. Поэтому можно реализовать `next`, `previous`, `parent`, `first child`, не имея parent pointers в самом дереве. Это особенно удобно для persistent/immutable trees.
- **Иерархические редакторы:** XML/HTML, JSON-подобные деревья, outline editors, меню, файловая иерархия. Текущий «курсор» — это focus; zipper позволяет двигаться вверх/вниз и локально редактировать.

Есть и очень конкретная связь с обычными алгоритмами поиска. Например, при вставке в бинарное дерево поиска мы спускаемся к месту вставки. Рекурсивная реализация неявно держит стек:

куда потом вернуться и какое соседнее поддереву сохранить.

Zipper просто превращает этот **стек вызовов в явную структуру данных**. После вставки мы идём обратно вверх и пересобираем дерево. Для persistent BST это особенно естественно.

Можно даже сказать:

zipper = дерево + курсор + стек обратного пути

но стек устроен не произвольно: каждый его элемент содержит ровно те данные, которые нужны, чтобы восстановить родителя.

И вот тут связь с производной становится содержательной. Если

$$T'$$

есть «дерево с дыркой», то это именно **тип всех возможных контекстов, в которые можно вставить поддереву**:

$$\text{context} \times \text{subtree} \longrightarrow \text{tree}.$$

То есть производная типа отвечает на весьма алгоритмический вопрос:

какую информацию надо сохранить, если мы вырезали один компонент структуры, но хотим потом без потерь вставить его или другой компонент обратно?

Для списка ответ был $L \times L$: всё слева и всё справа.

Для бинарного дерева — путь к корню + боковые ветви.

Это, пожалуй, самое конкретное объяснение того, зачем вообще «производная типа данных» нужна алгоритмически.

10/1/2026, 3:29:24 PM · gpt-5-6-thinking



Хорошо. Давай, как ты предложил, посмотрим на pointing и на связь с Seven Trees in One.

10/1/2026, 3:32:13 PM



Да. И здесь полезно разделить две близкие, но всё-таки разные линии: **pointing** живёт внутри перечислительной комбинаторики, а *Seven Trees in One* — скорее в алгебре рекурсивных типов. Они встречаются потому, что обе начинаются с одного и того же структурного уравнения.

Для наших каталановских деревьев

$$B = X + B^2$$

атомами являются листья.

Производная B' — дерево с **дыркой вместо одного листа**. Pointing означает: дырку снова заполняем настоящим атомом и считаем этот атом выделенным. Поэтому

$$B^\bullet = XB'$$

Комбинаторно это просто:

бинарное дерево с одним выделенным листом.

Мы уже нашли

$$B' = \frac{1}{1 - 2B},$$

значит

$$B^\bullet = \frac{X}{1 - 2B}.$$

А на уровне производящих функций

$$B(x) = \frac{1 - \sqrt{1 - 4x}}{2},$$

поэтому

$$B^\bullet(x) = xB'(x) = \frac{x}{\sqrt{1 - 4x}}.$$

Если дерево имеет n листьев, то способов выделить лист ровно n . Поэтому

$$[x^n] B^\bullet(x) = nC_{n-1} = \binom{2n-2}{n-1}.$$

То есть pointing превращает каталановы числа в центральные биномиальные коэффициенты со сдвигом:

$$C_{n-1} \longrightarrow nC_{n-1} = \binom{2n-2}{n-1}.$$

Это уже хороший пример того, как **операция над структурами автоматически порождает новое перечисление**.

Теперь связь с *Seven Trees in One*.

Если мы перестаём следить за размером и смотрим просто на **тип всех конечных бинарных деревьев**, то имеем

$$T \cong 1 + T^2.$$

Здесь 1 — лист, а T^2 — внутренняя вершина с левым и правым поддеревом.



Формально, если обращаться с этим как с обычным алгебраическим уравнением,

$$T^2 = T - 1,$$

откуда

$$T^3 = -1,$$

и затем

$$T^6 = 1, \quad \boxed{T^7 = T}.$$

Но здесь возникает проблема: в мире типов нет объекта « -1 ». Мы умеем складывать и перемножать типы, но вычитание вообще не обязано существовать.

И вот именно это делает *Seven Trees in One* интересным:

$$\boxed{T^7 \cong T}$$

оказывается реализуемо **настоящей биекцией**, несмотря на то что формальное кольцевое доказательство использует отрицательные величины.

То есть история такая:

$$\boxed{T \cong 1 + T^2 \xRightarrow{\text{формально}} T^7 = T}$$

и затем возникает глубокий вопрос:

если тождество типов получается после разрешения вычитания, можно ли реализовать его положительной биекцией, используя только реальные комбинаторные конструкции?

У Blass–Lawvere ответ в этом случае — да.

Здесь важно не смешать это с производящей функцией.

Для перечисления мы писали

$$B = X + B^2.$$

Для *Seven Trees in One* пишем

$$T = 1 + T^2.$$

Это **не означает**, что мы просто подставили $X = 1$ в производящую функцию. Это другое действие: мы **забыли градуировку по размеру** и рассматриваем весь рекурсивный тип целиком.

Так что я бы нарисовал две ветви от одного источника:

$$\boxed{\text{recursive tree type}}$$

в одну сторону:

$$B = X + B^2 \rightarrow \text{Catalan} \rightarrow B' \rightarrow XB' \rightarrow \text{pointed trees},$$

а в другую:

$$T = 1 + T^2 \rightarrow \text{algebra of types} \rightarrow T^7 \cong T \rightarrow \text{Seven Trees in One}.$$

И вот здесь возникает уже очень интересный общий сюжет:

комбинаторные структуры можно складывать, умножать, дифференцировать и иногда решать с ними алгебраические уравнения

Только надо помнить: **сложение и произведение имеют буквальный комбинаторный смысл; производная тоже имеет буквальный смысл через дырку; а вычитание уже опасно**. *Seven Trees in One* именно поэтому столь необычно: оно показывает, что иногда «незаконное» формальное вычитание угадывает совершенно законную биекцию.

10/1/2026, 3:32:37 PM · gpt-5-6-thinking



Да. Переходим к деревьям?

10/1/2026, 3:36:40 PM



Да. И теперь я бы выбрал **произвольные помеченные деревья**, но начал с **корневых** - это самый естественный следующий пример.

Причина очень хорошая: для корневого дерева рекурсивная структура прозрачна:

$$T = X \cdot \text{SET}(T)$$

то есть дерево = корень X + неупорядоченное множество корневых поддеревьев.

Для помеченных структур операция SET превращается в экспоненту, поэтому EGF удовлетворяет

$$T(x) = xe^{T(x)}.$$

Отсюда по формуле Лагранжа:

$$[x^n]T(x) = \frac{n^{n-1}}{n!},$$

то есть корневых помеченных деревьев на n вершинах

$$n^{n-1}.$$

А теперь появляется особенно красивая связь с нашим вопросом о **дифференцировании типов**.

Пусть U - вид обычных, некорневых помеченных деревьев. Тогда выделить в таком дереве одну вершину - значит сделать его корневым:

$$U^\bullet = XU' = T.$$

На уровне EGF:

$$xU'(x) = T(x).$$

А это немедленно означает: если корневых деревьев n^{n-1} , то некорневых в n раз меньше:

$$n^{n-2}.$$

То есть формула Кэли появляется буквально как **интегрирование/“снятие pointing”**:

$$n^{n-1} \longrightarrow n^{n-2}.$$

Более того, из $xU' = T$ и $T = xe^T$ получается

$$U'(x) = e^{T(x)}.$$

А после интегрирования, используя уравнение для T ,

$$U(x) = T(x) - \frac{1}{2}T(x)^2.$$

Это уже очень красивый пример, потому что тут сходятся сразу четыре линии:

$$\text{рекурсивный тип} \rightarrow \text{EGF} \rightarrow \text{Лагранж} \rightarrow \text{pointing / derivative} \rightarrow \text{Cayley}.$$

И это как раз естественно связывает нашу мини-теорию с задачей Арнольда №48.

Я бы поэтому следующий кусок построил именно так:

1. почему $T = X \text{SET}(T)$;



2. почему SET даёт e^T ;
3. получить $T = xe^T$;
4. извлечь n^{n-1} ;
5. затем через $XU' = T$ получить n^{n-2} ;
6. и только после этого посмотреть, что означает уже T' - корневое дерево с дыркой - и какую структуру оно кодирует.

Это будет заметно богаче списков и Каталана, но всё ещё очень гладко.

10/1/2026, 3:36:52 PM · gpt-5-6-thinking



Хорошо. Пойдём по пунктам. Я бы сказал, что некоторые пункты я понимаю (1,4,5), но всё равно хотелось бы посмотреть объяснение.

10/1/2026, 5:22:23 PM



Тогда действительно по одному пункту. Начнём с **1. Почему**

$$T = X \cdot \text{SET}(T)$$

для корневых помеченных деревьев.

Здесь T - вид конечных корневых деревьев, вершины которых несут различные метки. X означает одну вершину - корень.

Теперь удалим корень из дерева. Дерево распадётся на несколько связных компонент. Каждая компонента сама становится **корневым деревом**: её корень - та вершина, которая раньше была непосредственно соединена с удалённым корнем.

Например, схематически



после удаления верхней вершины превращается просто в набор

$$\{T_1, T_2, T_3\}.$$

Почему именно SET, а не SEQ? Потому что наше дерево **не плоское**: у детей вершины нет порядка «первый, второй, третий». Перестановка T_1, T_2, T_3 ничего не меняет.

Поэтому дерево состоит из

$$\underbrace{X}_{\text{корень}} \times \underbrace{\text{SET}(T)}_{\text{неупорядоченное множество поддеревьев}}.$$

И наоборот, это полностью обратимо: берём один атом X , берём конечное множество корневых деревьев и соединяем корень каждого из них с новым атомом. Получаем ровно одно корневое дерево.

То есть это настоящая биекция типов:

$$T \cong X \times \text{SET}(T).$$

Где здесь метки

Это существенно. Пусть набор меток

$$\{1, 2, 3, 4, 5\},$$

а корень имеет метку 3. После его удаления оставшиеся метки могут, например, разбиться на блоки

$$\{1, 5\}, \quad \{2\}, \quad \{4\}.$$

На каждом блоке живёт своё корневое дерево. Поэтому $\text{SET}(T)$ означает не просто «какое-то множество деревьев», а:

разбить оставшийся набор меток на непустые блоки и на каждом блоке построить T -структуру.

Это именно стандартное произведение/SET в теории species.

И тут уже начинает проглядывать следующий пункт. Для помеченных структур операция

SET

имеет EGF

$$e^z.$$

Поэтому структурное равенство

$$T = X \operatorname{SET}(T)$$

превратится в

$$T(x) = xe^{T(x)}.$$

Но это уже **пункт 2**, и там как раз стоит аккуратно объяснить, почему SET превращается именно в экспоненту - это, пожалуй, первая действительно нетривиальная ступень.

10/1/2026, 5:22:32 PM · gpt-5-6-thinking



Да, пункт 1 я понимаю. А вот пункт 2 нет.

10/1/2026, 5:24:13 PM



Именно здесь скрывается главный трюк **помеченной комбинаторики**. Начнём с самого простого случая, а потом обобщим.

Пусть X — один помеченный атом. Тогда

$$\text{SET}(X)$$

означает просто конечное множество помеченных атомов.

На любом фиксированном наборе из n меток такая структура ровно одна: само это множество. Поэтому числа структур:

$$1, 1, 1, 1, \dots$$

Но мы используем **экспоненциальную** производящую функцию:

$$\sum_{n \geq 0} 1 \cdot \frac{x^n}{n!} = \boxed{e^x}.$$

Уже отсюда

$$\boxed{\text{SET}(X) \longleftrightarrow e^x}.$$

Теперь заменим атом X произвольной структурой A .

Пусть

$$A(x) = \sum_{n \geq 1} a_n \frac{x^n}{n!}.$$

Я пока предполагаю, что пустых A -структур нет.

Рассмотрим сначала **упорядоченную пару** A -структур. На n метках надо:

1. выбрать k меток для первой структуры — $\binom{n}{k}$;
2. построить на них A -структуру — a_k вариантов;
3. на остальных построить вторую — a_{n-k} .

Получаем

$$\sum_k \binom{n}{k} a_k a_{n-k}.$$

А это ровно коэффициенты произведения EGF:

$$\boxed{A \times A \longleftrightarrow A(x)^2}.$$

Аналогично упорядоченная r -ка A -структур даёт

$$A(x)^r.$$

Но $\text{SET}_r(A)$ — это **неупорядоченный набор из r компонент**. Поэтому порядок r компонент надо забыть. Каждая неупорядоченная коллекция была посчитана $r!$ раз:

$$\boxed{\text{SET}_r(A) \longleftrightarrow \frac{A(x)^r}{r!}}.$$

Теперь разрешаем любое число компонент:

$$\text{SET}(A) = 1 + \text{SET}_1(A) + \text{SET}_2(A) + \dots.$$

Следовательно,

$$\text{SET}(A) \longleftrightarrow 1 + A(x) + \frac{A(x)^2}{2!} + \frac{A(x)^3}{3!} + \dots$$

и потому

$$\boxed{\text{SET}(A) \longleftrightarrow e^{A(x)}}.$$

Вот вся причина.

Есть ещё более содержательный способ это увидеть. Объект из $\text{SET}(A)$ на метках

$$\{1, \dots, n\}$$

строится так:

- разбиваем множество меток на блоки;
- на каждом блоке строим A -структуру.

Например,

$$\{1, 2, 3, 4, 5, 6\} = \{1, 4\} \sqcup \{2\} \sqcup \{3, 5, 6\},$$

а затем на этих трёх блоках строим три A -объекта.

Поэтому формула

$$\boxed{e^{A(x)}}$$

по существу кодирует **все разбиения множества меток на неупорядоченные блоки, каждый из которых несёт A -структуру.**

Это называется **exponential formula**.

Теперь к деревьям. У нас

$$T = X \cdot \text{SET}(T).$$

Корень даёт множитель x , а множество его поддеревьев —

$$e^{T(x)}.$$

Поэтому буквально:

$$\boxed{T(x) = x e^{T(x)}}.$$

То есть экспонента здесь появилась не каким-то аналитическим чудом: она просто означает

$$\boxed{\text{неупорядоченное множество помеченных компонент.}}$$

Мне кажется, это ключевая фраза, которую стоит удерживать:

$$\boxed{\text{SEQ} \leftrightarrow \frac{1}{1-A}, \quad \text{SET} \leftrightarrow e^A}$$

— первая для упорядоченной последовательности, вторая для неупорядоченного множества **в помеченном мире**.

10/1/2026, 5:24:23 PM · gpt-5-6-thinking



Очень красиво. Это было 2 и 3. Теперь 4.

10/1/2026, 6:50:31 PM



Да. Пункт 4 — это как раз место, где появляется **формула Лагранжа для обращения рядов**.

У нас

$$T(x) = xe^{T(x)}.$$

Это уравнение не даёт $T(x)$ в явном виде, но оно идеально подходит под схему

$$T = x\phi(T).$$

Для такого уравнения формула Лагранжа говорит:

$$[x^n]T(x) = \frac{1}{n}[u^{n-1}]\phi(u)^n.$$

В нашем случае

$$\phi(u) = e^u,$$

поэтому

$$[x^n]T(x) = \frac{1}{n}[u^{n-1}]e^{nu}.$$

Но

$$e^{nu} = \sum_{k \geq 0} \frac{n^k}{k!} u^k,$$

значит

$$[u^{n-1}]e^{nu} = \frac{n^{n-1}}{(n-1)!}.$$

Следовательно,

$$[x^n]T(x) = \frac{1}{n} \cdot \frac{n^{n-1}}{(n-1)!} = \frac{n^{n-1}}{n!}.$$

А поскольку $T(x)$ — **экспоненциальная** производящая функция,

$$T(x) = \sum_{n \geq 1} t_n \frac{x^n}{n!},$$

мы получаем

$$t_n = n^{n-1}.$$

То есть корневых помеченных деревьев на n вершинах ровно

$$n^{n-1}.$$

И тут хорошо видно, как вся цепочка работает без разрывов:

$$T = X \operatorname{SET}(T)$$

даёт

$$T(x) = xe^{T(x)},$$

а Лагранж переводит это функциональное уравнение в коэффициенты:

$$n^{n-1}.$$

Самая важная идея здесь такая:

рекурсивная структура даёт неявное уравнение для ПФ; формула Лагранжа извлекает из него коэффициенты напрямую.

Если хочешь, следующим сообщением я могу очень коротко показать, **почему именно формула Лагранжа имеет такой вид** в этом специальном случае, прежде чем переходить к пункту 5.

10/1/2026, 6:50:39 PM · gpt-5-6-thinking





Нет, давай перейдем к пункту 5 [хотя сам по себе он довольно очевиден, но давай проговорим]. Меня больше интересует вывод $U = T - \frac{1}{2}T^2$.

10/1/2026, 7:14:01 PM



Да. Пункт 5 сам по себе прост, а формула

$$U = T - \frac{1}{2}T^2$$

действительно гораздо интереснее. Здесь есть **два вывода**: аналитический — почти мгновенный, и комбинаторный — через теорему о диссимметрии деревьев. Начнём с первого.

Пусть U — вид **некорневых** помеченных деревьев, T — корневых.

Выделить одну вершину в некорневом дереве — значит выбрать корень. Поэтому

$$U^\bullet = XT'$$

если U' означает дырку, а более привычно на уровне EGF:

$$xU'(x) = T(x).$$

То есть

$$U'(x) = \frac{T(x)}{x}.$$

Но из

$$T = xe^T$$

следует

$$\frac{T}{x} = e^T.$$

Значит

$$U' = e^T.$$

Теперь дифференцируем уравнение $T = xe^T$:

$$T' = e^T + xe^T T'.$$

Но $xe^T = T$, поэтому

$$T' = e^T + TT',$$

то есть

$$(1 - T)T' = e^T.$$

А левая часть — это ровно

$$\frac{d}{dx} \left(T - \frac{1}{2}T^2 \right).$$

Следовательно,

$$U'(x) = \frac{d}{dx} \left(T - \frac{1}{2}T^2 \right).$$

И поскольку при $x = 0$

$$U(0) = T(0) = 0,$$

константа интегрирования нулевая:

$$U = T - \frac{1}{2}T^2.$$

Получилось удивительно гладко.

Но есть ещё более красивое **структурное объяснение**, и оно, мне кажется, тебе особенно понравится.

Для дерева можно отметить:

- вершину;
- ребро;
- ориентированное ребро.

Если отметить вершину, получаем корневое дерево:

$$\mathcal{U}^\bullet \cong T.$$

Если отметить ребро и удалить его, дерево распадается на **два корневых дерева**, причём порядок двух сторон не важен:

$$\mathcal{U}^{\text{edge}} \cong \text{SET}_2(T),$$

следовательно EGF:

$$\frac{T^2}{2}.$$

Если же ориентировать отмеченное ребро, две стороны уже различаются:

$$\mathcal{U}^{\text{directed edge}} \cong T^2.$$

И существует замечательная **теорема о диссимметрии деревьев**:

$$\mathcal{U} + \mathcal{U}^{\text{directed edge}} \cong \mathcal{U}^\bullet + \mathcal{U}^{\text{edge}}.$$

На уровне EGF это

$$U + T^2 = T + \frac{1}{2}T^2,$$

откуда снова

$$U = T - \frac{1}{2}T^2.$$

Здесь минус перестаёт быть загадочным: настоящая положительная биекция утверждает

$$U + T^2 \cong T + \text{SET}_2(T).$$

А уже после переноса члена возникает формальный минус.

Это очень в духе нашего *Seven Trees in One*: **алгебраическая формула с вычитанием скрывает положительную комбинаторную биекцию**.

Наконец, пункт 5 в перечислительном виде совсем очевиден. Корневых деревьев на n метках

$$n^{n-1}.$$

Каждое некорневое дерево имеет ровно n возможных корней, поэтому

$$u_n = \frac{n^{n-1}}{n} = \boxed{n^{n-2}},$$

то есть формула Кэли.

Таким образом, здесь у нас сразу три слоя одной истории:

pointing	:	$xU' = T,$
calculus	:	$U = T - \frac{1}{2}T^2,$
dissymmetry	:	$U + T^2 \cong T + \text{SET}_2(T).$

Последний, по-моему, особенно ценен: он показывает, **почему именно** появляется $T - \frac{1}{2}T^2$, а не просто позволяет его вычислить.

10/1/2026, 7:14:13 PM · gpt-5-6-thinking



Да. Здорово! И пункт 6: что такое T' ?

10/1/2026, 7:24:12 PM



Да. Здесь T' уже очень наглядно интерпретируется.

Напомню:

$$T = X \text{ SET}(T)$$

где T — корневые помеченные деревья.

По определению производной species,

$$T'$$

— это корневое дерево, в котором **одна вершина заменена специальной дыркой** $\square$. Дырка не считается обычной меткой, но сохраняет своё место в структуре дерева.

Например:



То есть у дырки вполне могут оставаться дочерние поддеревья.

Дифференцируем структурное уравнение

Из

$$T = X \text{ SET}(T)$$

получаем

$$T' = \text{SET}(T) + X \text{ SET}(T) T'.$$

Почему именно так, можно прочесть непосредственно.

Дырка находится либо:

1. **в корне**. Тогда после удаления корня остаётся просто множество его поддеревьев:

$$\text{SET}(T);$$

2. **ниже корня**. Тогда корень обычный, рядом с ветвью, содержащей дырку, висит неупорядоченное множество обычных поддеревьев:

$$X \text{ SET}(T) T'.$$

Но

$$X \text{ SET}(T) = T,$$

поэтому

$$T' = \text{SET}(T) + TT'.$$

И значит

$$T' = \frac{\text{SET}(T)}{1 - T}.$$

Теперь появляется очень хорошая структурная форма:

$$T' \cong \text{SET}(T) \times \text{SEQ}(T).$$

Что это значит?

Начинаем с дырки и идём от неё вверх к корню.

У самой дырки есть произвольное множество дочерних поддеревьев:

$$\text{SET}(T).$$

А затем каждый раз, поднимаясь на один уровень вверх, мы должны запомнить:

- родительскую вершину;
- все её боковые поддеревья, кроме той ветви, по которой проходит путь к дырке.

Но

$$X \text{ SET}(T) = T.$$

Поэтому каждый такой «слой контекста» сам имеет тип T .

А слои от дырки до корня естественно **упорядочены по высоте**. Поэтому они образуют последовательность:

$$\text{SEQ}(T).$$

Итак,

$$\text{дерево с дыркой} = \underbrace{\text{поддеревья под дыркой}}_{\text{SET}(T)} \times \underbrace{\text{последовательность контекстов до корня}}_{\text{SEQ}(T)}.$$

Это ровно тот же zipрег-мотив, который мы видели для бинарных деревьев, только теперь дети неупорядочены и их число произвольно.

На уровне EGF:

$$T'(x) = \frac{e^{T(x)}}{1 - T(x)}.$$

Поскольку

$$T = xe^T,$$

можно также написать

$$T'(x) = \frac{T(x)}{x(1 - T(x))}.$$

И pointing снова даёт

$$XT'$$

— корневое дерево с одной **выделенной настоящей вершиной**.

На n метках таких объектов

$$n \cdot n^{n-1} = n^n.$$

Так что здесь возникает ещё один красивый ряд:

U	:	n^{n-2}	некорневое дерево,
T	:	n^{n-1}	выбираем корень,
XT'	:	n^n	корень + ещё одна выделенная вершина.

И, пожалуй, самое приятное: T' показывает, что у произвольного дерева тот же общий принцип, что у списка и бинарного дерева:

производная = структура с дыркой = контекст, позволяющий восстановить целое.

10/1/2026, 7:24:25 PM · gpt-5-6-thinking



Извини, не понял этот пассаж: "ниже корня. Тогда корень обычный, рядом с ветвью, содержащей дырку, висит неупорядоченное множество обычных поддеревьев: $X \mathrm{SET}(T) T'$ ". Откуда взялся T' ? У нас только одна дырка, а выглядит так, как будто у всех поддеревьев есть по дырке.

10/1/2026, 7:28:56 PM



Ты прав — моя формулировка была слишком сжатой. В выражении

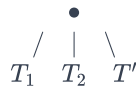
$$X \mathrm{SET}(T) T'$$

только один множитель T' содержит дырку.

Смысл такой. Если дырка находится не в корне, то у корня есть:

- сам корень: X ;
- несколько обычных дочерних поддеревьев: $\mathrm{SET}(T)$;
- ровно **одно особое дочернее поддерево**, внутри которого находится единственная дырка: T' .

То есть картинка примерно такая:



где $T_1, T_2, \dots$ — обычные деревья без дырок, а последний объект T' — одно дерево с дыркой где-то внутри.

Поэтому

$$X \mathrm{SET}(T) T'$$

означает не «все поддеревья с дырками», а

корень $\times$ множество обычных детей $\times$ один выделенный ребёнок с дыркой.

Именно потому, что дети неупорядочены, этот «дырочный» ребёнок естественно выделяется среди остальных.

Это ровно аналог правила производной

$$\frac{d}{dX} \mathrm{SET}(T) = \mathrm{SET}(T) T' :$$

в множестве компонент производная должна выбрать **ровно одну компоненту**, в которой произошло дифференцирование; все остальные остаются обычными.

Так что вся формула

$$T' = \mathrm{SET}(T) + X \mathrm{SET}(T) T'$$

читается буквально:

- дырка в корне;
- или корень цел, а **ровно в одном из его поддеревьев** находится дырка.

И да — это важное место: произведение $\mathrm{SET}(T) T'$ здесь означает «обычные поддеревья + одно специальное поддерево», а не «каждое поддерево дифференцируем».

10/1/2026, 7:29:05 PM · gpt-5-6-thinking



Да, я понял. Замечательно! А есть ещё интересные [в этом смысле] типы данных [помимо списка, дерева Каталана и просто дерева]? То есть типы, для которых есть похожие выводы с интересными результатами?

10/1/2026, 7:31:57 PM



Да, и есть несколько очень удачных примеров. Причём некоторые дают даже более красивые сюрпризы, чем деревья. Я бы выделил четыре, в таком порядке.

1. **Разбиения множества.** Это, пожалуй, лучший следующий короткий пример после деревьев:

$$\Pi = \text{SET}(\text{SET}_{\geq 1}(X)).$$

Отсюда сразу

$$\Pi(x) = \exp(e^x - 1),$$

то есть числа Белла. А производная имеет очень прозрачный смысл: у разбиения есть один блок, содержащий дырку; после удаления дырки этот блок превращается просто в произвольное множество. Поэтому

$$\Pi' = \text{SET}(X) \Pi$$

и

$$\Pi'(x) = e^x \Pi(x).$$

Из этого сразу получается классическая рекурсия для чисел Белла. Это прямо идеальный пример «дифференцирование структуры → новая комбинаторная декомпозиция → рекурсия коэффициентов».

2. **Перестановки как множества циклов.**

$$\mathcal{P} = \text{SET}(\text{CYC}(X)).$$

Поскольку

$$\text{CYC}(X) \longleftrightarrow \sum_{n \geq 1} \frac{x^n}{n} = -\log(1 - x),$$

имеем

$$\mathcal{P}(x) = \exp(-\log(1 - x)) = \frac{1}{1 - x}.$$

То есть $n!$ перестановок.

Особенно интересна производная: цикл, содержащий дырку, после удаления дырки **разрывается в линейный список**. Поэтому

$$\mathcal{P}' = \text{SEQ}(X) \mathcal{P}.$$

Очень красивый пример того, как производная меняет геометрию структуры:

цикл с дыркой $\longrightarrow$ линия.

3. **Леса корневых деревьев.** Если T — наши корневые помеченные деревья, то

$$F = \text{SET}(T).$$

Значит

$$F(x) = e^{T(x)}.$$

Но из

$$T = xe^T$$

следует

$$F(x) = \frac{T(x)}{x}.$$

Поэтому число лесов корневых деревьев на n метках равно

$$(n+1)^{n-1}.$$

И здесь есть совершенно элементарная биекция: добавляем новую вершину 0 и соединяем её со всеми корнями леса. Получаем одно дерево на $n+1$ вершинах с фиксированным корнем 0.

То есть формула

$$e^T = T/x$$

оказывается буквальной операцией «лес $\leftrightarrow$ дерево с добавленным общим корнем».

4. **Функции множества в себя — endofunctions.** Вот это я бы выбрал как наш будущий «более сложный пример».

Для отображения

$$f : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$$

рисуем стрелку $v \rightarrow f(v)$. Каждая связная компонента состоит из:

- одного ориентированного цикла;
- к каждой вершине цикла подвешено корневое дерево.

Поэтому структурно

$$E = \text{SET}(\text{CYC}(T)).$$

А значит

$$E(x) = \exp\left(\sum_{k \geq 1} \frac{T(x)^k}{k}\right) = \exp(-\log(1-T)) = \frac{1}{1-T(x)}.$$

Но ведь функций $n \rightarrow n$ очевидно

$$n^n.$$

Поэтому из структурной декомпозиции получается неожиданное тождество

$$\frac{1}{1-T(x)} = \sum_{n \geq 0} n^n \frac{x^n}{n!}.$$

Это особенно красиво после того, что мы уже сделали:

$$\begin{array}{ll} \text{деревья} & : n^{n-1}, \\ \text{некорневые деревья} & : n^{n-2}, \\ \text{endofunctions} & : n^n. \end{array}$$

И все три связаны одной структурной машиной.

Есть ещё хорошие кандидаты: связные графы через

$$\text{GRAPH} = \text{SET}(\text{CONNECTED}),$$

откуда логарифм извлекает связные объекты; плоские деревья $T = X \text{SEQ}(T)$; не пересекающиеся разбиения; necklaces; различные классы графов. Но я бы не расплылся.

Для нашего мини-курса мне видится особенно гладкая лестница:

списки → Catalan trees → labeled trees → set partitions → endofunctions.

Разбиения дадут нам ещё один очень чистый пример **производной**, а endofunctions — хороший итоговый пример, где SET, CYS, деревья и EGF соединятся сразу в одной формуле.

10/1/2026, 7:32:09 PM · gpt-5-6-thinking



Хорошо. Давай разберем set partitions. [А потом, чуть позже, и endofunctions].

10/1/2026, 7:36:37 PM



Да. Здесь всё получается особенно чисто, потому что сама структура разбиения почти буквально записывается через SET.

Пусть Π — вид разбиений конечного помеченного множества на непустые блоки.

Каждый блок — это непустое множество атомов:

$$B = \text{SET}_{\geq 1}(X).$$

А разбиение — это неупорядоченное множество таких блоков:

$$\Pi = \text{SET}(B) = \text{SET}(\text{SET}_{\geq 1}(X)).$$

Например,

$$\{\{1, 4\}, \{2\}, \{3, 5\}\}.$$

Порядок элементов внутри блока не важен, и порядок самих блоков тоже не важен — поэтому SET дважды.

Производящая функция

Для одного непустого блока:

$$B(x) = e^x - 1.$$

По exponential formula:

$$\Pi(x) = \exp(B(x)) = \exp(e^x - 1).$$

Следовательно,

$$\exp(e^x - 1) = \sum_{n \geq 0} B_n \frac{x^n}{n!},$$

где B_n — числа Белла:

$$1, 1, 2, 5, 15, 52, 203, \dots$$

То есть вся формула Белла получается непосредственно из

$$\text{partition} = \text{SET}(\text{nonempty SET}).$$

Теперь самое интересное: Π'

Π' — разбиение, в котором один атом заменён дыркой.

Где находится дырка? В каком-то одном блоке.

Удалим дырку из этого блока. Что от него останется?

Просто произвольное множество обычных атомов — возможно, даже пустое:

$$\text{SET}(X).$$

А все остальные блоки образуют совершенно обычное разбиение:

$$\Pi.$$

Поэтому возникает очень простая биекция:



$$\Pi' \cong \text{SET}(X) \times \Pi.$$

Например,

$$\{\{1, 4\}, \{2, \square, 7\}, \{3, 5\}\}$$

соответствует паре

$$(\{2, 7\}, \{\{1, 4\}, \{3, 5\}\}).$$

Здесь первая компонента — остальные элементы блока с дыркой, вторая — разбиение всего остального.

На уровне EGF:

$$\Pi'(x) = e^x \Pi(x).$$

И действительно:

$$\frac{d}{dx} e^{e^x - 1} = e^x e^{e^x - 1}.$$

Опять та же картина:

$$\text{производная типа} \leftrightarrow \text{структура с дыркой} \leftrightarrow \text{обычная производная EGF}.$$

Из производной сразу получается рекурсия Белла

Поскольку

$$\Pi(x) = \sum_{n \geq 0} B_n \frac{x^n}{n!},$$

то

$$\Pi'(x) = \sum_{n \geq 0} B_{n+1} \frac{x^n}{n!}.$$

А

$$e^x \Pi(x) = \left(\sum_{k \geq 0} \frac{x^k}{k!} \right) \left(\sum_{m \geq 0} B_m \frac{x^m}{m!} \right).$$

Сравнивая коэффициенты:

$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_{n-k}.$$

И это можно прочесть без всяких функций.

Берём множество из $n + 1$ элементов и выделяем элемент $*$. Чтобы построить разбиение:

- выбираем k из остальных n элементов, которые попадут в один блок с $*$:

$$\binom{n}{k};$$

- оставшиеся $n - k$ элементов разбиваем произвольным образом:

$$B_{n-k}.$$

Отсюда та же сумма.

Это, по-моему, очень удачный пример: **дифференцирование не просто подтверждает формулу — оно само показывает правильное комбинаторное разбиение объектов и тем самым рождает рекурсию.**

Есть и естественное refinement. Если считать разбиения не просто по размеру, но ещё и по числу блоков, вводим переменную u :

$$\Pi(x, u) = \exp(u(e^x - 1)).$$

Тогда

$$[x^n / n! u^k] \Pi(x, u) = S(n, k),$$

где $S(n, k)$ — числа Стирлинга второго рода.

То есть

$$\text{Bell} = \sum_k \text{Stirling}.$$

И здесь уже видна приятная иерархия:

$$\text{SET}_{\geq 1}(X) \rightarrow \text{SET}(\text{SET}_{\geq 1}(X)) \rightarrow e^{e^x - 1} \rightarrow S(n, k), B_n \rightarrow \Pi' = e^x \Pi.$$

А если нам всё ещё важна тема **сериализации**, то у разбиений есть хороший канонический код — *restricted growth string*. Например

$$\{\{1, 3, 5\}, \{2, 6\}, \{4\}\} \longleftrightarrow 1\,2\,1\,3\,1\,2,$$

если нумеровать блоки в порядке появления их наименьшего элемента. Условия на такую строку просты:

$$a_1 = 1, \quad a_i \leq 1 + \max(a_1, \dots, a_{i-1}).$$

Так что здесь тоже есть наша линия

$$\text{структура} \leftrightarrow \text{канонический код} \leftrightarrow \text{перечисление}.$$

Следующий особенно интересный шаг внутри этого примера — посмотреть, что означает **pointing** $X\Pi'$ и что меняется, если вместо одного выделенного элемента выделить целый блок. Это даст ещё одну хорошую иллюстрацию алгебры видов.

10/1/2026, 7:36:59 PM · gpt-5-6-thinking



Хорошо. Продолжаем.

10/1/2026, 7:42:10 PM



Да. Здесь как раз появляются две разные операции выделения, и их полезно развести.

1. Pointing: выделяем один элемент

Мы уже знаем

$$\Pi' = \text{SET}(X) \Pi.$$

Pointing означает вернуть в дырку настоящий атом:

$$\boxed{\Pi^\bullet = X \Pi'}.$$

Комбинаторно это просто разбиение с одним выделенным элементом.

Если выделенный элемент лежит в некотором блоке, то этот блок можно представить как

$$X \cdot \text{SET}(X),$$

то есть «выделенный элемент + остальные элементы его блока», а всё остальное — обычное разбиение:

$$\boxed{\Pi^\bullet = X \text{SET}(X) \Pi.}$$

На уровне EGF:

$$x \Pi'(x) = x e^x \Pi(x).$$

И на n элементах объектов ровно

$$\boxed{n B_n},$$

что, конечно, очевидно: в каждом разбиении выбираем один из n элементов.

2. А теперь выделяем не элемент, а целый блок

Это уже другая операция.

Один блок имеет тип

$$B = \text{SET}_{\geq 1}(X).$$

Разбиение с одним выделенным блоком состоит из:

- одного специального блока B ;
- обычного разбиения остальных элементов Π .

Поэтому

$$\boxed{\Pi^{\text{block}} = B \Pi = \text{SET}_{\geq 1}(X) \Pi.}$$

На уровне EGF:

$$\boxed{(e^x - 1) \Pi(x).}$$

Но мы уже знаем

$$\Pi'(x) = e^x \Pi(x),$$

следовательно

$$(e^x - 1)\Pi(x) = \Pi'(x) - \Pi(x).$$

То есть:

$$\text{EGF разбиений с выделенным блоком} = \Pi' - \Pi.$$

И отсюда сразу получается красивое тождество.

Пусть K_n — общее число блоков во всех разбиениях n -элементного множества. Тогда

$$K_n = B_{n+1} - B_n.$$

Потому что коэффициент при $x^n/n!$ в Π' равен B_{n+1} , а в Π — B_n .

То есть:

$$\sum_{\pi \in \Pi_n} \# \text{blocks}(\pi) = B_{n+1} - B_n.$$

Это уже довольно приятный результат, который буквально выпал из алгебры видов.

3. То же через дополнительную переменную

Если хотим считать ещё и число блоков, вводим u :

$$\Pi(x, u) = \exp(u(e^x - 1)).$$

Тогда

$$\frac{\partial}{\partial u} \Pi(x, u) = (e^x - 1)\Pi(x, u).$$

А при $u = 1$:

$$\left. \frac{\partial \Pi}{\partial u} \right|_{u=1} = (e^x - 1)\Pi(x).$$

То есть производная по u означает:

$$\boxed{\text{выделить один блок.}}$$

Это очень хорошая параллель:

$$\begin{aligned} \frac{\partial}{\partial x} &\leftrightarrow \text{выделить атом,} \\ \frac{\partial}{\partial u} &\leftrightarrow \text{выделить компоненту/блок.} \end{aligned}$$

Причём вторая производная по u ,

$$\frac{\partial^2}{\partial u^2},$$

считает разбиения с **двумя упорядоченно выделенными различными блоками**, и так далее.

Получается уже довольно общий принцип:

$$\boxed{\text{ввести маркер некоторого структурного компонента} \Rightarrow \text{дифференцировать по маркеру} \Rightarrow \text{выделить такой компонент.}}$$

Для atoms маркером служит x ; для числа блоков — u .

И это, пожалуй, одна из самых полезных идей всей темы: **производные ПФ — это не только техника извлечения коэффициентов, а оператор “пометить/выделить что-то внутри структуры”**.

Следующим шагом я бы уже действительно перешёл к **endofunctions**: там одновременно появятся **SET**, **CYC** и наши корневые деревья T , и получится очень красивое

$$\text{End} = \text{SET}(\text{CYC}(T)).$$

10/1/2026, 7:42:21 PM · gpt-5-6-thinking



Хорошо. Переходим к endofunctions.

10/1/2026, 7:45:03 PM



Да. Это, пожалуй, лучший «составной» пример: здесь одновременно работают **деревья, циклы, множества компонент и EGF**.

Пусть E — вид endofunctions, то есть функций

$$f : [n] \rightarrow [n].$$

Нарисуем ориентированный граф: из каждой вершины v проводится стрелка

$$v \rightarrow f(v).$$

У каждой вершины ровно одна исходящая стрелка.

1. Как устроена одна связная компонента?

Начнём из любой вершины и будем идти по стрелкам:

$$v, f(v), f^2(v), \dots$$

Так как вершин конечное число, рано или поздно какая-то вершина повторится. Значит возникает ориентированный цикл.

И в одной связной компоненте такой цикл ровно один.

К каждой вершине цикла могут быть подвешены вершины, которые в неё переходят; к ним — ещё вершины и т. д. Получается **корневое дерево**, причём стрелки направлены к корню, а корень лежит на цикле.

То есть одна связная компонента имеет вид

$$\boxed{\text{CYC}(T)}$$

где T — наш вид корневых помеченных деревьев.

А вся функция может иметь несколько связных компонент, порядок которых не важен:

$$\boxed{E = \text{SET}(\text{CYC}(T)).}$$

Вот это и есть главный структурный факт.

2. Почему $\text{CYC}(T)$ даёт логарифм?

Для помеченных структур

$$\text{CYC}(A)$$

имеет EGF

$$\sum_{k \geq 1} \frac{A(x)^k}{k} = -\log(1 - A(x)).$$

Почему $1/k$?

Упорядоченная k -ка A -структур дала бы

$$A(x)^k.$$

Но в цикле

$$(A_1, A_2, \dots, A_k)$$



и

$$(A_2, \dots, A_k, A_1)$$

— один и тот же объект. Есть k циклических сдвигов, поэтому делим на k :

$$\frac{A^k}{k}.$$

И суммируем по всем длинам цикла:

$$\text{CYC}(A) \longleftrightarrow A + \frac{A^2}{2} + \frac{A^3}{3} + \dots = -\log(1 - A).$$

Поэтому для одной связной компоненты endofunction:

$$C(x) = -\log(1 - T(x)).$$

3. Теперь множество компонент

Мы уже знаем:

$$\text{SET}(A) \longleftrightarrow e^{A(x)}.$$

Следовательно,

$$E(x) = \exp[-\log(1 - T(x))].$$

И вдруг всё схлопывается:

$$E(x) = \frac{1}{1 - T(x)}.$$

Это очень красивый результат.

4. Но endofunctions мы умеем считать напрямую

На n -элементном множестве каждый из n элементов может независимо выбрать один из n образов.

Поэтому функций

$$[n] \rightarrow [n]$$

ровно

$$n^n.$$

Следовательно,

$$\frac{1}{1 - T(x)} = \sum_{n \geq 0} n^n \frac{x^n}{n!}.$$

И это уже довольно неожиданное тождество.

Напомним, что

$$T(x) = xe^{T(x)}$$

и

$$T(x) = \sum_{n \geq 1} n^{n-1} \frac{x^n}{n!}.$$

Итак,

$$\frac{1}{1 - T(x)} = \sum_{n \geq 0} n^n \frac{x^n}{n!}.$$

То есть n^n возникло не потому, что мы просто сказали «для каждого элемента n вариантов», а ещё и как результат глубокой структурной декомпозиции:

$$\text{function} = \text{SET}(\text{CYC}(\text{rooted trees})).$$

5. Здесь особенно красиво замыкается наша серия

У нас уже были:

$$T = X \text{ SET}(T)$$

для корневых деревьев, дающее

$$n^{n-1}.$$

Некорневые деревья:

$$n^{n-2}.$$

А теперь endofunctions:

$$E = \text{SET}(\text{CYC}(T))$$

дают

$$n^n.$$

То есть числа

$$n^{n-2}, \quad n^{n-1}, \quad n^n$$

не просто стоят рядом — они соответствуют трём связанным классам структур.

6. И ещё одна приятная вещь: сериализация структуры

Endofunction можно «разобрать» канонически:

1. найти все циклы;
2. рассматривать каждый цикл как связную компоненту;
3. к каждой вершине цикла прикрепить её входящее корневое дерево.

То есть из произвольного функционального графа мы извлекаем буквально:

$$\text{SET of CYCLES of TREES.}$$

А обратно:

- берём набор циклов;
- в каждой позиции цикла сидит корневое дерево;
- корни деревьев соединяем по циклу.

Получается исходная функция.

Это настоящая биекция типов, не просто подсчёт.

И вот здесь уже можно сделать очень красивый следующий шаг в духе нашей темы:

$$\text{что означает } E'?$$

Потому что производная

$$E' = [\text{SET}(\text{CYC}(T))]'$$

имеет совершенно конкретное структурное толкование: **endofunction с одной дыркой**. И если продифференцировать SET, затем CYS, а затем T , мы увидим, как дырка последовательно выбирает

компоненту → цикл → одно дерево на цикле → место внутри дерева.

Это, по-моему, будет особенно хорошей кульминацией всей линии «дифференцирование типов».

10/1/2026, 7:45:15 PM · gpt-5-6-thinking



Это совершенно грандиозно! Спасибо за лекции! У меня есть только одна проблема: я понимаю, что ты делаешь, но вряд ли могу повторить это сам. Это как с языками: пассивное понимание vs активное использование. Я бы хотел предложить следующее: знаешь ли ты метод Константинова, он же метод листков? В этом методе предлагаются задачи, которые раскрывают материал + объяснения (до и после). Мы можем попробовать применить этот метод к какому-нибудь ответвлению (например, дифференцирование регулярных выражений). Или мы можем пройти этот курс ещё раз: с задачами. Или оба. Или ещё что-нибудь [если у тебя есть свои идеи на эту тему]. Проблема: мне будет сложно выписывать решения [ограничения клавиатуры + я не помню кодировки LaTeX], но я могу "описать" решение, а также дать ответ [а ты можешь раскрыть подробности и скорректировать решение.]

10/2/2026, 10:28:41 AM



Да, знаю именно этот подход: материал не сообщается сначала в готовом виде, а **раскладывается на последовательность задач**, каждая из которых заставляет открыть следующий кусочек; ученик идёт в своём темпе, а преподаватель проверяет и при необходимости даёт небольшой толчок. Именно так устроены опубликованные листки Константинова для матклассов. ([Math-Net.Ru](https://math-net.ru))

И для нашей темы это, по-моему, почти идеальный способ исправить проблему «всё понимаю, но сам не построю».

Я бы сделал **оба варианта**, но в таком порядке.

Сначала пройти наш нынешний курс ещё раз, уже **в листках**, без повторной лекции. Например, 4-5 листков:

1. **Алгебра структур:** $+$, $\times$, `SEQ`, `SET`; список как первый объект.
2. **Производная species:** дырка, pointing, список $\rightarrow$ Catalan trees $\rightarrow$ zipper.
3. **Помеченные структуры:** почему `SET` даёт e^A ; корневые деревья $T = X \text{ SET}(T)$; $T = xe^T$.
4. **Разбиения:** Bell/Stirling, дифференцирование, маркировка блоков.
5. **Синтез:** endofunctions, где тебе самому придётся догадаться до

$$\text{SET}(\text{CYC}(T)).$$

Причём я бы **не давал длинные листки сразу**. По 3-5 задач за раунд. Ты отвечаешь хоть так:

«Думаю, надо разбить на левую и правую части; значит что-то вроде $L \times L$ ».

Этого совершенно достаточно. Никакого LaTeX от тебя не требуется. Я превращаю это в точную запись, проверяю ход мысли и задаю следующий вопрос. Если ошибка - сначала маленькая подсказка, а не готовое решение. Это ближе к настоящему «листочному» режиму.

Особенно важно: задачи должны быть не просто упражнениями на уже известную формулу. Например, вместо «докажите $L' = L^2$ » лучше:

Из списка удалили один элемент, но оставили метку его места. Какие данные нужны, чтобы однозначно восстановить такой объект?

То есть формулу должен **родить ответ**, а не условие.

После этого я бы сделал вторую ветку - **производные регулярных выражений (Brzozowski derivatives)**. Это отличный transfer-test: тема новая, но многие формы подозрительно знакомы. Например,

$$D_a(R + S) = D_a R + D_a S,$$

а для произведения возникает аналог правила Лейбница, но с условием nullable. В конце выясняется, что производная regex - это residual language, а различные производные фактически дают состояния DFA.

Это особенно полезно потому, что мы сможем спросить:

Здесь «производная» - та же самая идея, что у *species*, или только похожая алгебра?

Ответ не совсем тривиален: **это разные операции**, и именно сравнение поможет понять обе лучше.

У меня есть ещё одна добавка к методу листков: сделать три прохода по каждой важной идее:

открыть → восстановить → перенести

Например:

- открыть $L' = L^2$ из задачи;
- через несколько дней самому восстановить её почти без подсказки;
- затем встретить похожую ситуацию в дереве или гегех и решить, что переносится, а что нет.

Именно третий шаг, мне кажется, превращает пассивное «да, это очевидно, когда вижу» в активное владение.

Так что я предлагаю **не идти дальше с лекциями прямо сейчас**, а начать сначала в листочном режиме с очень короткого **Листка 1: алгебра структур и списки**. Причём первые задачи будут простыми, но не механическими; мы довольно быстро доберёмся до производной.

Sources

Н. Н. Константинов, "Листки математического семинара для 10 класса", Матем. обр., 2007, № 1(41), 24-32 mathnet.ru

10/2/2026, 10:29:01 AM · gpt-5-6-thinking



Отлично! Поехали!

10/2/2026, 10:31:44 AM



Поехали. **Листок 1: алгебра структур и списки.** Пока без лекции; только задачи. Отвечать можно словами, формулами без LaTeX, как удобно. Я буду уточнять и оформлять.

Задача 1

Есть тип L конечных списков из атомов X .

Попробуй записать **рекурсивное уравнение для L** , используя только 1 , X , сложение и умножение типов.

Не вычисляй ничего; просто опиши, из каких двух случаев состоит список.

Задача 2

Из одного списка удалили один элемент, но **место удаления сохранили как дырку**:

$$[a, b, c, d, e] \longrightarrow [a, b, \square, d, e].$$

Какие две более простые структуры полностью определяют полученный объект?

Отсюда попробуй угадать формулу для L' .

Задача 3

Не пользуясь готовой формулой для L' , возьми своё рекурсивное уравнение из задачи 1 и **формально продифференцируй его**, пользуясь обычными правилами

$$1' = 0, \quad X' = 1, \quad (AB)' = A'B + AB'.$$

Получается ли тот же результат, что в задаче 2?

Задача 4

Теперь пусть

$$L(x) = 1 + x + x^2 + x^3 + \dots$$

Объясни **комбинаторно**, почему коэффициент при x^n равен 1, если мы пока забыли метки и считаем только форму списка.

Затем найди $L(x)$ в замкнутом виде.

Давай начнём **только с задачи 1.**

10/2/2026, 10:31:51 AM · gpt-5-6-thinking





Задача 1. L - это либо пустой список $()$, как в Haskell [то есть, 1] или пара атомхсписок. То есть, формально, в теории типов : $L = 1 + XL$. [Отсюда обычное уравнение: $L = 1 + xL$. Вопрос: а почему можно перейти от уравнения типов к обычному?]

10/2/2026, 10:36:26 AM



Да, задача 1 решена точно:

$$L = 1 + XL.$$

И твой вопрос — как раз ключевой: **почему из равенства типов можно перейти к равенству производящих функций?**

Потому что операция «взять производящую функцию» устроена так, что она сохраняет основные комбинаторные операции.

Для species:

$$A \mapsto A(x)$$

ведёт себя так:

$$\begin{aligned} 1 &\mapsto 1, & X &\mapsto x, \\ A + B &\mapsto A(x) + B(x), \\ A \cdot B &\mapsto A(x)B(x). \end{aligned}$$

То есть это, по существу, **гомоморфизм алгебры типов в алгебру формальных рядов**.

Самое нетривиальное место — произведение.

Если

$$A(x) = \sum a_n \frac{x^n}{n!}, \quad B(x) = \sum b_n \frac{x^n}{n!},$$

то структура $A \cdot B$ на n метках строится так:

- выбираем k меток для A : $\binom{n}{k}$;
- строим A -структуру: a_k способов;
- на остальных строим B -структуру: b_{n-k} способов.

Итого

$$c_n = \sum_k \binom{n}{k} a_k b_{n-k}.$$

А теперь смотри:

$$A(x)B(x) = \sum_n \left(\sum_k \frac{a_k}{k!} \frac{b_{n-k}}{(n-k)!} \right) x^n.$$

Но

$$\frac{1}{k!(n-k)!} = \frac{1}{n!} \binom{n}{k}.$$

Поэтому

$$A(x)B(x) = \sum_n \left(\sum_k \binom{n}{k} a_k b_{n-k} \right) \frac{x^n}{n!}.$$

То есть произведение series **в точности кодирует разбиение множества меток между двумя компонентами.**

Вот почему из

$$L = 1 + XL$$

можно законно перейти к

$$L(x) = 1 + xL(x).$$

Для списков это особенно гладко, потому что

$$L(x) = 1 + x + x^2 + x^3 + \dots.$$

Есть ещё одна тонкость: в этом конкретном примере та же формула получилась бы и для обычной производящей функции непомеченных форм списка, потому что формы списка каждого размера ровно одна. Но в общем случае для species естественно использовать именно **EGF**.

Коротко:

$$\text{алгебра типов} \longrightarrow \text{формальные ряды}$$

работает потому, что

$$+ \text{ и } \times$$

имеют одинаковые комбинаторные и алгебраические правила подсчёта.

Теперь можно идти к задаче 2.

10/2/2026, 10:36:41 AM · gpt-5-6-thinking



Задача 2. Нам нужно знать список слева и, независимо от него [произведение], список справа. То есть, $L' = L \times L$. $L' = L^2$.

10/2/2026, 10:39:34 AM



Да, именно так:

$$L' = L \times L = L^2.$$

Смысл совершенно точный: список с одной дыркой однозначно задаётся парой

(левый список, правый список).

Например,

$$[a, b, \square, c, d] \longleftrightarrow ([a, b], [c, d]).$$

И это настоящая биекция, не просто совпадение чисел.

Единственная тонкость в языке species: метки исходного множества делятся между левым и правым списком, то есть произведение $L \times L$ включает и выбор этого разбиения меток.

Теперь **задача 3**: возьми

$$L = 1 + XL$$

и продифференцируй формально по обычному правилу произведения. Посмотрим, получится ли снова $L' = L^2$.

10/2/2026, 10:39:39 AM · gpt-5-6-thinking





Задача 3. $L = 1/(1-X)$, $L' = 1/(1-X)^2$, то есть, $L' = L^2$. Или, $L' = L + XL'$, $L'(1-X) = L$ и опять: $L' = L^2$. Есть ли вариант без деления на $(1-X)$?

10/2/2026, 10:44:57 AM



Да — и есть очень красивый вариант, вообще без деления.

После дифференцирования:

$$L' = L + XL'$$

Теперь возьмём исходное уравнение

$$L = 1 + XL$$

и просто умножим его на L :

$$L^2 = L + XL^2$$

Сравни:

$$L' = L + XL'$$

$$L^2 = L + XL^2$$

То есть и L' , и L^2 удовлетворяют одному и тому же рекурсивному уравнению

$$Y = L + XY$$

А его решение однозначно определяется по размерам объектов: сначала коэффициент размера 0, потом размера 1, потом 2, и т. д. Поэтому

$$L' = L^2$$

Мне этот вариант даже нравится больше: он остаётся внутри **рекурсивной алгебры типов**, не требуя перехода к $1/(1-X)$.

Если перейти к формальным рядам, можно ещё сказать: разность $D = L' - L^2$ удовлетворяет

$$D = XD,$$

а единственный формальный ряд с этим свойством — $D = 0$.

Но первый вариант концептуально чище.

Итак, задача 3 решена двумя путями:

$$\text{структурно: } L' = L \times L,$$

$$\text{алгебраически: } L' = L + XL' \text{ \& } L^2 = L + XL^2.$$

Можно идти к задаче 4.

10/2/2026, 10:45:13 AM · gpt-5-6-thinking





Задача 4. Если мы говорим о EGF, то коэффициент $1 = n!/n!$. А для n элементов у нас как раз $n!$ вариантов. А вот с OGF я не понимаю: 1 потому, что у нас только одна форма списка, если мы забудем про элементы? Вообще говоря, что представляет собой OGF? Различные формы? А EGF - для меток? Я просто не понимаю, что и когда использовать и почему.

10/2/2026, 10:51:26 AM



Да — именно здесь стоит остановиться и развести понятия. Ты правильно чувствуешь: **OGF не означает автоматически “формы”, а EGF не означает автоматически “метки”**. Формально обе просто способы упаковать последовательность чисел. Но в комбинаторике у них есть разные естественные области применения.

Если a_n — число объектов размера n , то:

$$\text{OGF: } A(x) = \sum_{n \geq 0} a_n x^n,$$

$$\text{EGF: } A(x) = \sum_{n \geq 0} a_n \frac{x^n}{n!}.$$

Разница — только в $n!$. Но именно это $n!$ делает EGF естественной для **помеченных объектов**.

Для списка длины n :

- если мы забыли метки и считаем только форму, форма ровно одна:

$$[], [\bullet], [\bullet, \bullet], \dots$$

Поэтому OGF:

$$1 + x + x^2 + \dots = \frac{1}{1-x}.$$

- если у нас n различных меток, их можно расположить в списке $n!$ способами. Тогда EGF:

$$\sum_{n \geq 0} n! \frac{x^n}{n!} = \sum_{n \geq 0} x^n.$$

Получилась та же функция, но **по совершенно разным причинам**.

Это совпадение здесь случайно приятное.

Главный критерий выбора такой.

Для **непомеченных** объектов естественно OGF, потому что при произведении двух классов размеры просто складываются:

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

Именно это даёт обычное произведение рядов.

Для **помеченных** объектов надо ещё выбрать, какие k из n меток пойдут в первую компоненту:

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

И именно EGF поглощает этот биномиальный коэффициент:

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

Вот почему species естественно живут с EGF.

Очень коротко:



OGF	: размеры складываются,
EGF	: размеры складываются + метки надо распределить.

Хороший контраст дают разбиения множества.

Если элементы **помечены**, число разбиений — Bell numbers B_n , и естественная EGF:

$$\exp(e^x - 1).$$

Если элементы неразличимы и нас интересует только размер блоков, мы фактически считаем **целочисленные разбиения** $p(n)$, и естественная OGF:

$$\prod_{k \geq 1} \frac{1}{1 - x^k}.$$

Это уже совершенно разные последовательности.

Так что на твой вопрос «OGF — это различные формы?» я бы ответил:

часто да, но не по определению.

OGF просто хранит a_n без нормировки. Однако именно для помеченных форм она обычно является естественным инструментом.

А EGF:

тоже просто хранит a_n , но делит коэффициент на $n!$; благодаря этому она особенно естественна для помеченных структур.

И для нашего курса полезно держать одно практическое правило:

если метки существенны и при сборке их надо распределять между компонентами — думай EGF.
--

если считаем формы по размеру без распределения меток — сначала думай OGF.
--

Для списка обе дороги дают одну и ту же $1/(1-x)$, поэтому он слегка маскирует разницу. У помеченных деревьев эта разница уже станет существенной.

10/2/2026, 10:51:41 AM · gpt-5-6-thinking



Ага. Вот это, пожалуй, самое важное для меня на текущий момент (понимание OGF vs EGF). Замечательно! Следующий листок?

10/2/2026, 10:59:49 AM



Да. Тогда **Листок 2** я бы специально построил вокруг того различия, которое сейчас стало центральным:

Листок 2. Деревья Каталана: OGF, EGF и производная

Пусть B - полные плоские бинарные деревья, где атомами считаются листья.

Задача 1

Запиши рекурсивное уравнение для B .

Подсказка только семантическая: дерево - либо один лист, либо корень с двумя **упорядоченными** поддеревьями.

После этого скажи: какую производящую функцию здесь естественнее использовать, если мы считаем **формы деревьев**, забывая метки - OGF или EGF? Почему?

Задача 2

Пусть

$$B(x) = \sum_{n \geq 1} b_n x^n,$$

где b_n - число форм с n листьями.

Переведи уравнение типов в уравнение для $B(x)$, а затем выпиши первые несколько коэффициентов вручную:

$$b_1, b_2, b_3, b_4, b_5.$$

Не используй пока явную формулу Каталана.

Задача 3

Теперь представим, что n листьев имеют различные метки.

Если форм дерева b_n , сколько будет **помеченных** деревьев?

И что произойдёт, если эти числа упаковать в EGF?

Это задача специально на сравнение OGF и EGF: здесь должно произойти интересное совпадение.

Задача 4

Удаляем один лист, оставляя дырку.

Попробуй сначала **комбинаторно**, без дифференцирования, описать B' :

- если дырка - всё дерево;
- если нет, что надо запоминать на каждом шаге от дырки к корню?

После этого попробуй записать уравнение для B' .

Задача 5

Только после задачи 4 продифференцируй

$$B = X + B^2$$

формально и сравни с тем, что получилось комбинаторно.



Начнём опять **только с задачи 1.**

10/2/2026, 11:00:02 AM · gpt-5-6-thinking



Задача 1. $B = 1 + X \times B^2$ [вроде так, но ... почему 1 перед + и X после? Если мы стираем метки? С B^2 всё понятно, и хочется написать просто $B = 1 + B^2$.] Естественно использовать OGF, поскольку [цитирую] "при произведении двух классов размеры просто складываются".

10/2/2026, 11:06:57 AM



Да — тут ты поймал важную развилку. Я в условии зафиксировал, что **атомами считаются листья**, поэтому правильное уравнение именно

$$B = X + B^2.$$

Не $1 + XB^2$.

Почему? Потому что есть два случая:

- дерево — один лист: это один атом X ;
- или внутренний узел с двумя упорядоченными поддеревьями: B^2 .

Внутренний узел мы в размере не считаем.

Если же выбрать другой размер — **число внутренних вершин**, — тогда лист имеет размер 0, то есть даёт 1, а внутренняя вершина даёт X . Тогда получится уже

$$C = 1 + XC^2.$$

Обе модели правильны, просто считают размер по-разному.

И вот здесь особенно важно: **“забыть метки” не значит заменить X на 1**. X отмечает не метку, а один атом размера. Метки и размер — разные вещи.

Поэтому для нашей модели по листьям:

$$B = X + B^2 \quad \longrightarrow \quad B(x) = x + B(x)^2.$$

А запись

$$B = 1 + B^2$$

возникает уже в другом контексте — когда мы вообще забываем градуировку по размеру и рассматриваем только рекурсивный тип «лист или пара деревьев». Именно это было в *Seven Trees in One*. Для перечисления по размеру такая запись недостаточна: мы потеряли информацию, сколько атомов в объекте.

И да, твой выбор **OGF** совершенно правильный:

$$\text{формы без меток} \Rightarrow \text{естественно OGF}$$

потому что при произведении классов размеры просто складываются.

Так что задача 1 в окончательной форме:

$$B = X + B^2, \quad B(x) = x + B(x)^2.$$

Теперь задача 2: из этого уравнения получить первые b_n , не используя формулу Каталана.

10/2/2026, 11:07:15 AM · gpt-5-6-thinking





Секундочку: то есть, размер - это количество атомов? [В первом варианте листьев, во-втором: "не листьев"]?

10/2/2026, 11:14:19 AM



Да, именно. В нашей алгебре X означает **один атом размера 1**, а 1 — структуру размера 0. Поэтому «размер» — это количество тех объектов, которые мы решили считать атомами.

Для полного бинарного дерева можно выбрать разные градуировки.

Если атомы — **листья**, то

$$B = X + B^2.$$

Лист имеет размер 1, внутренний узел сам размера не добавляет. Дерево с n листьями имеет размер n .

Если атомы — **внутренние вершины**, то лист считается структурой размера 0, поэтому

$$C = 1 + XC^2.$$

Здесь X — именно внутренняя вершина. Дерево с n внутренними вершинами имеет размер n .

Можно выбрать даже третью градуировку — считать **все вершины**. Тогда и лист, и внутренняя вершина дают по одному атому:

$$D = X + XD^2.$$

Так что принцип такой:

X = то, одну штуку чего мы считаем единицей размера.

Это не обязательно «метка» и не обязательно «вершина». Сначала мы выбираем статистику размера, а потом строим производящую функцию относительно неё.

Для полных бинарных деревьев первые два варианта почти эквивалентны, потому что

$$\# \text{листьев} = \# \text{внутренних вершин} + 1.$$

Отсюда и два обычных каталановских индексирования:

$$C_{n-1} \quad \text{для } n \text{ листьев,}$$

или

$$C_n \quad \text{для } n \text{ внутренних вершин.}$$

И да: во втором твоё «не листья» лучше точно называть **внутренними вершинами**.

10/2/2026, 11:14:31 AM · gpt-5-6-thinking





А! О сколько нам открытий чудных готовит сын ошибок трудных! Вот поэтому активное владение так и важно: это совершенно ключевой момент (уже второй), который я не понимал! Задача 2: ну будем считать для листьев, то есть $B = x + B^2$. Заменяем, B справа на выражение для B : $B = x + (x + B^2)^2 = x + x^2 + 2xB^2 + B^4$. Ещё раз заменяем: $B = x + x^2 + 2x^3 + \dots$ и так далее.

10/2/2026, 11:20:28 AM



Да! И твой способ совершенно законный. Более того, это хороший **конструктивный способ решать уравнение формальных рядов**:

$$B = x + B^2.$$

Начинаем с грубого приближения $B = x + \dots$ и подставляем его обратно. Каждая новая подстановка определяет ещё несколько коэффициентов.

Ты получил:

$$B = x + (x + B^2)^2 = x + x^2 + 2xB^2 + B^4.$$

Так как $B = x + O(x^2)$,

$$B^2 = x^2 + O(x^3),$$

поэтому

$$2xB^2 = 2x^3 + O(x^4),$$

и

$$B = x + x^2 + 2x^3 + \dots$$

Чтобы идти дальше, удобнее теперь перейти от последовательных подстановок к сравнению коэффициентов. Пусть

$$B(x) = b_1x + b_2x^2 + b_3x^3 + \dots$$

Из

$$B = x + B^2$$

получаем

$$b_1 = 1,$$

а при $n \geq 2$

$$b_n = \sum_{k=1}^{n-1} b_k b_{n-k}.$$

Теперь:

$$b_2 = b_1 b_1 = 1,$$

$$b_3 = b_1 b_2 + b_2 b_1 = 2,$$

$$b_4 = 1 \cdot 2 + 1 \cdot 1 + 2 \cdot 1 = 5,$$

$$b_5 = 1 \cdot 5 + 1 \cdot 2 + 2 \cdot 1 + 5 \cdot 1 = 14.$$

Итак,

$$B(x) = x + x^2 + 2x^3 + 5x^4 + 14x^5 + \dots$$

- Каталан со сдвигом:



$$b_n = C_{n-1}.$$

Но мне особенно нравится то, что ты начал именно с **итерации самого структурного уравнения**. Это показывает, что производящая функция здесь не внешняя формула, а буквально развивается из рекурсивного определения дерева.

Теперь задача 3 как раз проверит сегодняшнее главное открытие: что произойдёт, если те же формы деревьев снабдить различными метками на листьях, и перейти от OGF к EGF.

10/2/2026, 11:20:40 AM · gpt-5-6-thinking



Задача 3. Не совсем понял, но попробую. Если форм b_n (для n листьев/атомов), то различных деревьев $(n+1)! \times b_n$ ($n+1$ вершин). EGF даёт: $(n+1)!/n! \times b_n$ [??] Что-то не так, мне кажется.

10/2/2026, 11:36:02 AM



Да — ты правильно почувствовал, что что-то съехало. Ошибка именно в том ключевом месте, которое мы только что обнаружили: **метки ставятся на атомы**, а атомами мы выбрали листья.

То есть при n листьях:

- форм дерева: b_n ;
- на n листьях можно расставить n различных меток $n!$ способами.

Поэтому помеченных деревьев:

$$n! b_n.$$

Не $(n+1)!b_n$: внутренние вершины мы здесь **не метим и не считаем атомами**.

Кстати, у полного бинарного дерева с n листьями вообще не $n+1$, а

$$n-1$$

внутренних вершин, то есть всего

$$2n-1$$

вершин.

Теперь EGF помеченных деревьев:

$$\sum_{n \geq 1} n! b_n \frac{x^n}{n!} = \sum_{n \geq 1} b_n x^n.$$

То есть получается ровно та же функция:

$$B_{\text{EGF}}(x) = B_{\text{OGF}}(x).$$

Но смысл коэффициентов разный:

$$\underbrace{b_n}_{\text{OGF}}$$

считает непомеченные формы,

а

$$\underbrace{n! b_n}_{\text{EGF}}$$

считает те же формы с различными метками на листьях.

Это очень удачный пример того, почему OGF и EGF иногда **численно совпадают как формальные ряды**, хотя комбинаторная интерпретация у них совершенно разная.

И это как раз проверка ключевого принципа:

$$\boxed{\text{размер} = \text{число атомов}, \quad \text{метки ставятся именно на атомы.}}$$

Теперь можно переходить к задаче 4 — производной дерева Каталана.

10/2/2026, 11:36:11 AM · gpt-5-6-thinking





"Метки ставятся на атоме" - теперь понятно. Да, тот же ряд. Задача 4. Если у нас только один лист (X), то остается пустое место (1), Если дырка внутри [дыркой может быть только атом, то есть лист, правильно!], то, чтобы полностью восстановить дерево, начиная от этой дырки [это наша цель?], надо запомнить путь вверх (куда идти: влево/вправо - это true/false, то есть 2), дерево "по другую сторону" для каждого предка. То есть, $1 + 2 \times \text{Seq}(B)$ - примерно так. Из формулы: $x + B^2$, производная: $B' = 1 + 2 \times B \times B'$ или $B' = 1/(1-2B) = 1 + 2B/(1-x)$. В принципе, похоже.

10/2/2026, 11:56:00 AM



Да — ты почти всё схватил, и здесь есть две небольшие, но важные поправки.

Во-первых, **да: дыркой может быть только лист**, потому что именно листья у нас являются атомами X . Производная берётся по X . Если бы мы отдельно считали внутренние вершины другим атомом, можно было бы дифференцировать и по нему.

Во-вторых, твоя структурная идея абсолютно правильная, но формула должна быть не

$$1 + 2 \text{Seq}(B),$$

а

$$B' = \text{Seq}(2B).$$

Почему? Каждый **один шаг вверх** от дырки требует запомнить пару данных:

(лево/право, соседнее дерево),

то есть тип одного слоя контекста:

$$2B.$$

А таких слоёв может быть $0, 1, 2, \dots$. Поэтому

$$B' = 1 + (2B) + (2B)^2 + (2B)^3 + \dots = \text{Seq}(2B).$$

Пустая последовательность соответствует ровно случаю, когда исходное дерево состояло из единственного листа:

$$X \longrightarrow \square.$$

И теперь формальное дифференцирование идеально совпадает:

$$B = X + B^2,$$

$$B' = 1 + 2BB'.$$

То есть либо путь вверх пуст, либо имеется первый слой $2B$, за которым снова идёт контекст B' :

$$B' = 1 + (2B)B'.$$

Отсюда

$$B' = \frac{1}{1 - 2B},$$

то есть именно

$$1 + 2B + (2B)^2 + \dots.$$

У тебя в последней строке ещё случайно появилось $1/(1-x)$; там должно быть $1/(1-2B)$.

И мне кажется, здесь мы получили хороший результат активного прохода: ты сам восстановил главное содержание **tree zipper**:

контекст дерева = последовательность (направление + соседнее поддерево).

Фактически задачи 4 и 5 ты решил одновременно.

10/2/2026, 11:56:19 AM · gpt-5-6-thinking



Ага, не совсем решил, но идеи правильные. Что-то понял. Вопрос о zippers. Почему мы собираем контекст? Цель? Восстановить всю структуру с этой точки/дырки?

10/2/2026, 12:04:38 PM



Да. Самая короткая формулировка такая:

`zipper` = выделенная часть структуры + всё, что нужно, чтобы восстановить целое.

То есть если у нас есть дерево и мы сфокусировались на некотором поддереве, мы как бы «вырезаем» его:

$$T \longrightarrow (\text{focus}, \text{context}).$$

`focus` — текущее поддерево.

`context` — оставшая информация, которой не хватает, чтобы вставить `focus` обратно и получить исходное дерево.

Для бинарного дерева контекст — это как раз путь к корню, и на каждом шаге надо помнить:

- были мы левым или правым ребёнком;
- какое поддерево находилось с другой стороны.

Например, если было

$$((a\ b)\ (c\ d))$$

и `focus` — поддерево $(c\ d)$, то контекст грубо можно мыслить как

$$((a\ b)\ \square).$$

Если `focus` заменить на другое дерево S , контекст немедленно даёт

$$((a\ b)\ S).$$

Поэтому да: главная цель — **восстановление структуры из локальной позиции**.

Но алгоритмически ещё важнее, что это даёт локальную навигацию.

Можно:

вниз к ребёнку,
вверх к родителю,
влево/вправо,
заменить текущее поддерево,

не теряя остального дерева.

Именно поэтому `zipper` похож на курсор в структуре данных.

Есть ещё очень красивая абстрактная формулировка:

если F — тип структуры, то

$$F'$$

— тип **контекста с одной дыркой**.

А операция подстановки

$$F' \times X \rightarrow F$$

говорит: «вставь атом/подструктуру в дырку и восстанови целое».

Для деревьев обычно дырка может содержать не просто атом, а целое поддерево, поэтому practically мы думаем:

context + subtree → whole tree.

Так что твоя формулировка совершенно верная:

мы собираем контекст именно затем, чтобы из выбранной точки можно было восстановить всю структуру — и, более того, редактировать её локально и потом собрать обратно.

10/2/2026, 12:04:42 PM · gpt-5-6-thinking



Да, но в нашем случае, наши атомы только листья, поэтому мы поддеревья не редактируем. Фактически, мы можем только заменить лист. Для редактирования вершин нам надо $B = X + XB^2$. Я правильно понимаю?

10/2/2026, 12:07:55 PM



Да — и здесь ты поймал ещё одну важную тонкость. Я раньше немного смешал **две разные производные/два разных вида дырки**.

Если у нас

$$B = X + B^2,$$

то X — только листья. Поэтому производная по X ,

$$\frac{\partial B}{\partial X},$$

действительно выделяет/удаляет **лист**, а не произвольную вершину. В этом смысле ты совершенно прав: такой B' сам по себе даёт контекст для замены листа.

Если мы хотим считать **каждую вершину атомом**, то естественно написать

$$B = X + XB^2.$$

Здесь:

- лист = одна вершина X ;
- внутренняя вершина = X плюс два поддерева.

Тогда производная по X выделяет произвольную вершину дерева.

Но есть ещё третий, и для zipper'ов самый важный, вариант.

Для zipper'a обычно дырка означает не «удалённый атом», а **удалённое целое поддерево**. Тогда надо смотреть на конструктор дерева как на функцию от рекурсивного аргумента:

$$F(Y) = X + Y^2.$$

И дифференцировать именно по Y :

$$F'(Y) = 2Y.$$

Это говорит: если в одном узле вынуть один из двух дочерних subtree-slots, надо помнить:

- слева или справа была дырка: 2;
- соседнее поддерево: Y .

После подстановки $Y = B$:

$$2B.$$

А полный путь от вырезанного поддерева до корня — последовательность таких слоёв:

$$\text{Seq}(2B).$$

Вот это и есть стандартный zipper-контекст для **произвольного поддерева**.

Так что лучше различать:

$\partial/\partial X$	— выделяем атом, который решили считать X ;
$\partial/\partial Y$	— выделяем рекурсивный slot, т.е. целое поддерево.

И да: твоя формула

$$B = X + XB^2$$

правильна, если цель — сделать **все вершины атомами и уметь выделять любую вершину**. Но для обычного tree zipper, где мы хотим вырезать и заменить целое поддерево, ещё точнее использовать производную по рекурсивному аргументу конструктора.

10/2/2026, 12:08:11 PM · gpt-5-6-thinking



Я примерно понимаю, что ты хочешь сказать, но для активного понимания это слишком сложно. Мы можем попробовать сделать это целью одного из листов (не обязательно следующего). Кстати, я бы хотел заметить, что у меня проблемы даже с предыдущим листком (а именно, я пошёл в правильном направлении, но не дал правильный ответ). То есть, к сожалению, "разжевывать" для меня надо ещё подробнее (больше задач, но они проще).

10/2/2026, 12:14:48 PM



Да, это разумная коррекция курса. И я согласен с твоей самооценкой: во втором листке направление мысли было хорошее, но несколько раз точная формула не получалась с первого раза. Значит, размер шага надо уменьшить.

Дальше я буду строить листки так: одна задача = одна маленькая идея. Не «получи $B' = \text{Seq}(2B)$ », а, например: сначала определить, что именно считается атомом; потом что означает дырка; потом описать один шаг вверх; потом тип одного слоя; потом два слоя; потом произвольное число слоёв; и лишь после этого записать Seq . То есть формула должна появляться в самом конце как название уже понятой конструкции.

И ещё важное правило: если ты сказал что-то вроде «примерно $1 + 2\text{Seq}(B)$ », я не буду считать тему усвоенной только потому, что идея близка. Мы разберём, **почему именно** это не та формула, и через одну-две маленькие задачи заставим правильную конструкцию возникнуть самостоятельно.

Тему

атомная дырка vs дырка-поддерево

и производную по рекурсивному аргументу действительно отложим. Сделаем потом отдельный листок, где это будет **целью**, а не побочным усложнением.

Я бы сейчас не шёл дальше к новым структурам. Лучше сделать короткий **Листок 2½: закрепление деревьев Каталана**, примерно на 6-8 совсем маленьких шагов, чтобы довести до активного владения три вещи:

X = единица выбранного размера,

OGF vs EGF,

B' = контекст листа.

И начнём с совсем простого вопроса.

Задача 1. Для полного бинарного дерева мы решили считать размером число листьев. Какой размер имеют:

- один лист;
- дерево из корня с двумя листьями;
- дерево с тремя листьями?

И что в записи $B = X + B^2$ означает X ?

10/2/2026, 12:15:01 PM · gpt-5-6-thinking





1,2,3; X означает один лист (простейшее дерево, состоящее из одного листа) - база рекурсии.

10/2/2026, 12:17:55 PM



Да, точно.

1, 2, 3

и

X = один лист, то есть один атом размера 1.

Причём ты хорошо добавил: это ещё и **база рекурсивного определения**.

Следующая микрозадача.

Задача 2. В том же соглашении что означает член

B^2

в уравнении

$B = X + B^2?$

Ответь словами: из каких двух объектов строится такой случай и почему именно произведение.

10/2/2026, 12:18:00 PM · gpt-5-6-thinking





Два независимых друг от друга поддеревя, причём корень мы не считаем.

10/2/2026, 12:18:46 PM



Да. Точнее:

$$B^2 = B \times B$$

— это **упорядоченная пара** независимых поддеревьев: левое и правое.

Корень здесь действительно не считается атомом, поэтому отдельного множителя X нет.

Задача 3. Если левое поддерево имеет k листьев, а правое — m листьев, какой размер имеет всё дерево? И как это отражается в произведении OGF?

10/2/2026, 12:18:52 PM · gpt-5-6-thinking



$k+m$ листьев. В OGF: член x^{k+m} отвечает за деревья с $k+m$ листьями и получить его можно так: слева 0, справа $k+m$ + слева 1, справа $k+m-1$, ..., слева k , справа m (наш случай) и так далее. Для всех таких подслучаев коэффициенты суммируются.

10/2/2026, 12:22:42 PM



Да, идея правильная: размеры складываются, а коэффициенты при одинаковой общей степени суммируются.

Только одна важная поправка: в нашем B **нет дерева размера 0**, потому что минимальное дерево — один лист.

Поэтому для общего размера n разбиения идут не от $0 + n$, а от

$$1 + (n - 1), 2 + (n - 2), \dots, (n - 1) + 1.$$

Если

$$B(x) = \sum_{n \geq 1} b_n x^n,$$

то коэффициент при x^n в $B(x)^2$ равен

$$\sum_{k=1}^{n-1} b_k b_{n-k}.$$

И это буквально: выбираем форму левого поддерева размера k и форму правого размера $n - k$.

Задача 4. Почему в этой формуле стоит именно произведение $b_k b_{n-k}$, а затем сумма по k ?

10/2/2026, 12:22:48 PM · gpt-5-6-thinking



Ага. То есть, если есть правое поддереву, то есть и левое, и наоборот? Например, линейная конструкция невозможна? Я как-то не уверен.. Сумма произведений: у нас есть b_k деревьев с k листьями, $b_{(n-k)}$ деревьев с $n-k$ листьями. Они независимы: то есть, можем взять любое поддереву слева и любое справа. Поэтому произведение. Для любого k , $n-k$ получаем n листьев, то есть всё складывается.

10/2/2026, 12:27:25 PM



Да, всё верно.

И на твой вопрос: **в нашей модели линейная конструкция невозможна**. Мы рассматриваем именно **полные бинарные деревья**: каждая внутренняя вершина имеет ровно двух детей. Поэтому если есть внутренняя вершина, у неё обязательно есть и левое, и правое поддереву.

Отсюда:

- слева можно выбрать любое из b_k деревьев;
- справа — любое из b_{n-k} ;
- выборы независимы, поэтому

$$b_k b_{n-k}.$$

- затем суммируем по всем допустимым k :

$$b_n = \sum_{k=1}^{n-1} b_k b_{n-k}.$$

Это именно правило «произведение для независимых выборов, сумма для альтернатив».

Следующая микрозадача.

Задача 5. Почему для этих **форм** деревьев естественно использовать OGF, а не EGF? Ответь без формул, только словами.

10/2/2026, 12:27:29 PM · gpt-5-6-thinking





"потому что при произведении двух классов размеры просто складываются"

10/2/2026, 12:28:11 PM



Да. И добавим вторую половину мысли:

мы считаем только **формы**, без распределения различных меток между частями.

Поэтому обычное произведение OGF уже делает ровно то, что нужно: размеры складываются.

То есть:

OGF естественна здесь, потому что считаем формы и размер просто складывается.

Следующая микрозадача.

Задача 6. Теперь те же деревья, но каждый из n листьев получает уникальную метку.

Если форм дерева с n листьями b_n , сколько будет помеченных деревьев?

10/2/2026, 12:28:16 PM · gpt-5-6-thinking





$n! \times b_n$ - произвольная нумерация листьев

10/2/2026, 12:30:52 PM



Да, точно:

$$n! b_n$$

потому что на каждой из b_n форм можно расставить n различных меток по n листьям $n!$ способами.

Следующая микрозадача.

Задача 7. Если эти числа $n!b_n$ упаковать в EGF,

$$\sum_{n \geq 1} n! b_n \frac{x^n}{n!},$$

во что это превращается? И что в этом немного забавно?

10/2/2026, 12:30:56 PM · gpt-5-6-thinking



$n!$ сокращаются и получаем тот же ряд, что и для OGF.

10/2/2026, 12:31:42 PM



Да, именно:

$$\sum_{n \geq 1} n! b_n \frac{x^n}{n!} = \sum_{n \geq 1} b_n x^n.$$

То есть **EGF помеченных деревьев совпала как формальный ряд с OGF непомеченных форм.**

Забавно именно то, что смысл у ряда разный:

- в OGF коэффициент b_n = число форм;
- в EGF тот же коэффициент возникает после нормировки числа помеченных объектов $n!b_n$.

Следующая микрозадача.

Задача 8. В нашей модели атомы — листья. Если мы берём производную B' , что именно означает дырка: можно ли удалить внутреннюю вершину, или только лист? Почему?

10/2/2026, 12:31:46 PM · gpt-5-6-thinking





Только лист. Удалять можно только атом.

10/2/2026, 12:32:23 PM



Да. Именно:

B' = дерево с одной дыркой вместо листа

потому что производная берётся по X , а X у нас обозначает один лист-атом.

Следующая микрозадача.

Задача 9. Предположим, дырка не является всем деревом целиком. Значит, она находится где-то ниже корня.

Чтобы сделать **один шаг вверх** от дырки к её родителю, какие две вещи надо запомнить?

10/2/2026, 12:32:26 PM · gpt-5-6-thinking





Звучит двусмысленно. Чтобы сделать шаг вверх, ничего запоминать не надо. Нужно запомнить, чтобы иметь возможность восстановить что-то по запомненным данным. Переформулируй.

10/2/2026, 12:35:03 PM



Да, согласен: формулировка была плохая.

Переформулирую точнее.

Задача 9. Пусть у нас есть дерево с дыркой вместо одного листа. Мы хотим хранить достаточно информации, чтобы потом восстановить исходное дерево.

Рассмотрим **одного родителя на пути от дырки к корню**.

Какие данные об этом родителе надо сохранить, чтобы, зная уже восстановленную нижнюю часть, восстановить и самого родителя?

То есть вопрос не «как подняться», а:

что надо запомнить на одном уровне контекста?

10/2/2026, 12:35:08 PM · gpt-5-6-thinking





Вот из-за таких вопросов ученики и путаются. Давай я сформулирую за тебя (ибо твоя формулировка мне всё равно не нравится). У нас уже есть одно из поддеревьев некой вершины (не листа, "родителя"). Что нужно, чтобы восстановить полностью дерево этого родителя? Ответ: куда прицепляем поддерево, которое уже есть: слева или справа? (2 варианта) и второе поддерево полностью (и мы уже знаем куда его цеплять). Родитель не атом, индивидуальности не имеет, так что о нём нам знать ничего не надо.

10/2/2026, 12:50:02 PM



Да. Твоя формулировка значительно лучше и педагогически чище.

И ответ именно такой:

$$\text{один слой контекста} = 2 \times B$$

потому что надо сохранить:

- с какой стороны стоит уже известное поддерево: слева или справа;
- второе поддерево целиком.

А сам родитель дополнительных данных не требует, потому что в нашей модели внутренняя вершина не является атомом и не имеет собственной метки/идентичности.

То есть если обозначить уже восстановленную нижнюю часть через S , один шаг восстановления имеет два вида:

$$(S, B) \quad \text{или} \quad (B, S).$$

Это и есть множитель $2B$.

Следующая микрозадача, теперь уже аккуратно:

Задача 10. Если между дыркой и корнем таких уровней контекста может быть $0, 1, 2, \dots$, какой тип описывает весь контекст целиком?

10/2/2026, 12:50:05 PM · gpt-5-6-thinking





Тип Seq[2B], то есть [] или [2B] или [2B, 2B] или ...

10/2/2026, 12:52:11 PM



Да, точно.

$$B' = \text{SEQ}(2B)$$

то есть

$$B' = 1 + 2B + (2B)^2 + (2B)^3 + \dots$$

Смысл:

- [] — между дыркой и корнем нет ни одного уровня;
- [2B] — один уровень;
- [2B, 2B] — два уровня;
- и т. д.

Причём пустая последовательность соответствует случаю, когда исходное дерево состояло только из одного листа, который и заменили дыркой.

Теперь совсем маленький следующий шаг.

Задача 11. Любая последовательность либо пустая, либо состоит из первого элемента и оставшейся последовательности.

Какое рекурсивное уравнение для B' следует из

$$B' = \text{SEQ}(2B)?$$

10/2/2026, 12:52:18 PM · gpt-5-6-thinking



$$B' = 1 + 2B \times B'$$

10/2/2026, 12:54:04 PM



Да, ровно:

$$B' = 1 + 2B B'$$

Пустой контекст даёт 1, непустой — первый слой $2B$ и оставшийся контекст B' .

И теперь видно, что это в точности совпадает с формальным дифференцированием

$$B = X + B^2$$

по X :

$$B' = 1 + 2B B'.$$

Вот теперь формула уже не “падает с неба”: мы сначала построили её как описание контекста, и только потом увидели, что обычное правило производной даёт то же самое.

Следующая микрозадача:

Задача 12. Реши это уравнение относительно B' , используя геометрический ряд, и свяжи ответ обратно с $\text{SEQ}(2B)$.

10/2/2026, 12:54:07 PM · gpt-5-6-thinking





$B' = 1/(1-2B)$, что, собственно, и есть уравнение для $\text{Seq}(2B)$

10/2/2026, 12:55:28 PM



Да. Именно:

$$B' = \frac{1}{1-2B}$$

и это равно

$$\text{SEQ}(2B) = 1 + 2B + (2B)^2 + \dots$$

То есть мы замкнули круг тремя способами:

$$\text{контекст дерева} \longleftrightarrow \text{SEQ}(2B) \longleftrightarrow B' = 1 + 2BB' \longleftrightarrow \frac{1}{1-2B}.$$

И вот это уже, мне кажется, можно считать активным пониманием именно этого куса: ты сам восстановил все промежуточные шаги, а не просто узнал финальную формулу.

На этом Листок 2½ я бы завершил. Следующий логичный листок — **помеченные корневые деревья и переход к $T = XSET(T)$, EGF и $T = xe^T$** , но тоже в таком же мелком темпе.

10/2/2026, 12:55:35 PM · gpt-5-6-thinking



Да. Темп я бы чуть-чуть ускорил. Что-то среднее между 5 задачами (раньше, чересчур трудно) и 12 (сейчас, чересчур легко). Может, 9? (комбинируя часть микро-вопросов).

10/2/2026, 12:58:02 PM



Да, 9 - похоже на хороший темп. Сделаем так: **около 9 задач на листок**, каждая - не микро-вопрос, но и не скачок через несколько идей. Близкие шаги будем объединять.

И пойдём, как раньше, **по одной задаче за раз**, чтобы можно было подправлять уровень сложности на ходу.

Листок 3. Помеченные корневые деревья

Задача 1.

Пусть T - корневые помеченные деревья.

После удаления корня дерево распадается на несколько корневых поддеревьев.

Попробуй записать тип T через X , T и операцию `SET`.

Нужно объяснить две вещи:

- почему корень даёт именно X ;
- почему дети корня образуют именно `SET(T)`, а не `SEQ(T)`.

10/2/2026, 12:58:10 PM · gpt-5-6-thinking





Встречный вопрос: а что такое корневые помеченные деревья? Что в них особенного? Какие у нас атомы? Что считаем? Что помечаем?

10/2/2026, 1:02:20 PM



Да — это как раз тот вопрос, который надо было поставить **до** формулы. Здесь модель заметно отличается от каталановских деревьев.

Корневое помеченное дерево здесь означает обычное конечное дерево, у которого:

- выбрана одна вершина как **корень**;
- **каждая вершина является атомом**;
- все n вершин имеют различные метки, например $1, \dots, n$;
- порядок детей вершины не задан: это **не плоское** дерево.

То есть размер дерева:

размер = число вершин.

И X означает **одну вершину**.

Например, на метках $\{1, 2, 3, 4\}$ дерево может иметь корень 3, соединённый с 1 и 4, а 1 соединён ещё с 2. Метки — часть структуры: если поменять метки 1 и 4, обычно получится другое помеченное дерево.

Это принципиально отличается от нашего каталановского примера:

	Catalan	сейчас
атомы	листья	все вершины
метки	можно было добавить листьям	все вершины помечены
дети	ровно 2, упорядочены	сколько угодно, не упорядочены
размер	#листьев	#вершин

Теперь исходная задача становится осмысленной.

Если удалить корень, что останется? Несколько компонент; каждая из них сама естественно становится корневым помеченным деревом — корнем будет вершина, которая была соседом удалённого корня.

И вот теперь можно спросить:

почему весь тип должен иметь вид

$$> T = X \cdot \text{SET}(T)? >$$

Попробуй уже с этой точной моделью.

10/2/2026, 1:02:28 PM · gpt-5-6-thinking



Ага. Причём одна вершина/атом ещё и дополнительно выделена как корень. И это X в нашей формуле. Удаляем корень и получаем множество (порядка нет!) корневых деревьев (у которых загадочным образом тоже выделен корень, а раньше - до удаления корня - того не было. Что смущает. С одной стороны корень это просто атом. С другой стороны: выбрали другой корень - получили другое дерево [??]) Итого: $T = X \times \text{SET}(T)$

10/2/2026, 1:07:53 PM



Да — формула правильная:

$$T = X \cdot \text{SET}(T).$$

И ты очень правильно заметил место, которое кажется «магическим»: **откуда у компонент после удаления корня берутся собственные корни?**

Они не выбираются заново. Они определяются автоматически.

Пусть исходный корень — r . Удаляем r . Каждая оставшаяся компонента была соединена с r ровно одним ребром. Значит, в каждой компоненте есть ровно одна вершина, которая раньше была непосредственным соседом r . **Вот она и становится корнем этой компоненты.**

То есть корень каждого поддерева возникает канонически из прежней связи с удалённым корнем.

И да:

выбрали другой корень в том же неукоренённом дереве — обычно получили другое корневое дерево.

Корневое дерево — это не просто граф-дерево, а пара

(дерево, выделенная вершина).

Поэтому на одном и том же неукоренённом дереве с n вершинами обычно есть n разных корневых версий.

В формуле

$$T = X \text{ SET}(T)$$

множитель X — это именно **выделенная вершина-корень**. Остальные вершины распределяются между дочерними корневыми поддеревьями.

Очень полезно представить это на множестве меток U :

1. выбираем одну метку $r \in U$ для корня;
2. оставшиеся метки $U \setminus \{r\}$ разбиваются на блоки;
3. на каждом блоке строится корневое дерево;
4. корень каждого такого дерева соединяется с r .

Поскольку дети корня не упорядочены, получаем именно

$$\text{SET}(T),$$

а не $\text{SEQ}(T)$.

Так что твоё беспокойство было по делу, но здесь никакого дополнительного выбора корней у компонент нет — они наследуются от ребра, которым были прикреплены к старому корню.

Следующая задача тогда уже естественная:

Задача 2. Почему для этих помеченных деревьев естественно использовать EGF, а не OGF? Попробуй объяснить словами, без формул.

10/2/2026, 1:08:07 PM · gpt-5-6-thinking





Ага пара (дерево, вершина) всё объясняет. Задача 2. У нас не просто форма. У нас нумерация. Другая нумерация на той же форме - другое дерево. Способов перенумеровать $n!$. При перемножении, эти способы надо учитывать (выбрать k из n). Что соответствует EGF.

10/2/2026, 1:11:10 PM



Да — **основная причина выбрана правильно**: при произведении помеченных структур надо распределить общий набор меток между компонентами, и биномиальные коэффициенты естественно поглощаются EGF.

Но одну фразу я бы поправил, потому что здесь легко заложить будущую ошибку:

«на одной форме есть $n!$ различных нумераций»

— **не всегда** для неплоских деревьев. У формы могут быть симметрии. Например, корень с двумя листьями: перестановка двух листьев ничего не меняет как граф. В общем случае число различных маркировок данной формы равно

$$\frac{n!}{|\text{Aut}(T)|}.$$

Поэтому EGF здесь лучше мотивировать не через « $n!$ маркировок каждой формы», а именно через твою вторую мысль:

при произведении надо выбрать, какие метки идут в какую компоненту.

Это фундаментальная причина.

Задача 3. Пусть $A(x)$ — EGF некоторого помеченного класса A .

Что должна давать конструкция

$$\text{SET}_k(A),$$

то есть **неупорядоченное множество ровно из k A -компонент**?

Попробуй пройти в два шага:

1. что соответствует **упорядоченной** k -ке A -объектов;
2. что надо сделать, чтобы забыть порядок k компонент?

А затем суммировать по всем $k \geq 0$.

10/2/2026, 1:11:24 PM · gpt-5-6-thinking



Верно. Я забыл про автоморфизмы. Задача 3.1: упорядоченный список - это A^k .
 3.2: чтобы забыть порядок, надо разделить на $k!$ - все перестановки - это одно и то же. Суммирование: $\sum (A^k/k!)$, $k \geq 0$ это $\exp(A)$ [формальный ряд]

10/2/2026, 1:16:10 PM



Да, именно так.

$$\text{SET}_k(A) \longleftrightarrow \frac{A(x)^k}{k!}$$

и затем

$$\text{SET}(A) \longleftrightarrow \sum_{k \geq 0} \frac{A(x)^k}{k!} = e^{A(x)}.$$

Маленькая техническая оговорка: здесь удобно предполагать, что A не имеет структуры размера 0, чтобы $\text{SET}(A)$ не порождал бесконечно много пустых компонент. Для наших деревьев это выполнено.

Теперь **Задача 4.**

У нас уже есть структурное уравнение

$$T = X \cdot \text{SET}(T).$$

Переведи его в уравнение для EGF $T(x)$.

Нужно отдельно сказать, чему соответствует X , и чему — $\text{SET}(T)$.

10/2/2026, 1:16:16 PM · gpt-5-6-thinking





$T = x \cdot \exp(T)$, то есть, X переходит в x , а Set в $\exp$.

10/2/2026, 1:18:04 PM



Да:

$$T(x) = x e^{T(x)}.$$

И объяснение ровно такое:

- $X \mapsto x$, потому что корень — один атом размера 1;
- $\text{SET}(T) \mapsto e^{T(x)}$, потому что дети корня образуют неупорядоченное множество помеченных корневых деревьев.

Это важный момент: мы получили не просто формулу, а прямой перевод структуры в функциональное уравнение.

Задача 5. Теперь не выводим пока формулу Кэли. Просто скажи: если

$$T(x) = \sum_{n \geq 1} t_n \frac{x^n}{n!},$$

то что означает t_n комбинаторно? И почему мы ожидаем, что t_n будет связано с числом всех помеченных деревьев на n вершинах?

10/2/2026, 1:18:08 PM · gpt-5-6-thinking





t_n - это количество пар (дерево, вершина) с учётом маркировок. Хмм, мы так формулу построили - не понимаю вопроса.

10/2/2026, 1:20:08 PM



Да, ты прав: вопрос был плохо поставлен и почти тавтологичен. Твой ответ полный:

$$t_n = \#\{(\text{помеченное дерево на } n \text{ вершинах, выбранный корень})\}.$$

То есть если u_n — число обычных помеченных деревьев, то сразу

$$t_n = n u_n.$$

Я хотел именно это проговорить, но сформулировал так, будто нужно обнаружить что-то ещё.

Задача 6. Теперь извлечём t_n из

$$T = x e^T.$$

Используем формулу Лагранжа в готовом виде:

если

$$Y = x \phi(Y),$$

то

$$[x^n]Y = \frac{1}{n} [u^{n-1}] \phi(u)^n.$$

В нашем случае что такое $\phi(u)$, во что превращается $\phi(u)^n$, и каков коэффициент при u^{n-1} ?

10/2/2026, 1:20:18 PM · gpt-5-6-thinking





$\phi(u)$ это $\exp(u)$, $\phi^n(u)$, это $\exp(nu)$. Коэффициент: $n^{n-1}/(n-1)!$ То есть, коэффициент при x^n - это $n^{n-1}/n!$ То есть, число корневых деревьев: n^{n-1} . Делим n (учитывая выбор корней) и получаем: n^{n-2}

10/2/2026, 1:29:13 PM



Да — и ты за один раз фактически решил **задачи 6 и 7**.

Из

$$T = xe^T$$

по Лагранжу:

$$[x^n]T = \frac{1}{n} [u^{n-1}]e^{nu} = \frac{1}{n} \frac{n^{n-1}}{(n-1)!} = \frac{n^{n-1}}{n!}.$$

Поскольку это EGF,

$$t_n = n^{n-1}.$$

А t_n считает пары

(неукоренённое помеченное дерево, выбранная вершина),

поэтому

$$t_n = n u_n,$$

и

$$u_n = n^{n-2}.$$

То есть формулу Кэли ты получил полностью.

Теперь **Задача 8** — уже про pointing.

Пусть U — тип обычных, неукоренённых помеченных деревьев.

Что означает:

$$U'$$

и что означает

$$XU'?$$

Попробуй объяснить словами, почему должно быть

$$XU' = T.$$

А затем переведи это в уравнение для EGF.

10/2/2026, 1:29:25 PM · gpt-5-6-thinking



U' - у нас есть дырка (может быть в любой вершине). XU' - пара: атом (вершина) и помеченное дерево с дыркой. То есть, мы "заполняем" дырку вершиной и тем самым выделяем её. Что, конечно, соответствует корневому дереву. Получаем, $xU' = T$. Сомнения: мы берём произвольный атом (из нумерованного множества $1..n$) и дерево с нумерованными (помеченными) вершинами: $1..n$ с пропуском одного числа. А как насчёт использования того же номера для выделенной вершины и вершины внутри дерева? Почему это не проблема?

10/2/2026, 1:38:38 PM



Да — это как раз важная тонкость species, и твое сомнение очень уместно.

Ключевой момент: в произведении

$$XU'$$

метки **автоматически делятся на непересекающиеся части**.

То есть на множестве меток $\{1, \dots, n\}$:

- одна метка r идёт в X ;
- остальные $n - 1$ меток идут в U' .

А дырка в U' — это **не одна из этих меток**. Это специальный символ $*$.

Формально:

$$U'[S] = U[S \sqcup \{*\}].$$

То есть U' -структура на множестве S — это обычное дерево на S плюс ещё один специальный атом $*$, который мы интерпретируем как дырку.

Например, пусть глобальные метки:

$$\{1, 2, 3, 4\}.$$

Выбрали 3 для X . Тогда U' живёт на

$$\{1, 2, 4\}$$

плюс специальная дырка $*$.

Например:

дерево на $\{1, 2, 4, *\}$.

Теперь вставляем атом 3 вместо $*$. Получаем дерево на

$$\{1, 2, 3, 4\}$$

и вершина 3 становится выделенной.

То есть никакого дублирования нет:

метка 3 либо в X , либо в U' , но не в обоих.

Именно это гарантирует species-произведение.

Поэтому твоя фраза

«дерево с нумерованными вершинами $1..n$ с пропуском одного числа»

почти правильна, но лучше сказать:

U' имеет $n - 1$ обычных меток плюс одну специальную дырку $*$.

А X поставляет пропущенную метку.

Тогда

$$XU' = T$$

становится буквально биекцией:

$$(\text{метка } r, \text{ дерево с дыркой}) \longleftrightarrow (\text{дерево с выделенной вершиной } r).$$

И на EGF:

$$xU'(x) = T(x).$$

Это хороший момент: **умножение species — не обычное декартово произведение на одном и том же наборе меток; оно всегда подразумевает разбиение набора меток между факторами.**

10/2/2026, 1:38:53 PM · gpt-5-6-thinking





Вот это-то меня и смущает: XU' предполагает независимость X и U' . В то же время у нас X однозначно определяется по U' .

10/2/2026, 1:40:49 PM



Да! Именно слово «**независимы**» здесь и вводит в заблуждение. Для произведения species это не означает, что X и U' выбираются независимо на одном и том же наборе меток.

Правильное определение:

$$(FG)[S] = \bigsqcup_{S=A \sqcup B} F[A] \times G[B].$$

То есть сначала мы **разбиваем общий набор меток** S на две непересекающиеся части A, B , а уже затем строим F -структуру на A и G -структуру на B .

Для

$$XU'$$

$X[A]$ существует только тогда, когда A состоит из одной метки. Поэтому

$$(XU')[S] = \bigsqcup_{r \in S} U'[S \setminus \{r\}].$$

То есть буквально:

1. выбираем метку r , которая пойдёт в X ;
2. на остальных метках строим U' -структуру.

А

$$U'[S \setminus \{r\}] = U[(S \setminus \{r\}) \sqcup \{*\}],$$

то есть дерево, где вместо r стоит специальная дырка $*$.

После этого заменяем

$$* \mapsto r,$$

и получаем дерево с выделенной вершиной r .

Теперь твоя мысль:

если я уже вижу U' на метках $1, \dots, n$ с одной пропущенной меткой, то X определяется однозначно.

Совершенно верно.

Но это не проблема. Это просто означает, что после того как зафиксирован **общий набор меток** S и U' -компонента вместе с её набором меток, дополнение действительно однозначно:

$$A = S \setminus B.$$

То есть компоненты не «независимы» в смысле выбора меток. Независимость была только в другом смысле: **после разбиения меток структуры на двух частях строятся отдельно.**

Я бы поэтому вообще перестал употреблять здесь слово «независимо».

Лучше:

произведение species = разделить метки между двумя компонентами и построить обе структуры.

И для нашего случая:

XU' = выбрать одну метку + построить дерево с дыркой на остальных.

После заполнения дырки выбранной меткой получаем ровно rooted tree.

Вот это, по-моему, правильная картинка.

10/2/2026, 1:41:03 PM · gpt-5-6-thinking



А на комбинаторику это не влияет? В смысле $|X \times Y|$ должно быть равно $|X| \times |Y|$

10/2/2026, 1:43:17 PM



Да — и здесь как раз надо очень строго развести **обычное декартово произведение множеств** и **произведение species**. Я раньше слишком свободно писал $X \times Y$, откуда и возникает вполне законное недоумение.

Для обычных конечных множеств:

$$|A \times B| = |A| |B|.$$

Безусловно.

Но для species F и G произведение FG на одном фиксированном наборе меток S определяется так:

$$(FG)[S] = \bigsqcup_{S=A \sqcup B} F[A] \times G[B].$$

То есть **для каждого фиксированного разбиения** $S = A \sqcup B$ действительно:

$$|F[A] \times G[B]| = |F[A]| \cdot |G[B]|.$$

А потом надо сложить результаты по всем возможным разбиениям меток.

Если $|S| = n$, a_k — число F -структур на k метках, b_{n-k} — число G -структур, то

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

Вот почему для EGF произведение species соответствует обычному произведению функций.

То есть схема такая:

$$\underbrace{\text{фиксированное разбиение меток}}_{A,B} \Rightarrow |F[A]| \cdot |G[B]|$$

а затем

$$\text{суммируем по всем разбиениям } S = A \sqcup B.$$

Для нашего конкретного

$$XU'$$

всё особенно просто. X может жить только на одноэлементном множестве. Поэтому на n метках:

- выбираем, какая из n меток достанется X : n способов;
- на остальных $n - 1$ метках строим U' .

Но U' на $n - 1$ обычных метках — это U -структура на этих $n - 1$ метках плюс специальная дырка, то есть фактически дерево на n вершинах с одной вершиной заменённой дыркой. Таких столько же, сколько обычных U -деревьев на n метках:

$$u_n.$$

Поэтому

$$|(XU')[n]| = n u_n,$$

что как раз равно числу корневых деревьев:

$$t_n = n u_n.$$

Так что **комбинаторика полностью согласована**. Просто правильнее воспринимать species-product не как обычное $A \times B$, а как:

разделить метки + взять декартово произведение структур на частях.

Я бы даже дальше писал $F \cdot G$, а не $F \times G$, чтобы не провоцировать именно эту путаницу.

10/2/2026, 1:43:28 PM · gpt-5-6-thinking



"Вот почему для EGF произведение species соответствует обычному произведению функций." А, ключевой момент номер 3!

10/2/2026, 1:45:51 PM



Да - именно. Это действительно **ключевой момент №3**:

species-product $\longleftrightarrow$ ordinary product of EGF

не потому, что структуры «просто независимы», а потому что species-product включает **суммирование по всем разбиениям набора меток**, а биномиальные коэффициенты от этих разбиений как раз поглощаются факториалами в EGF.

То есть:

$$(FG)[n] \Rightarrow \sum_k \binom{n}{k} f_k g_{n-k},$$

а

$$F(x)G(x)$$

имеет ровно такие коэффициенты в EGF-нормировке.

Мне кажется, теперь уже вырисовываются три опорных вещи, которые раньше были скрыты:

1. X = атом выбранного размера;

2. OGF vs EGF определяется тем, как комбинируются структуры и метки;

3. произведение species = разбиение меток + декартово произведение на частях.

Это уже очень хороший фундамент. Следующий шаг будет намного устойчивее.

10/2/2026, 1:45:54 PM · gpt-5-6-thinking





Хорошо. Продолжаем. Уравнение есть.

10/2/2026, 1:46:51 PM



Да. У нас уже есть два ключевых уравнения:

$$T = xe^T$$

и

$$xU' = T.$$

Значит

$$U' = \frac{T}{x}.$$

А из первого уравнения

$$\frac{T}{x} = e^T.$$

Итак,

$$U' = e^T.$$

Задача 9

Нужно теперь получить U через T , не интегрируя e^T напрямую.

Возьми

$$T = xe^T$$

и продифференцируй по x . Постарайся выразить e^T через T и T' , а затем узнать в полученном выражении производную какой-нибудь простой функции от T .

Цель — самостоятельно получить

$$U = \text{что-то простое из } T.$$

10/2/2026, 1:46:57 PM · gpt-5-6-thinking





$T' = \exp(T) + x \cdot \exp(T) \cdot T'$, то есть, $\exp(T) = T'/(1+xT')$. Что-то не вижу ничего знакомого..

10/2/2026, 3:29:34 PM



Да, твоя алгебра верна, но ты пошёл в менее удобную сторону.

Вместо того чтобы решать относительно e^T , используй исходное уравнение

$$xe^T = T.$$

Тогда из

$$T' = e^T + xe^T T'$$

сразу получается

$$\boxed{T' = e^T + TT'}.$$

Теперь перенеси TT' влево:

$$e^T = T' - TT'.$$

И вот здесь следующий маленький вопрос: **производная какой простой функции от T равна $T' - TT'$?**

10/2/2026, 3:29:44 PM · gpt-5-6-thinking



T-T^2/2

10/2/2026, 3:31:20 PM



Да:

$$\frac{d}{dx} \left(T - \frac{1}{2} T^2 \right) = T' - T T' = e^T.$$

А у нас уже было

$$U' = e^T.$$

Значит

$$U' = \frac{d}{dx} \left(T - \frac{1}{2} T^2 \right).$$

Следовательно,

$$U = T - \frac{1}{2} T^2 + C.$$

Теперь последняя микрозадача: чему равна константа C ? Подсказка: посмотри на $x = 0$, то есть на пустой набор меток.

10/2/2026, 3:31:22 PM · gpt-5-6-thinking





Если меток нет, то и деревьев нет, то есть ноль и тут и там и константа равна 0

10/2/2026, 3:32:34 PM



Да. Поэтому

$$U = T - \frac{1}{2}T^2.$$

И теперь это уже получено тобой по цепочке:

$$xU' = T,$$

$$T = xe^T,$$

$$U' = e^T,$$

$$T' = e^T + TT',$$

$$e^T = T' - TT' = \frac{d}{dx} \left(T - \frac{1}{2}T^2 \right),$$

откуда

$$U = T - \frac{1}{2}T^2.$$

На этом Листок 3 можно считать завершённым. Следующий я бы посвятил **set partitions**: там будет SET внутри SET, Bell/Stirling, и снова производная даст содержательную рекурсию.

10/2/2026, 3:32:37 PM · gpt-5-6-thinking



Хорошо. $U = T - T^2/2$. Это что-то значит геометрически/топологически?

10/2/2026, 3:35:14 PM



Да — и здесь есть очень красивый смысл, почти буквально **эйлерова характеристика дерева**.

T считает некорневые деревья с **выделенной вершиной**. Для каждого дерева на n вершинах таких вариантов n .

А

$$\frac{T^2}{2} = \text{SET}_2(T)$$

считает деревья с **выделенным неориентированным ребром**: если разрезать отмеченное ребро, получаются два корневых дерева, причём порядок сторон не важен. У дерева на n вершинах ребер ровно $n - 1$.

Поэтому для каждого отдельного дерева:

$$\#V - \#E = n - (n - 1) = 1.$$

И при суммировании по всем деревьям получается

$$U = T - \frac{1}{2}T^2.$$

То есть формула — это фактически

$$\chi(\text{tree}) = V - E = 1.$$

Это уже вполне топологический смысл: конечное дерево — контрактивный одномерный комплекс, поэтому его эйлерова характеристика равна 1.

Есть и более геометрическая версия через **центр дерева**. У каждого конечного дерева есть канонический центр — либо одна вершина, либо одно ребро; его можно найти, последовательно снимая листья. На этом строится общая теорема о диссимметрии деревьев:

$$U + (\text{дерево с ориентированным ребром}) \cong (\text{дерево с отмеченной вершиной}) + (\text{дерево с отмеченным ребром}).$$

Здесь

$$\text{ориентированное ребро} \leftrightarrow T^2, \quad \text{неориентированное ребро} \leftrightarrow \frac{T^2}{2},$$

поэтому

$$U + T^2 = T + \frac{1}{2}T^2,$$

и снова

$$U = T - \frac{1}{2}T^2.$$

Но для нашего уровня первая интерпретация даже красивее:

$$\text{unrooted tree} = \text{vertex-pointed trees} - \text{edge-pointed trees}$$

потому что у каждого дерева

$$V - E = 1.$$

То есть да: формула, которую мы получили чисто аналитически из $T = xe^T$, на самом деле скрывает элементарную топологию дерева.

10/2/2026, 3:35:26 PM · gpt-5-6-thinking





Да, $V - E = 1$, это очевидно, и связь с T и $T^{2/2}$ тоже очень ясная. Ну что ж, следующий листок.

10/2/2026, 3:37:56 PM



Отлично.

Листок 4. Разбиения множества: Bell, Stirling и производная

Здесь будет примерно 9 задач. Главные цели:

$$\text{SET}(\text{SET}_{\geq 1}(X)), \quad e^{e^x - 1},$$

а затем понять комбинаторно, почему

$$\Pi' = e^X \Pi$$

и как из этого возникает рекурсия Белла.

Задача 1

Берём конечное множество различных меток, например

$$\{1, 2, 3, 4, 5\},$$

и разбиваем его на непустые блоки, например

$$\{\{1, 4\}, \{2, 5\}, \{3\}\}.$$

Ответь на три связанных вопроса:

1. Что здесь является атомом X ?
2. Какой тип имеет **один блок**?
3. Как из блоков строится всё разбиение - `SET`, `SEQ` или что-то ещё?

Попробуй сразу собрать из этого одно уравнение для типа Π всех разбиений.

10/2/2026, 3:38:02 PM · gpt-5-6-thinking





Атомом является элемент/метка. Один блок имеет тип $\exp(X)-1$ - множество непустых атомов. Разбиение - это множество блоков: $\Pi = \exp(\exp(X)-1)$.
 $\Pi' = \exp(\exp(x)-1) \times \exp(x) = \Pi \times \exp(X)$ - формальные ряды.

10/2/2026, 3:43:00 PM



Да — по существу ты сразу решил несколько шагов вперёд. Но здесь важно снова развести **тип** и **его EGF**.

На уровне типов:

$$B = \text{SET}_{\geq 1}(X)$$

— один непустой блок, и

$$\Pi = \text{SET}(B) = \text{SET}(\text{SET}_{\geq 1}(X)).$$

А уже на уровне EGF:

$$B(x) = e^x - 1,$$

поэтому

$$\Pi(x) = e^{e^x - 1}.$$

То есть твоя запись была правильной по содержанию, но e^X — это уже скорее язык производящих функций, а не сам тип.

И производную ты тоже получил правильно:

$$\Pi'(x) = e^{e^x - 1} e^x = \Pi(x) e^x.$$

На уровне species это соответствует

$$\Pi' = \Pi \cdot \text{SET}(X).$$

Теперь важнее всего понять это **без дифференцирования формулы**.

Задача 2

Берём разбиение и заменяем один атом дыркой.

Например:

$$\{\{1, 4\}, \{2, \square, 5\}, \{3, 6\}\}.$$

Разбей такую структуру на две части:

1. что остаётся от блока, в котором была дырка;
2. что представляют собой все остальные блоки.

Из этого попробуй самостоятельно объяснить, почему

$$\Pi' = \text{SET}(X) \cdot \Pi.$$

10/2/2026, 3:43:10 PM · gpt-5-6-thinking



У нас есть пара: любое множество меток (в том числе пустое, если был один элемент и теперь на его месте дырка) $\text{SET}(X)$ и множество непустых множеств, то есть Π . Отсюда, $\Pi' = \text{SET}(X) \times \Pi$.

10/2/2026, 3:49:47 PM



Да, именно.

$$\Pi' = \text{SET}(X) \cdot \Pi$$

Комбинаторно:

- бывший блок с дыркой после удаления дырки превращается в произвольное множество обычных атомов, возможно пустое;
- все остальные блоки образуют обычное разбиение.

И снова важно: это произведение species, то есть общий набор меток делится между этими двумя частями.

Задача 3

Переведи это структурное равенство в EGF и затем сравни коэффициенты.

То есть из

$$\Pi' = \text{SET}(X) \cdot \Pi$$

получи формулу для $\Pi'(x)$, а затем рекурсию для чисел Белла B_n , если

$$\Pi(x) = \sum_{n \geq 0} B_n \frac{x^n}{n!}.$$

Попробуй сначала сам дойти до суммы.

10/2/2026, 3:49:55 PM · gpt-5-6-thinking



$\Pi' = \exp(x) \cdot \Pi$. С другой стороны, $\Pi' = \sum (B_{n+1}/n! \cdot x^n)$. То есть,
 $B_{n+1}/n! = \sum (1/k! \cdot B_{n-k}/(n-k)!)$, $B_{n+1} = \sum (C(n, k) \cdot B_{n-k}) = \sum (C(n, k) \cdot B_k)$

10/2/2026, 4:04:33 PM



Да, всё правильно.

Из

$$\Pi'(x) = e^x \Pi(x)$$

получаем

$$\sum_{n \geq 0} B_{n+1} \frac{x^n}{n!} = \left(\sum_{k \geq 0} \frac{x^k}{k!} \right) \left(\sum_{m \geq 0} B_m \frac{x^m}{m!} \right).$$

Поэтому

$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_{n-k}$$

и после замены индекса $k \mapsto n - k$:

$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k.$$

То есть ты получил стандартную рекурсию Белла напрямую из производной species.

И важно, что у неё есть очень прозрачный комбинаторный смысл: фиксируем один специальный элемент $*$, выбираем k из остальных n элементов, которые **не** лежат с ним в одном блоке — или, эквивалентно, выбираем элементы его блока; оставшиеся элементы разбиваются произвольно. Обе формы суммы эквивалентны.

Задача 4. Теперь введём ещё одну переменную u , которая будет считать число блоков. Какую двухпеременную EGF ты ожидаешь для разбиений, если каждый блок должен давать дополнительный множитель u ?

10/2/2026, 4:04:36 PM · gpt-5-6-thinking





Было $\exp(\exp(x)-1)$, где $\exp(x)-1$ отвечает за число элементов. Мне кажется, $\exp(u \cdot (\exp(x)-1))$, но я не уверен.

10/2/2026, 4:08:41 PM



Да, именно:

$$\Pi(x, u) = \exp(u(e^x - 1)).$$

Почему: один непустой блок по-прежнему даёт

$$e^x - 1,$$

а если каждый блок должен дополнительно иметь вес u , то один блок даёт

$$u(e^x - 1).$$

Разбиение — это **SET** блоков, значит внешняя экспонента:

$$\text{SET}(u \cdot \text{SET}_{\geq 1}(X)) \longrightarrow \exp(u(e^x - 1)).$$

И теперь коэффициенты имеют смысл

$$\Pi(x, u) = \sum_{n, k \geq 0} S(n, k) u^k \frac{x^n}{n!},$$

где $S(n, k)$ — числа Стирлинга второго рода.

Следующая задача.

Задача 5

Что комбинаторно означает

$$\frac{\partial}{\partial u} \Pi(x, u)?$$

И что получится, если после дифференцирования положить $u = 1$?

10/2/2026, 4:08:45 PM · gpt-5-6-thinking





Комбинаторно мы выкидываем целый блок и заменяем его дыркой [знаю, но не уверен почему]. Если после дифференцирования положить $u = 1$, получаем $\Pi \times (\exp(x) - 1) = \Pi' - \Pi$. Вопрос, что бы это значило комбинаторно? дифференцирование по u - это выкинули целый блок, $u=1$ - заменили дырку пустым множеством??

10/2/2026, 4:17:54 PM



Да — здесь как раз важно остановиться: ты смешал **две разные производные**.

Производная по X у species действительно означает «вынуть атом и оставить дырку». Но u — не атом. Это просто **маркер числа блоков**.

Если разбиение имеет k блоков, оно получает вес

$$u^k.$$

Дифференцируем:

$$\frac{d}{du} u^k = k u^{k-1}.$$

Множитель k означает:

выбрать один из k блоков и выделить его.

Блок никуда не исчезает. Его элементы остаются на месте. Мы просто говорим: «вот этот блок специальный».

Поэтому

$$\frac{\partial}{\partial u} \Pi(x, u)$$

считает разбиения **с одним выделенным блоком**, а u^{k-1} продолжает считать остальные невыделенные блоки.

Теперь ставим $u = 1$. Это не «заполнить дырку пустым множеством», а просто:

перестать учитывать число остальных блоков.

Получаем

$$\left. \frac{\partial \Pi}{\partial u} \right|_{u=1} = (e^x - 1) \Pi(x).$$

И это совершенно понятно структурно:

выделенный непустой блок $\times$ обычное разбиение остальных элементов.

То есть

$$\text{SET}_{\geq 1}(X) \cdot \Pi.$$

А теперь твоё

$$(e^x - 1) \Pi = \Pi' - \Pi$$

имеет очень красивый комбинаторный смысл.

Мы знаем:

$$[x^n/n!] \Pi' = B_{n+1}, \quad [x^n/n!] \Pi = B_n.$$

Следовательно число разбиений n -элементного множества с выделенным блоком равно

$$B_{n+1} - B_n.$$

Почему?

Добавим новый элемент $*$ и рассмотрим разбиения $n + 1$ элементов.

Их B_{n+1} . Из них ровно B_n имеют singleton-блок

$$\{*\}.$$

Во всех остальных $*$ сидит в блоке ещё с какими-то элементами. Удаляем $*$ — и тот блок, где он находился, становится **выделенным непустым блоком** разбиения исходных n элементов.

Получаем биекцию:

$$\{\text{разбиения } n + 1 \text{ элементов, где } * \text{ не singleton}\} \leftrightarrow \{\text{разбиения } n \text{ элементов с выделенным блоком}\}.$$

Поэтому

$$\sum_{\pi \in \Pi_n} \# \text{blocks}(\pi) = B_{n+1} - B_n.$$

И я бы отметил это как ещё один ключевой принцип:

$$\frac{\partial}{\partial X} \text{ — дырка в атоме; } \quad \frac{\partial}{\partial u} \text{ — выделение объекта, который } u \text{ считает.}$$

Это разные операции, хотя обе выглядят как «дифференцирование».

10/2/2026, 4:18:04 PM · gpt-5-6-thinking



Пассивно понятно, активно не очень. Возможно, у нас ещё будут примеры частного дифференцирования в дальнейшем, и станет понятней.

10/2/2026, 4:25:54 PM



Да, именно так и сделаем. Сейчас достаточно удержать только правило на уровне узнавания:

если u отмечает число каких-то компонент, то $\frac{\partial}{\partial u}$ выделяет одну из них.

Не надо пока считать это освоенным активно.

Дальше у нас ещё будут удобные случаи с дополнительными маркерами — например, число циклов у перестановок или число компонент у endofunctions. Там одна и та же идея повторится несколько раз, и тогда можно будет сделать отдельный маленький листок именно на частные производные.

А текущий листок лучше продолжить без застревания на этом месте. Следующий шаг можно сделать снова более структурным.

10/2/2026, 4:25:59 PM · gpt-5-6-thinking





Хорошо. Продолжим.

10/2/2026, 4:46:35 PM



Продолжим с более устойчивого места.

Задача 6

Пусть $S(n, k)$ — число разбиений n -элементного множества ровно на k непустых блоков.

Добавим новый элемент $*$ к множеству из n элементов. Разбиение $n + 1$ элементов на k блоков можно получить двумя взаимноисключающими способами:

1. $*$ образует отдельный одноэлементный блок;
2. $*$ входит в один из уже существующих блоков.

Попробуй из этого получить рекурсию для $S(n + 1, k)$ через числа с n элементами.

10/2/2026, 4:46:43 PM · gpt-5-6-thinking



1. Не считая блока с $\setminus^*$ у нас n элементов в $k-1$ блоке (взаимно-однозначное соответствие). 2. Не считая $\setminus^*$ у нас n элементов в k блоках, но один блок из k выделен, что умножает ситуацию на k . $S(n+1, k) = S(n, k-1) + k \times S(n, k)$.

10/2/2026, 6:24:36 PM



Да, точно:

$$S(n+1, k) = S(n, k-1) + k S(n, k).$$

И ты очень хорошо сформулировал второй случай: после удаления $*$ остаётся разбиение на k блоков, причём нужно ещё выбрать, **в какой из этих k блоков** вернуть $*$. Отсюда множитель k .

Это уже почти активное понимание той самой идеи «маркер/выделение компоненты»: здесь множитель k возник не из формальной производной, а из прямого выбора одного из k блоков.

Задача 7

Теперь попробуем получить ту же рекурсию из двухпеременной EGF

$$\Pi(x, u) = \exp(u(e^x - 1)).$$

Найди $\partial \Pi / \partial x$ и постарайся переписать результат через Π и $\partial \Pi / \partial u$. Затем сравним коэффициенты.

10/2/2026, 6:24:38 PM · gpt-5-6-thinking





$d\Pi/dx = \exp(u \times (\exp(x)-1)) \times u \times \exp(x) = \Pi \times u \times \exp(x)$; $d\Pi/du = \exp(u \times (\exp(x)-1)) \times (\exp(x)-1) = \Pi \times (\exp(x)-1) = \Pi \times \exp(x) - \Pi$. $d\Pi/dx = (d\Pi/du + \Pi) \times u$.

10/2/2026, 6:31:47 PM



Да, всё точно:

$$\frac{\partial \Pi}{\partial x} = u \left(\frac{\partial \Pi}{\partial u} + \Pi \right).$$

И это очень приятное уравнение: слева мы «добавляем один атом», а справа возникают те же два случая, что ты только что разобрал комбинаторно.

Теперь **Задача 8** — сравнить коэффициенты.

У нас

$$\Pi(x, u) = \sum_{n, k \geq 0} S(n, k) u^k \frac{x^n}{n!}.$$

Попробуй определить коэффициент при

$$u^k \frac{x^n}{n!}$$

отдельно в трёх выражениях:

$$\frac{\partial \Pi}{\partial x}, \quad u\Pi, \quad u \frac{\partial \Pi}{\partial u}.$$

Из этого должна буквально выпасть уже знакомая рекурсия для $S(n+1, k)$.

10/2/2026, 6:31:53 PM · gpt-5-6-thinking





$d\Pi/dx: S(n+1, k)$, $u\Pi: S(n, k-1)$, $u d\Pi/du: kS(n, k)$. Да, работает.

10/2/2026, 6:42:42 PM



Да. И это как раз хороший активный пример того, что раньше было только «пассивно понятно»:

$$\frac{\partial}{\partial u}$$

действительно вносит множитель k , потому что выбирает один из k блоков.

Получили:

$$S(n+1, k) = S(n, k-1) + kS(n, k)$$

двумя путями:

- напрямую по комбинаторике;
- из PDE для $\Pi(x, u)$.

Это очень полезное совпадение: формула не просто верна, а отражает тот же разбор на случаи.

Задача 9

Последняя на листке.

Возьми

$$\Pi(x, u) = \exp(u(e^x - 1))$$

и положи $u = 1$.

Что получится? Как связаны числа Белла B_n и числа Стирлинга второго рода $S(n, k)$?

10/2/2026, 6:42:45 PM · gpt-5-6-thinking



$\Pi(x, 1) = \Pi(x)$, то есть сумма $S(n, k) \text{ } k \geq 0 = B_n$.

10/2/2026, 6:45:46 PM



Да, именно:

$$\Pi(x, 1) = \Pi(x)$$

и потому для каждого n

$$B_n = \sum_{k=0}^n S(n, k) .$$

То есть число Белла просто суммирует разбиения по всем возможным числам блоков.

На этом **Листок 4** завершён. И по ощущениям он уже получился заметно лучше по темпу: достаточно мелко, чтобы не прыгать через ключевые места, но без чрезмерной дробности.

Следующий естественный листок — **endofunctions**. Там как раз соберутся вместе три операции, которые мы уже видели отдельно:

SET, CYC, T,

и получится структура

$$\text{End} = \text{SET}(\text{CYC}(T)).$$

Это хороший итоговый синтез.

10/2/2026, 6:45:48 PM · gpt-5-6-thinking





Хорошо. Попробуем Endofunctions.

10/2/2026, 7:33:33 PM



Хорошо.

Листок 5. Endofunctions

Пусть E - класс отображений

$$f : S \rightarrow S$$

конечного помеченного множества S в себя.

Каждый элемент $v \in S$ считаем атомом; значит размер - это $|S|$. Удобно изображать функцию ориентированным графом: из каждой вершины v идёт ровно одна стрелка

$$v \rightarrow f(v).$$

Задача 1

Возьми одну **связную компоненту** такого графа.

Начни с произвольной вершины v и последовательно иди:

$$v, f(v), f^2(v), \dots$$

Ответь на два связанных вопроса:

1. Почему в конечном множестве мы обязательно рано или поздно попадём в уже встречавшуюся вершину?
2. Что тогда обязательно находится внутри каждой связной компоненты функционального графа?

Пока не пытайся записывать `СУС(Т)` - сначала только геометрия графа.

10/2/2026, 7:33:40 PM · gpt-5-6-thinking





1. Потому что множество конечное - несколько странный вопрос. Если бы все вершины были бы разные, то их было бы [минимум] счётное количество. 2. Цикл.

10/2/2026, 7:35:44 PM



Да. Первый подпункт действительно был слишком тривиальным; можно было сразу спросить содержательную вещь.

Ты получил главное:

в каждой связной компоненте есть цикл.

Теперь **Задача 2.**

Нужно понять структуру всей компоненты.

1. Почему в одной связной компоненте функционального графа не может быть двух разных циклов?
2. Если взять вершину цикла и посмотреть на все вершины, которые в конце концов попадают именно в неё, что за структура получается вокруг этой вершины?

Цель — увидеть, что одна компонента состоит из **одного цикла, а к его вершинам прикреплены корневые деревья.**

10/2/2026, 7:35:48 PM · gpt-5-6-thinking





1. В последовательности может быть префикс, затем первое появление повторяющейся вершины, затем цикл. Две последовательности, заканчивающиеся разными циклами не пересекаются, поскольку в каждой точке продолжение однозначно. С другой стороны префиксов, заканчивающихся тем же циклом может быть много, они могут входить в цикл в разных точках, и поскольку префиксы ациклические, те из них, которые заканчиваются в конкретной точке цикла, должны формировать дерево (возможно, тривиальное, линейное).

10/2/2026, 7:41:48 PM



Да, именно. И это уже правильное структурное доказательство.

У каждой вершины есть единственный «будущий путь». Поэтому если две траектории когда-нибудь встретились, дальше они совпадают навсегда. Значит, вершины, уходящие в разные циклы, не могут лежать в одной связной компоненте.

А для фиксированной вершины цикла все вершины, которые впервые попадают в цикл именно через неё, действительно образуют **корневое дерево**:

- корень — эта вершина цикла;
- остальные стрелки направлены к корню;
- ветвлений назад может быть сколько угодно;
- циклов внутри этой части уже нет.

То есть одна связная компонента выглядит как

цикл, в каждой вершине которого сидит корневое дерево.

Причём тривиальное дерево допустимо: вершина цикла может не иметь никаких дополнительных предшественников.

Задача 3

Мы уже обозначали через T тип корневых помеченных деревьев.

Как теперь записать тип **одной связной компоненты** endofunction?

Нужно только выбрать между чем-то вроде $\text{SEQ}(T)$, $\text{SET}(T)$, $\text{Cyc}(T)$ и объяснить почему.

10/2/2026, 7:41:55 PM · gpt-5-6-thinking





Сус(T) - цикл, в котором каждый элемент представляет корневое дерево.

10/2/2026, 7:43:24 PM



Да, точно:

$$C = \text{CYC}(T)$$

где C — тип одной связной компоненты endofunction.

Почему не **SEQ** и не **SET** :

- не **SEQ** , потому что у цикла нет выделенного начала;
- не **SET** , потому что у вершин цикла есть циклический порядок.

Задача 4

Вся endofunction может иметь несколько связных компонент, и порядок этих компонент не важен.

Как записать весь тип E через C ? А затем подставь $C = \text{CYC}(T)$.

10/2/2026, 7:43:26 PM · gpt-5-6-thinking



Set(Cyc(T))

10/2/2026, 7:44:11 PM



Да:

$$E = \text{SET}(\text{CYC}(T)).$$

Это главный структурный результат всего листка.

Теперь **Задача 5**.

Если $A(x)$ — EGF помеченного класса A , то какова EGF для

$$\text{CYC}(A)?$$

Попробуй вывести её, а не вспоминать готовую формулу:

- упорядоченная k -ка A -объектов даёт $A(x)^k$;
- сколько раз один и тот же цикл посчитан среди всех линейных записей длины k ?
- что получится после суммирования по $k \geq 1$?

10/2/2026, 7:44:16 PM · gpt-5-6-thinking



- A^k
- Цикл длины k можно повернуть k раз [то есть, он посчитан k раз], A^k/k , $k \geq 1$
- $\sum (a^k/k)$, $k \geq 1$ - это ряд Лейбница для $-\log(1-a)$

10/2/2026, 7:47:41 PM



Да. Всё правильно, только маленькая терминологическая поправка:

$$\sum_{k \geq 1} \frac{a^k}{k} = -\log(1-a)$$

обычно называют рядом Меркатора, не рядом Лейбница.

Итак,

$$\text{CYC}(A) \longleftrightarrow -\log(1 - A(x)).$$

Для нашей компоненты:

$$C(x) = -\log(1 - T(x)).$$

Задача 6

Теперь используй

$$E = \text{SET}(C)$$

и правило

$$\text{SET}(A) \longleftrightarrow e^{A(x)}.$$

Во что превращается $E(x)$? Постарайся упростить до совсем простой функции от $T(x)$.

10/2/2026, 7:47:45 PM · gpt-5-6-thinking



Set $\rightarrow \exp; \exp(-\ln(1-T)) = 1/\exp(\ln(1-T)) = 1/(1-T)$, то есть Seq(T)

10/2/2026, 7:51:07 PM



Да — вычисление совершенно правильное:

$$E(x) = \frac{1}{1 - T(x)}.$$

Но вот последний шаг

«то есть Seq(T)»

нужно произнести осторожнее:

у E и Seq(T) одинаковые EGF

но это **не означает автоматически**, что

$$E \cong \text{Seq}(T)$$

как species.

Это ещё один важный принцип: **производящая функция забывает часть структуры**. Разные типы могут иметь одинаковое число объектов каждого размера.

Здесь

$$\text{SET}(\text{CYC}(T)) \longrightarrow \exp(-\log(1 - T)) = \frac{1}{1 - T},$$

а отдельно

$$\text{Seq}(T) \longrightarrow 1 + T + T^2 + \dots = \frac{1}{1 - T}.$$

Ряды совпали, но структурные описания совершенно разные:

множество циклов деревьев

против

линейная последовательность деревьев.

Это, я бы сказал, **ключевой момент №4**:

одинаковые EGF $\nRightarrow$ одинаковые species.

А теперь **Задача 7**.

С другой стороны, endofunction $f : [n] \rightarrow [n]$ можно посчитать вообще без всей нашей структурной теории.

Сколько существует таких функций?

И что тогда должно быть коэффициентом при $x^n/n!$ в

$$\frac{1}{1 - T(x)}?$$

10/2/2026, 7:51:17 PM · gpt-5-6-thinking



Вопрос: я понимаю, что одинаковые EGF не означают одинаковые структуры. С другой стороны, можно надеяться на естественный изоморфизм между структурами (который не обязан существовать). Он существует?

10/2/2026, 7:53:14 PM



Да — здесь очень красивый ответ:

как species — нет; как классы на $[n]$ после выбора порядка меток — да.

То есть равенство

$$\text{SET}(\text{CYC}(T)) \quad \text{и} \quad \text{SEQ}(T)$$

на уровне EGF действительно намекает на биекцию, и такую биекцию можно построить. Но она **не естественна относительно произвольной перенумерации меток**, поэтому это не изоморфизм species.

Почему не species-изоморфизм, можно увидеть уже при $n = 2$. Рассмотрим перестановку меток $1 \leftrightarrow 2$.

Для endofunctions две функции остаются неизменными при такой перенумерации:

$$1 \mapsto 1, 2 \mapsto 2$$

и

$$1 \mapsto 2, 2 \mapsto 1.$$

А у $\text{SEQ}(T)$ на двух метках ни одна структура не фиксируется обменом $1 \leftrightarrow 2$: либо у нас одно двухвершинное корневое дерево, и меняется корень, либо два одноточечных дерева в последовательности, и меняется их порядок.

Значит, действия группы S_2 различны, поэтому натуральной биекции species быть не может.

Но если метки уже имеют порядок

$$1 < 2 < \dots < n,$$

то можно сделать вполне красивую явную биекцию. Это по сути вариант преобразования Фохта.

Начинаем с

$$\text{SET}(\text{CYC}(T)).$$

В каждом цикле деревьев:

1. поворачиваем цикл так, чтобы первым стояло дерево, корень которого имеет **наибольшую метку** в этом цикле;
2. упорядочиваем сами циклы по этим максимумам;
3. стираем скобки между циклами.

Получается одна последовательность деревьев:

$$\text{SEQ}(T).$$

И главное — границы циклов можно восстановить: начала циклов определяются по record maxima в последовательности корневых меток. Это ровно тот же трюк, который превращает циклическую запись перестановки в линейную.

Так что здесь есть очень красивое различие:

одинаковая EGF $\Rightarrow$ равномощность в каждом размере,

иногда даже

явная биекция после выбора порядка меток,

но не обязательно



натуральный изоморфизм species.

И это, пожалуй, **ключевой момент №5**: species помнят не только количество структур, но и то, **как структуры ведут себя при перенумерации меток**. EGF эту информацию забывает.

10/2/2026, 7:53:43 PM · gpt-5-6-thinking



Задача 7. Таких функций, очевидно, n^n (любое в любое), и это коэффициент.

10/2/2026, 8:00:09 PM



Да, именно.

$$|E[n]| = n^n$$

потому что каждый из n элементов независимо выбирает один из n образов.

А поскольку

$$E(x) = \frac{1}{1 - T(x)}$$

— это EGF, получаем

$$\frac{1}{1 - T(x)} = \sum_{n \geq 0} n^n \frac{x^n}{n!}.$$

То есть структурная декомпозиция endofunctions приводит к тождеству, в котором коэффициенты оказываются тривиально понятными напрямую.

Следующая задача.

Задача 8

Мы уже знаем:

$$T(x) = \sum_{n \geq 1} n^{n-1} \frac{x^n}{n!}.$$

Почему формула

$$\frac{1}{1 - T(x)}$$

комбинаторно очень естественно даёт n^n , если помнить, что $1/(1 - T)$ — это EGF последовательности корневых деревьев после выбора линейного порядка меток?

Попробуй объяснить это не формулой, а словами.

10/2/2026, 8:00:11 PM · gpt-5-6-thinking





То есть, есть список корневых деревьев и общее количество вершин равно n . Добавим объединяющий корень. Получим одно корневое дерево с $n+1$ вершиной. Таких деревьев $(n+1)^n$. Что-то не получается..

10/2/2026, 8:10:57 PM



Да — и здесь твоя попытка очень полезна, потому что она показывает, где я в задаче 8 сформулировал слишком быстро.

Если взять **список** корневых деревьев и добавить общий корень, то порядок деревьев в списке никуда естественно не девается. А в обычном корневом дереве дети нового корня **не упорядочены**. Поэтому это не биекция.

Кроме того, если новый корень имеет заранее фиксированную новую метку, то обычных деревьев на $n + 1$ вершинах с этим фиксированным корнем было бы

$$(n + 1)^{n-1},$$

а не $(n + 1)^n$.

Так что твоё «что-то не получается» — совершенно правильный сигнал.

Правильная словесная интерпретация

$$\frac{1}{1 - T}$$

здесь идёт не через добавление общего корня, а через ту биекцию, которую мы только что обсуждали:

$$\text{SET}(\text{CYC}(T)) \longleftrightarrow \text{SEQ}(T)$$

после выбора порядка меток.

То есть:

1. endofunction раскладывается на циклы корневых деревьев;
2. каждый цикл канонически разрезаем, используя порядок меток;
3. циклы канонически упорядочиваем;
4. стираем разделители.

Получаем последовательность корневых деревьев.

А поскольку endofunctions на n метках ровно

$$n^n,$$

то и таких канонически кодированных последовательностей тоже n^n .

Так что задача 8 в моей формулировке была неудачной: я как будто подталкивал к «простому» объяснению через список деревьев, которого в таком виде нет. Лучше считать её исправленной:

$$\frac{1}{1 - T} \text{ имеет коэффициенты } n^n \text{ потому, что } \text{SEQ}(T) \text{ биективно endofunctions после выбора порядка меток.}$$

И это как раз ещё один хороший пример того, что одинаковая EGF иногда скрывает **нетривиальную биекцию**, а не очевидную конструкцию.

10/2/2026, 8:11:06 PM · gpt-5-6-thinking





Хорошо, но как это естественно объяснить через коэффициенты для T ? Мне казалось, что должен быть путь через объединение в дерево.

10/2/2026, 8:14:47 PM



Да! Твоя интуиция была правильной: **есть очень естественный путь через объединение списка корневых деревьев в одно дерево**. Только соединять их надо не общим корнем, а **в цепочку — spine**.

Пусть дана последовательность корневых деревьев

$$(T_1, T_2, \dots, T_k),$$

на непересекающихся множествах меток, всего n вершин. Обозначим их корни

$$r_1, r_2, \dots, r_k.$$

Добавим ещё одну специальную вершину $*$ и проведём новые рёбра

$$r_1 - r_2 - r_3 - \dots - r_k - *.$$

Корнем всего получившегося дерева объявим r_1 .

То есть список деревьев превращается в одно дерево, у которого есть выделенный «хребет»

$$r_1 \rightarrow r_2 \rightarrow \dots \rightarrow r_k \rightarrow *,$$

а исходные T_i висят на соответствующих вершинах хребта.

И это обратимо! Если нам дали корневое дерево на

$$\{1, \dots, n, *\},$$

где $*$ — лист, берём единственный путь от корня до $*$:

$$r_1, r_2, \dots, r_k, *.$$

Удаляем рёбра этого пути. Получившиеся компоненты, укоренённые в $r_1, \dots, r_k$, и есть исходная последовательность

$$(T_1, \dots, T_k).$$

Поэтому

$$\text{SEQ}(T) \longleftrightarrow \{\text{корневые деревья на } n + 1 \text{ вершинах, где } * \text{ — лист}\}.$$

Теперь считаем правую сторону.

По формуле Кэли дерево на $n + 1$ вершинах кодируется Prüfer-последовательностью длины

$$n - 1.$$

Чтобы специальная вершина $*$ была листом, она **не должна встречаться** в Prüfer-коде. Поэтому каждый из $n - 1$ элементов кода можно выбрать из остальных n меток:

$$n^{n-1}.$$

После этого надо выбрать корень дерева. Он не может быть $*$, потому что в нашей конструкции корень — r_1 , одна из исходных n вершин. Значит ещё n вариантов:

$$n \cdot n^{n-1} = \boxed{n^n}.$$

И вот теперь получается действительно естественное объяснение:

$$\boxed{[x^n/n!] \frac{1}{1 - T(x)} = n^n.}$$



Причём непосредственно через коэффициенты T та же формула разворачивается как

$$n^n = \sum_{k \geq 1} \sum_{n_1 + \dots + n_k = n} \frac{n!}{n_1! \dots n_k!} \prod_{i=1}^k n_i^{n_i-1}.$$

Здесь:

- выбираем размеры деревьев $n_1, \dots, n_k$;
- распределяем n меток между ними:

$$\frac{n!}{n_1! \dots n_k!};$$

- на каждом блоке строим корневое дерево:

$$n_i^{n_i-1}.$$

И вся чудовищная сумма схлопывается в n^n , потому что **список корневых деревьев = одно корневое дерево со специальным листом**.

Так что твоё ощущение «должен быть способ склеить всё в дерево» было очень точным. Ошибка была только в способе склейки: не «общий корень», а **соединить корни последовательности в путь и закончить его новым листом**.

И это, кстати, гораздо приятнее предыдущей ссылки на биекцию через циклы: здесь равенство

$$\text{SEQ}(T) \leftrightarrow n^n$$

становится совершенно геометрически видимым.

10/2/2026, 8:14:58 PM · gpt-5-6-thinking



Несколько тяжеловесно, но красиво. Формула с двойной суммой по тяжести и внешнему виду напоминает формулу Фаа ди Бруно.

10/2/2026, 8:28:21 PM



Да — сходство не случайное. Но точнее сказать так:

наша двойная сумма — частный случай той же комбинаторики, которая стоит за Фаа ди Бруно.

Когда мы извлекаем коэффициент из композиции

$$A(T(x)),$$

для EGF общий ответ выражается через **частичные полиномы Белла**:

$$[x^n/n!] A(T(x)) = \sum_k A^{(k)}(0) B_{n,k}(t_1, t_2, \dots)/k!.$$

Это и есть комбинаторная сторона формулы Фаа ди Бруно: множество n меток разбивается на k блоков, на каждом блоке строится внутренняя T -структура, а затем A организует эти k компонент.

У нас

$$A(z) = \frac{1}{1-z} = \text{SEQ}(z).$$

То есть после разбиения меток на блоки мы ещё **упорядочиваем блоки**. Поэтому вместо просто разбиения множества получаем упорядоченную последовательность блоков. Фактор $k!$ как раз отвечает за их порядок.

И наша формула

$$\sum_{k \geq 1} \sum_{n_1 + \dots + n_k = n} \frac{n!}{n_1! \dots n_k!} \prod_i n_i^{n_i - 1}$$

— это запись через **упорядоченные размеры блоков** $n_1, \dots, n_k$.

Если сгруппировать одинаковые размеры и забыть порядок, начинают буквально появляться коэффициенты полиномов Белла — тот самый аппарат Фаа ди Бруно.

Получается приятное замыкание с предыдущим листком:

set partitions $\longrightarrow$ Bell polynomials $\longrightarrow$ composition of EGFs $\longrightarrow$ Faà di Bruno.

A species дают очень хорошее объяснение тому, почему Фаа ди Бруно выглядит настолько страшно: она просто пытается алгебраически описать все способы **разбить набор входов на блоки и обработать каждый блок внутренней функцией**.

То есть её тяжеловесность — это тень довольно простой комбинаторной операции: **composition/substitution of structures**. Это, кстати, отличный кандидат на отдельный будущий листок.

10/2/2026, 8:28:30 PM · gpt-5-6-thinking





Хорошо. Продолжим?

10/2/2026, 8:30:43 PM



Да. Осталась как раз хорошая **заключительная задача листка про endofunctions** — она соединит производную, цикл и дерево с дыркой.

Задача 9

У нас

$$E = \text{SET}(\text{CYC}(T)).$$

Рассмотрим E' : endofunction, в которой **одна вершина заменена дыркой**.

Попробуй разобрать такую структуру по слоям.

1. Все связные компоненты, **кроме** той, где находится дырка, образуют что?
2. В особой компоненте дырка находится внутри одного из корневых деревьев T , сидящих на цикле. Что остаётся от этого дерева?
3. Если в цикле выделить одно место и «разрезать» цикл в этом месте, во что превращаются остальные T -деревья цикла: в $\text{SET}(T)$, $\text{SEQ}(T)$ или $\text{CYC}(T)$?

Попробуй из этих трёх частей собрать формулу для E' .

10/2/2026, 8:30:52 PM · gpt-5-6-thinking





1. E, то есть SET(CYC(T))
2. T' (?)
3. SEQ(T)
4. E' = E × SEQ(T) × T' (?)

10/2/2026, 8:34:56 PM



Да — всё четыре пункта правильно. Получилось

$$E' = E \cdot \text{SEQ}(T) \cdot T'.$$

И это очень хороший пример **цепного правила на уровне структур**.

Дырка последовательно делает три выбора:

1. Среди компонент **SET** выбирается одна особая компонента, содержащая дырку. Все остальные дают

$$E = \text{SET}(\text{CYC}(T)).$$

2. В особой циклической компоненте выбирается одно из сидящих на цикле деревьев — то, внутри которого дырка. Это дерево превращается в

$$T'.$$

3. Как только одно место цикла выделено, цикл можно разрезать в этом месте. Остальные деревья получают линейный порядок:

$$\text{CYC}'(T) = \text{SEQ}(T).$$

Поэтому

$$E' = E \cdot \text{SEQ}(T) \cdot T'.$$

Это же можно увидеть формально:

$$\begin{aligned} E &= \text{SET}(\text{CYC}(T)), \\ E' &= \underbrace{\text{SET}(\text{CYC}(T))}_E \cdot \underbrace{\text{CYC}'(T)}_{\text{SEQ}(T)} \cdot T'. \end{aligned}$$

То есть буквально

$$(\text{SET} \circ \text{CYC} \circ T)' = (\text{SET}') \cdot (\text{CYC}') \cdot T'.$$

На уровне EGF это проверяется мгновенно:

$$E(x) = \frac{1}{1 - T(x)},$$

поэтому

$$E'(x) = \frac{T'(x)}{(1 - T(x))^2}.$$

А правая сторона нашей структурной формулы даёт

$$E(x) \cdot \frac{1}{1 - T(x)} \cdot T'(x) = \frac{T'(x)}{(1 - T(x))^2}.$$

Так что листок закончился особенно удачно: **цепное правило производной оказалось буквально маршрутом дырки внутрь составной структуры**:

component → cycle position → tree → hole.

Вот это уже, по-моему, очень содержательный активный результат.

10/2/2026, 8:35:08 PM · gpt-5-6-thinking



Похоже, мы закончили с эндофункциями. Какие у тебя планы на следующие листки?

10/2/2026, 8:37:00 PM



Да. Я бы теперь не расширял тему бесконечно, а сделал **несколько целевых листков**, каждый с одной новой идеей и с опорой на уже освоенное.

Мой план такой:

1. Листок 6: повторение и перенос.

Короткий смешанный листок на 8–9 задач без новых понятий: списки, Catalan trees, rooted trees, set partitions, endofunctions. Цель — проверить, что ты сам выбираешь `SEQ/SET/CYC`, OGF/EGF, понимаешь product of species и умеешь читать производную как дырку/pointing.

2. Листок 7: дифференцирование по маркерам.

То место, где пока есть пассивное понимание:

$$\partial/\partial u$$

как «выделить одну компоненту, которую считает u ». Сделать на нескольких примерах: блоки разбиения, циклы перестановки, компоненты графа/endofunction. После 3–4 повторений должно стать активным.

3. Листок 8: composition / substitution of species.

Что значит

$$F(G(X)),$$

почему появляются разбиения множества на блоки, Bell polynomials и откуда растёт Faà di Bruno. Это естественное продолжение твоего замечания о двойной сумме.

4. Листок 9: атомная дырка vs дырка-поддерево / настоящий zipper.

Специально вернуться к нашей путанице:

$$\frac{\partial}{\partial X} \quad \text{vs} \quad \frac{\partial}{\partial Y}$$

для рекурсивного конструктора. Сначала очень простые типы, потом бинарное дерево. Цель — чтобы ты сам мог сказать, что именно удаляется и какой контекст сохраняется.

5. Листок 10: производные регулярных выражений.

Brzozowski derivatives:

$$D_a(R + S), \quad D_a(RS), \quad D_a(R^*)$$

и связь с nullable, residual languages и DFA. Это будет очень хороший transfer-test: та же «алгебра производной», но совсем другой объект.

6. Листок 11: перестановки и циклы.

Небольшой, но красивый:

$$\text{PERM} = \text{SET}(\text{CYC}(X)),$$

числа Стирлинга первого рода, marker for number of cycles, differentiation with respect to marker. Это одновременно закрепит `CYC` и частные производные.

7. Листок 12: synthesis.

Несколько задач, где тип структуры не дан. Нужно самому распознать:



SET, SEQ, CYC, composition, pointing, derivative.

Это уже будет тест именно на активное владение.

Я бы сейчас сделал **Листок 6 — повторение и перенос**, прежде чем вводить ещё одну большую тему. Он покажет, что уже действительно перешло из пассивного в активное, а что надо ещё раз прогнать.

10/2/2026, 8:37:04 PM · gpt-5-6-thinking



Хорошо. Продолжаем. Листок 6.

10/3/2026, 10:32:57 AM



Хорошо.

Листок 6. Повторение и перенос

Здесь **новых понятий почти не будет**. Цель - посмотреть, что уже можно восстановить самому. Около 9 задач; идём по одной.

Задача 1. Две модели деревьев

Сравним два уже знакомых класса.

А. Полные плоские бинарные деревья Каталана, где размером считаем число листьев.

В. Произвольные корневые помеченные деревья, где размером считаем число вершин.

Для каждого класса скажи:

1. что является атомом X ;
2. каково структурное уравнение;
3. OGF или EGF естественнее использовать и почему.

Можно отвечать компактно: сначала А, потом В.

10/3/2026, 10:33:18 AM · gpt-5-6-thinking



1. A: лист, B: вершина, 2.A: $T = X + T^2$ [единственный лист или вершина и два поддерева.] B: $T = \text{пары (выделенная корневая вершина, множество [неупорядоченное] поддеревьев)} = X \cdot \exp(T)$. [Если поддеревьев нет получаем пару $(X, 1)$ - одна вершина]. 3. A: OGF - мы считаем только различные формы, без нумерации. Умножение в OGF нам подходит. B: EGF - вершины перенумерованы, номера учитываются в перечислении, при перемножении нам нужно выбрать часть номеров из общего набора. EGF так и работает.

10/3/2026, 12:59:50 PM



Да, почти всё правильно. Две небольшие поправки — обе важные.

Для **A** структурно лучше сказать

$$T = X + T^2$$

как «либо один лист X , либо **упорядоченная пара поддеревьев**». Внутренняя вершина здесь не атом и размера не добавляет.

Для **B** ты уже слегка смешал тип и его EGF. На уровне типов:

$$T = X \cdot \text{SET}(T).$$

A уже после перехода к EGF:

$$T(x) = x e^{T(x)}.$$

Твоё объяснение выбора OGF/EGF — точное: в A считаем формы, в B при сборке компонентов надо ещё распределять метки между ними.

Задача 2. Узнать операцию по описанию

Для каждого описания назови наиболее естественную операцию: **SET**, **SEQ**, **CYC** или произведение.

1. Несколько неупорядоченных корневых деревьев.
2. Несколько корневых деревьев в линейном порядке.
3. Корневые деревья расположены по окружности, начала нет.
4. Один специальный блок разбиения и обычное разбиение остальных элементов.

Пока только типовые операции, без производящих функций.

10/3/2026, 12:59:55 PM · gpt-5-6-thinking





1 SET, 2 SEQ, 3 CYC, 4 произведение.

10/3/2026, 1:01:44 PM



Да, все четыре правильно.

Для 4-го пункта точнее:

$$\text{SET}_{\geq 1}(X) \cdot \Pi$$

если речь о разбиении с одним выделенным блоком.

Задача 3. Узнать производящую функцию по операции

Пусть $A(x)$ — EGF помеченного класса A .

Что соответствуют следующим конструкциям?

1. $\text{SET}(A)$
2. $\text{SEQ}(A)$
3. $\text{CYC}(A)$

Попробуй написать три формулы и кратко объяснить, откуда каждая берётся.

10/3/2026, 1:01:47 PM · gpt-5-6-thinking





1 $\exp(A)$: суммируем кортежи A длиной $k = 0$ $[A^k]$, $\inf$ и удаляем порядок, то есть делим на $k!$, 2 $1/(1-A)$ - суммируем упорядоченные кортежи A^k , 3 $-\ln(1-A)$ - суммируем кортежи A^k , но поскольку у нас цикл, каждый кортеж можно объединить с его k поворотами, то есть нужно разделить A^k на k . Результат из формулы Меркатора (которые у нас было принято называть рядом Лейбница).

10/3/2026, 1:08:21 PM



Да, всё три формулы и объяснения правильные:

$$\text{SET}(A) \leftrightarrow e^A$$

$$\text{SEQ}(A) \leftrightarrow \frac{1}{1-A}$$

$$\text{CYC}(A) \leftrightarrow -\log(1-A)$$

И да, для $-\log(1-A)$ корректнее говорить **ряд Меркатора**.

Задача 4. Product of species

Пусть F и G — помеченные species, а

$$F(x) = \sum f_n \frac{x^n}{n!}, \quad G(x) = \sum g_n \frac{x^n}{n!}.$$

Объясни словами и формулой, почему для произведения species FG коэффициент на n метках равен

$$\sum_{k=0}^n \binom{n}{k} f_k g_{n-k},$$

и почему это ровно соответствует обычному произведению EGF

$$F(x)G(x).$$

Здесь мне важен именно смысл $\binom{n}{k}$.

10/3/2026, 1:08:24 PM · gpt-5-6-thinking





Произведение species это не просто произведение обычных рядов. Сначала нам нужно распределить метки (то есть выбрать k меток из n), а потом перемножить и сложить, с коэффициентом, соответствующих числу возможностей выбора k меток из n , то есть $C(n, k)$. FG именно это и делает.

10/3/2026, 1:12:08 PM



Да, именно. Я бы только уточнил одну фразу: **обычное произведение EGF как раз и кодирует этот дополнительный выбор меток.**

Для фиксированного k :

- выбираем k меток из n :

$$\binom{n}{k};$$

- на них строим F -структуру:

$$f_k;$$

- на остальных — G -структуру:

$$g_{n-k}.$$

Поэтому

$$(FG)_n = \sum_{k=0}^n \binom{n}{k} f_k g_{n-k}.$$

А факториальная нормировка EGF делает так, что обычное умножение рядов даёт ровно эти биномиальные коэффициенты.

Задача 5. Производная как дырка

Для трёх классов скажи, что означает производная:

1. L' , если L — списки;
2. B' , если $B = X + B^2$ — каталановские деревья, где атомы — листья;
3. T' , если T — корневые помеченные деревья, где атомы — все вершины.

Мне нужен именно **словесный смысл дырки** в каждом случае.

10/3/2026, 1:12:14 PM · gpt-5-6-thinking





Давай не будем торопиться. Если я могу что-то сказать по поводу EGF, это не значит, что я всё понимаю. Разберём пример. Возьмем множество/Set. 1. Я даже не уверен, что я понимаю, как писать OGF для множества. По-моему, она не должна отличаться от OGF для списков, поскольку, с точки зрения формы, это в принципе одно и то же: 1 форма для каждого размера. То есть, $1/(1-x)$. 2. EGF известна: $\exp(x)$. Что интересно: подразумевается, что мы нумеруем элементы, то есть $a_n = n!$. Но эта нумерация теряется за счёт автоморфизмов [мы это проговаривали один раз и только один, что маловато для понимания: форма с точностью до автоморфизмов (OGF) или помеченная структура с точностью до автоморфизмов (EGF)]. Например, если у нас есть корневое дерево с вершиной 1 и листьями 2 и 3, мы можем переставить 2 и 3 местами (автоморфизм) - это то же самое дерево. Таким образом, у нас есть только три дерева, с корнями 1,2,3. Если нумерации нет, но есть выделенная вершина, то у нас только одно дерево: вершина и два листа]. Итак, нумерация в множестве теряется и у нас опять только одно множество размера n . В EGF есть некоторая путаница: $n!/n!/n! = 1/n!$, то есть $n!$ перестановок, $n!$ автоморфизмов, и $1/n!$ - дополнительный коэффициент в EGF. Надеюсь, я правильно понимаю ситуацию. Теперь возьмем пару множеств: $\text{Set} \times \text{Set}$. k элементов слева, $n-k$ справа, итого в паре n элементов. Сколько таких пар. $\exp(x) \times \exp(x) = \exp(2x)$, то есть, таких пар 2^n . При перемножении мы получаем сумму строки треугольника Паскаля $\sum C(n, k) = 2^n$ (было бы неплохо, если бы ты подробно расписал детали умножения $\exp(x) \times \exp(x)$ - мне их тяжело писать.)

10/3/2026, 2:50:50 PM



Да. Ты сейчас подошёл к очень важному месту, и почти всё у тебя правильно. Но в рассуждении про EGF есть **один лишний $n!$** . Его стоит разобрать тщательно.

Для начала: **OGF множества** ты понимаешь правильно. Если мы забыли метки и рассматриваем конечное множество только как форму, то для каждого размера n форма ровно одна:

$$\emptyset, \{\bullet\}, \{\bullet, \bullet\}, \dots$$

поэтому

$$\text{SET}_{\text{unlabeled}}(x) = 1 + x + x^2 + \dots = \frac{1}{1-x}.$$

И да: у списка непомеченных одинаковых атомов тоже ровно одна форма каждого размера, поэтому его OGF та же:

$$\frac{1}{1-x}.$$

Но **SET** и **SEQ** от этого, конечно, не становятся одинаковыми структурами. OGF просто не видит разницы в этом примере. Теперь EGF.

Для species **SET** на **фиксированном наборе меток**

$$S = \{1, \dots, n\}$$

существует ровно **одна** **SET**-структура:

$$\{1, \dots, n\}.$$

Поэтому непосредственно

$$a_n = 1,$$

и EGF:

$$\sum_{n \geq 0} 1 \frac{x^n}{n!} = e^x.$$

То есть здесь не надо сначала говорить « $n!$ нумераций», потом делить на автоморфизмы, а потом ещё раз на $n!$. В определении EGF мы сразу считаем **различные структуры на фиксированном наборе меток**. Их для SET ровно одна.

Но твоя мысль про автоморфизмы очень хорошая, потому что существует **второй способ** получить ту же EGF.

Для любой непомеченной формы A размера n :

$$\#\{\text{различных маркировок } A\} = \frac{n!}{|\text{Aut}(A)|}.$$

После EGF-нормировки делим на $n!$:

$$\frac{n!/|\text{Aut}(A)|}{n!} = \frac{1}{|\text{Aut}(A)|}.$$

Поэтому есть чрезвычайно полезная формула:

$$F(x) = \sum_{\text{непомеченным формам } A} \frac{x^{|A|}}{|\text{Aut}(A)|}.$$

Вот здесь автоморфизмы входят совершенно естественно.

Для множества размера n есть одна непомеченная форма, и

$$|\text{Aut}| = n!,$$

потому что можно произвольно переставить элементы. Поэтому её вклад:

$$\frac{x^n}{n!}.$$

Суммируем:

$$e^x.$$

То есть правильно не

$$\frac{n!}{n!} \frac{1}{n!},$$

а либо просто

$$\frac{1}{n!}$$

из определения EGF, либо через формы и автоморфизмы:

$$\frac{n!/n!}{n!} = \frac{1}{n!}.$$

Твой пример с корневым деревом как раз идеален. Возьмём форму

корень с двумя листьями.

У неё

$$|\text{Aut}| = 2$$

— листья можно переставить.

На метках $\{1, 2, 3\}$ различных маркировок:

$$\frac{3!}{2!} = 3.$$

Действительно: корень можно выбрать как 1, 2 или 3. Поэтому вклад этой одной формы в EGF:

$$3 \frac{x^3}{3!} = \boxed{\frac{x^3}{2}}.$$

А это ровно

$$\frac{x^3}{|\text{Aut}|}.$$

Вот это, мне кажется, стоит запомнить:

OGF: каждая непомеченная форма весит 1;

EGF: каждая непомеченная форма фактически весит $\frac{1}{|\text{Aut}|}$.

Теперь твой главный пример:

SET · SET

Это **упорядоченная пара множеств**

$$(A, B)$$

на общем наборе из n меток, причём

$$A \sqcup B = S.$$

Каждая метка должна решить: пойти налево или направо. Поэтому уже непосредственно:

$$\boxed{2^n}$$

структур.

Посмотрим, как именно это возникает из

$$e^x e^x.$$

Расписываем:

$$e^x = \sum_{k \geq 0} \frac{x^k}{k!},$$

поэтому

$$e^x e^x = \left(\sum_{k \geq 0} \frac{x^k}{k!} \right) \left(\sum_{m \geq 0} \frac{x^m}{m!} \right).$$

Чтобы получить x^n , надо взять степени, сумма которых равна n :

$$k + m = n.$$

Поэтому коэффициент при x^n :

$$\sum_{k=0}^n \frac{1}{k!} \frac{1}{(n-k)!}.$$

Теперь выносим $1/n!$:

$$\frac{1}{k!(n-k)!} = \frac{1}{n!} \frac{n!}{k!(n-k)!} = \frac{1}{n!} \binom{n}{k}.$$

Следовательно,

$$[x^n](e^x e^x) = \frac{1}{n!} \sum_{k=0}^n \binom{n}{k}.$$

А строка Паскаля суммируется в

$$\sum_{k=0}^n \binom{n}{k} = 2^n.$$

Итого

$$e^x e^x = \sum_{n \geq 0} 2^n \frac{x^n}{n!} = e^{2x}.$$

Комбинаторный смысл каждого слагаемого:

$$\binom{n}{k}$$

= выбрать k из n меток для левого множества; оставшиеся $n - k$ автоматически идут вправо.

А сумма

$$\binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{n} = 2^n$$

= все способы разбить метки на «левые» и «правые».

И здесь особенно полезно сравнить с OGF.

Если меток **нет**, то пара множеств общей мощности n определяется только их размерами:

$$(0, n), (1, n-1), \dots, (n, 0).$$

То есть форм ровно

$$n + 1.$$

Поэтому:

$$\frac{1}{1-x} \cdot \frac{1}{1-x} = \frac{1}{(1-x)^2} = \sum_{n \geq 0} (n+1)x^n.$$

Получилась очень хорошая пара для сравнения:

непомеченные SET $\times$ SET	$n + 1$
помеченные SET $\cdot$ SET	2^n

Почему такая огромная разница? Потому что в первом случае видны только **размеры двух частей**, а во втором существенно, **какие именно метки** попали в каждую часть.

Мне кажется, вот этот пример стоит немного подержать: он почти идеально объясняет одновременно OGF, EGF, автоморфизмы и species-product.

10/3/2026, 2:51:01 PM · gpt-5-6-thinking



Честно говоря, не вижу разницу между твоим подходом и моим. В обоих случаях у нас есть $n!$ перестановок и мы делим на $n!$ дважды (коэффициент и размер группы автоморфизмов). Получаем то же самое.

Задача 5. Для трёх классов скажи, что означает производная:

1. (L') , если (L) — списки;
2. (B') , если $(B = X + B^2)$ — каталановские деревья, где атомы — листья;
3. (T') , если (T) — корневые помеченные деревья, где атомы — все вершины.

Моё понимание производной по атому (!) для структуры: пары (фокус, контекст), где фокус - местоположение некоторого атома структуры (считаем, что вместо него дырка), а контекст - информация, которая нужна для полного восстановления [редактирования, хирургии] структуры, начиная с этого атома. Или эквивалентная этому информация. [Это неформальное понимание. Я не понимаю, почему в конкретных случаях мы получаем ту или иную структуру, а не, например, ей информационно эквивалентную. Формально, мы, кажется, производную для произвольной структуры не определяли.] 1. Для списков: пара подсписков, слева и справа - однозначно определяют фокус и контекст, но нет ли альтернативы? 2. Путь вверх: влево или вправо, x для каждой вершины на пути её другое поддереву. Понятно, что это хватит. Непонятно, есть ли другой вариант для той же информации. 3. Например, подлес (множество деревьев), соответствующее дырке x путь к корню, состоящий из атомов и множеств остальных поддеревьев у каждого предка.

10/3/2026, 3:28:31 PM



Да. И сначала про $n!$: **ты прав — между твоим подходом и моим по существу разницы нет.** Я слишком настойчиво пытался «исправить» то, что ты уже правильно понимал.

Если начать с одной непомеченной формы A размера n , то

$$\# \text{различных маркировок} = \frac{n!}{|\text{Aut}(A)|}.$$

А затем EGF делит это число ещё на $n!$:

$$\frac{n! / |\text{Aut}(A)|}{n!} = \frac{1}{|\text{Aut}(A)|}.$$

Для множества

$$|\text{Aut}(A)| = n!,$$

поэтому

$$\frac{n! / n!}{n!} = \frac{1}{n!}.$$

Именно это ты и говорил. Так что здесь ставим галочку: **понимание правильное.**

Теперь гораздо более интересный вопрос о производной.

Формальное определение производной species

Вот чего нам действительно не хватало:

$$F'[U] = F[U \sqcup \{*\}]$$

где $*$ — один новый специальный атом, не принадлежащий набору обычных меток U .

То есть F' -структура — это просто F -структура, в которой один атом **особый**. Мы интерпретируем его как дырку.

Это определение. А все наши «контексты», «пути вверх», «левый и правый список» — уже **различные способы перекодировать F'** .

И отсюда сразу ответ на твой вопрос:

может ли быть другая информационно эквивалентная структура?

Конечно. И обычно их много.

Мы ищем удобный естественный изоморфизм

$$F' \cong G.$$

Не обязаны утверждать, что F' буквально *есть* G .

Ключевое слово здесь **естественный**: биекция должна работать одинаково при любой перенумерации атомов, а не зависеть, например, от того, что метки случайно называются $1, 2, 3, \dots$

1. Список

Формально L' — линейный порядок на

$$U \sqcup \{*\}.$$

Например:

$$a, c, *, d, b.$$

Разрезаем в $*$:

$$(a, c) \quad | \quad (d, b).$$

Получаем

$$L' \cong L \cdot L.$$

Но это **не единственная кодировка**. Можно, например, хранить:

- линейный порядок всех обычных элементов;
- число k , показывающее, после какого элемента вставить дырку.

Для n элементов $k \in \{0, \dots, n\}$.

Это та же информация. Просто $L \cdot L$ гораздо лучше согласуется с нашей алгеброй species.

2. Каталановское дерево

Формально:

$$B'[U] = B[U \sqcup \{*\}],$$

где $*$ занимает место одного **листа**.

Теперь можно хранить само дерево с отмеченным специальным листом — это уже полное определение B' .

Но мы нашли гораздо более полезную перекодировку.

Идём от $*$ к корню. Для каждого предка храним:

- был путь слева или справа — 2;
- второе поддереву — B .

Получаем слой

$$2B,$$

а весь путь —

$$B' \cong \text{SEQ}(2B).$$

Можно было хранить отдельно

(строка L/R, список боковых деревьев),

с условием одинаковой длины. Это тоже эквивалентная информация. `SEQ(2B)` просто удобнее.

3. Произвольное корневое дерево

И здесь твоё описание практически точное.

Формально T' — корневое дерево, в котором одна вершина заменена $*$.

Посмотрим на специальную вершину.

Под ней висит произвольное неупорядоченное множество обычных корневых деревьев:

$$\text{SET}(T).$$

Это твой «подлес, соответствующий дырке».

Теперь идём от неё к корню. Для каждого предка надо сохранить:

- саму вершину предка: X ;
- множество всех его остальных дочерних поддеревьев:

$$\text{SET}(T).$$

Один слой поэтому имеет тип

$$X \cdot \text{SET}(T).$$

Но мы уже знаем:

$$X \cdot \text{SET}(T) = T.$$

Поэтому каждый уровень пути кодируется просто одним T , а уровни упорядочены снизу вверх:

$$\text{SEQ}(T).$$

В итоге

$$T' \cong \text{SET}(T) \cdot \text{SEQ}(T).$$

Это ровно твоя идея:

«подлес у дырки x путь к корню, состоящий из атомов и множеств остальных поддеревьев у каждого предка».

Только мы свернули каждый элемент пути

$$X \cdot \text{SET}(T)$$

обратно в T .

И я бы чуть поправил твоё общее определение.

Не совсем

$$\text{производная} = (\text{focus}, \text{context}).$$

Точнее:

$$F' = \text{контекст с одной атомной дыркой}.$$

А

$$XF'$$

— уже исходная структура с **выделенным настоящим атомом** (pointing).

То есть:

$$F' \quad \text{— вынули атом;}$$

$$XF' \quad \text{— положили атом обратно, но оставили его выделенным.}$$

Это различие нам ещё пригодится.

И главное, что ты сейчас заметил: записи вроде

$$L' \cong L^2, \quad B' \cong \text{SEQ}(2B), \quad T' \cong \text{SET}(T) \text{SEQ}(T)$$

— не «определения производной». Это **теоремы о удобных представлениях уже определённой производной**. Вот этого слоя нам раньше действительно не хватало.

10/3/2026, 3:28:44 PM · gpt-5-6-thinking



Ага. А я недоумевал: сначала говорим о дырке в структуре, а потом получаем новую структуру, причём она всегда одна и та же для каждой конкретной, хотя ту же самую информацию можно закодировать и по-другому. Возникает впечатление, что есть какой-то жесткий алгоритм получения структуры производной. А теперь выясняется, что вариантов много - по крайней мере, теоретически - и это имеет смысл.

10/3/2026, 3:35:52 PM



Да, именно. Это очень важное прояснение.

Есть **жёстко определённый объект**

$$F'[U] = F[U \sqcup \{*\}],$$

то есть F' -структура с одним специальным атомом $*$.

А дальше уже начинается свобода:

$$F' \cong G_1 \cong G_2 \cong \dots$$

где G_i — разные удобные способы кодировать ту же информацию.

Для списка:

$$L' \cong L \times L,$$

но можно было бы хранить «список без дырки + позицию дырки».

Для бинарного дерева:

$$B' \cong \text{SEQ}(2B),$$

но можно хранить путь L/R отдельно от списка боковых поддеревьев.

Для корневого дерева:

$$T' \cong \text{SET}(T) \times \text{SEQ}(T),$$

но и это лишь одна удобная нормальная форма.

То есть никакого единственного «алгоритма, который обязан выдать именно эту структуру» нет. Есть:

определение производной

и затем

поиск удобного естественного представления этой производной.

И вот это уже очень похоже на обычную математику: один и тот же объект можно представлять разными координатами, базисами, кодировками, нормальными формами. Главное — чтобы между представлениями была естественная биекция.

Это, по-моему, снимает именно ту странность, которая тебя всё время беспокоила.

10/3/2026, 3:35:54 PM · gpt-5-6-thinking



Да, снимает, спасибо! Продолжаем?

10/3/2026, 3:36:49 PM



Да.

Задача 6. Производная vs pointing

Пусть T — наши **корневые помеченные деревья** на n вершинах.

Разбери три объекта:

1. Что формально означает T' ?
2. Что означает XT' ?
3. Если корневых помеченных деревьев на n вершинах

$$t_n = n^{n-1},$$

сколько структур типа XT' на n метках?

Постарайся отдельно проговорить, **что уже было выделено в T** и **что дополнительно выделяет pointing XT'** .

10/3/2026, 3:36:58 PM · gpt-5-6-thinking



1. Т структура с одним специальным атомом $\backslash*$ (или её эквивалентное представление).
2. Формально: пара (атом, структура в 1), которую можно интерпретировать так: заткнули дырку атомом и, тем самым, выделили это место/атом.
3. Каждому дереву в Т соответствует n различных деревьев в XT' (n разных "фокусов"), то есть n^n . В Т выделен корень (как корень), в XT' дополнительно выделен один конкретный атом.

10/3/2026, 3:43:28 PM



Да, всё правильно.

Особенно важно последнее:

T = дерево с выделенным корнем

а

XT' = то же дерево + ещё одна выделенная вершина

причём эта дополнительная вершина вполне может совпасть с корнем.

Поэтому на каждом T -объекте размера n есть ровно n способов выбрать focus:

$$n \cdot n^{n-1} = n^n.$$

И формально ты тоже сказал точно: T' — это T -структура на обычных метках плюс специальный атом $*$; XT' возвращает вместо $*$ обычный атом и тем самым превращает дырку в выделенную вершину.

Задача 7. Одинаковая EGF — одинаковая структура?

Мы уже встретили

$$E = \text{SET}(\text{CYC}(T))$$

для endofunctions и получили

$$E(x) = \frac{1}{1 - T(x)}.$$

Но

$$\text{SEQ}(T)$$

имеет ту же EGF.

Ответь на два вопроса:

1. Что отсюда **можно** заключить про число объектов размера n ?
2. Чего отсюда **нельзя** заключить про сами species?

10/3/2026, 3:43:35 PM · gpt-5-6-thinking



1. Число объектов одно и то же. 2. Естественного изоморфизма между species может не существовать (хотя, это хороший намёк на наличие такового, и стоит проверить - а вдруг есть).

10/3/2026, 3:45:41 PM



Да, ровно так.

1. **Можно заключить**, что для каждого n число объектов одинаково:

$$|E[n]| = |\text{SEQ}(T)[n]|.$$

2. **Нельзя заключить**, что species изоморфны:

$$E \not\cong \text{SEQ}(T)$$

может оказаться вполне возможно, потому что species помнит ещё и действие перенумераций меток.

И твоя оговорка очень важна: одинаковая EGF — это **сильный намёк искать биекцию**, но не доказательство естественного изоморфизма.

Задача 8. Структура по EGF не восстанавливается

Попробуй теперь сформулировать своими словами разницу между тремя уровнями:

1. одинаковое число объектов каждого размера;
2. существует какая-то биекция между объектами каждого размера;
3. существует естественный изоморфизм species.

Что добавляется на каждом следующем уровне?

10/3/2026, 3:45:43 PM · gpt-5-6-thinking





3 => 2 => 1. Причём, равенства нет. Но, что конкретно добавляется сказать трудно, не повторяя формулировок 2 и 3. Твоё мнение?

10/3/2026, 3:48:17 PM



Да. И тут есть важная поправка к самой лестнице:

$$3 \Rightarrow 2 \Rightarrow 1$$

но если в пункте 2 понимать просто «**существует какая-то биекция**», то для конечных множеств 2 фактически не сильнее 1. Равные мощности уже гарантируют существование биекции.

Поэтому я бы переформулировал уровни так.

1. **Одинаковое перечисление.**

Для каждого n

$$|F[n]| = |G[n]|.$$

Мы знаем только числа.

2. **Явная комбинаторная биекция.**

Есть конкретный способ

$$\beta_n : F[n] \rightarrow G[n].$$

Это уже гораздо содержательнее: можно взять объект и реально преобразовать его в другой. Но биекция может использовать случайные особенности записи меток — например порядок

$$1 < 2 < \dots < n.$$

3. **Естественный изоморфизм species.**

Биекция не зависит от того, как метки названы. Для любой перенумерации

$$\sigma : U \rightarrow V$$

должен коммутировать квадрат:

$$G[\sigma] \circ \beta_U = \beta_V \circ F[\sigma].$$

То есть можно:

- сначала преобразовать структуру, потом переименовать метки;
- или сначала переименовать метки, потом преобразовать;

результат обязан быть один и тот же.

Вот это и есть конкретное добавление на уровне 3:

совместимость с перенумерацией меток.

Для нашего примера

$$\text{SET}(\text{CYC}(T)) \quad \text{и} \quad \text{SEQ}(T)$$

имеют одинаковую EGF. Можно построить явную биекцию, используя порядок меток: в цикле начать с максимальной метки и т. п. Это уровень 2.

Но если произвольно переименовать метки, «максимальная» вершина может стать другой. Значит преобразование не коммутирует с перенумерациями. Поэтому это не уровень 3.

Так что я бы запомнил иерархию именно так:



same counts < explicit bijection < natural bijection.

Причём первый переход — скорее от чисел к конструкции, а второй — от конструкции к конструкции, **не зависящей от имён объектов**.

10/3/2026, 3:48:28 PM · gpt-5-6-thinking



Ага, вот и теория категорий.

10/3/2026, 3:49:50 PM



Да — буквально. Здесь теория категорий появляется не как внешняя абстракция, а потому что мы уже естественным образом пришли к её языку:

- есть объекты: конечные множества меток;
- есть морфизмы: биекции/перенумерации;
- species F каждому множеству U сопоставляет множество структур $F[U]$;
- каждой биекции $\sigma : U \rightarrow V$ сопоставляет перенос структуры

$$F[\sigma] : F[U] \rightarrow F[V].$$

То есть species — это функтор

$$F : \mathbf{FinBij} \rightarrow \mathbf{Set}.$$

А естественный изоморфизм двух species — это как раз естественное преобразование, у которого все компоненты биективны.

Так что наш критерий

$$G[\sigma] \circ \beta_U = \beta_V \circ F[\sigma]$$

— это буквально условие естественности.

И мне нравится, что мы пришли к этому не через определения, а через вполне практический вопрос:

«Если две структуры имеют одинаковую EGF, что значит, что они действительно “одинаковы” как структуры?»

Ответ: не просто одинаковые числа, не просто произвольная биекция, а биекция, совместимая со всеми перенумерациями.

Вот здесь категория действительно возникает сама собой.

10/3/2026, 3:49:53 PM · gpt-5-6-thinking





Замечательно. Продолжаем? Листок 7 или всё ещё Листок 6?

10/3/2026, 3:52:21 PM



Всё ещё **Листок 6**: у нас осталась одна заключительная задача. После неё — Листок 7 про маркеры и частные производные.

Задача 9. Синтез

Для каждого выражения скажи, что оно означает **структурно**, не вычисляя производящую функцию:

$$\text{SET}(T), \quad \text{CYC}(T), \quad XT', \quad T'.$$

Где T — корневые помеченные деревья.

Особенно важно для последних двух различить:

- «дерево с дыркой»;
- «дерево с дополнительно выделенной вершиной».

10/3/2026, 3:52:26 PM · gpt-5-6-thinking





По порядку: лес, возможно пустой; цикл, где у каждой вершины своё дерево; дерево с дополнительно выделенной вершиной; дерево с дыркой (или дерево с одним специальным атомом *)

10/3/2026, 3:54:54 PM



Да, всё четыре правильно.

Чуть точнее второй пункт:

$$\text{CYC}(T)$$

— это **цикл корневых деревьев**: сами элементы цикла — T -объекты.

И важное различие в последних двух:

T' = дерево с одним специальным атомом *

а

XT' = дерево с одной дополнительно выделенной обычной вершиной.

На этом **Листок 6 закончен**.

Листок 7. Маркеры и частные производные

Цель: сделать активным правило

$$\frac{\partial}{\partial u}$$

как операцию «выделить одну из структур, которые считает маркер u ».

Задача 1

Пусть

$$F(x, u) = \sum_{n,k} a_{n,k} u^k \frac{x^n}{n!},$$

где u считает некоторое число компонент внутри объекта.

Что комбинаторно означает множитель k , который появляется после дифференцирования

$$\frac{\partial}{\partial u} u^k = k u^{k-1}?$$

Попробуй ответить без формул: что мы сделали с объектом, у которого было k таких компонент?

10/3/2026, 3:54:59 PM · gpt-5-6-thinking



То есть, в структуре у нас есть n атомов, которые разбиты на k компонент. Мы выделили одну из компонент: одна выделенная, $k-1$ невыделенных, k вариантов выделения (одна из компонент). Это комбинаторный смысл. Вопрос: а почему дифференцирование соответствует такому комбинаторному смыслу? Это же, кстати, и касается дифференцирования ряда по x : почему оно соответствует дырке где-то? Есть ли в этом глубокий смысл или, наоборот, мы говорим, что это дифференцирование структуры, только потому что формулы совпадают?

10/3/2026, 4:12:24 PM



Да — здесь есть **глубокий смысл**, и это не просто случайное совпадение формул. На самом деле одна из красот species именно в том, что обычное дифференцирование рядов является «тенью» структурной операции.

Можно посмотреть на это снизу вверх.

Для маркера u идея почти буквальная. Если у объекта k компонент, его вес содержит

$$u^k = u \cdot u \cdots u.$$

Дифференцируя, мы выбираем, **какой из k множителей u** был продифференцирован:

$$\frac{d}{du} u^k = \underbrace{u^{k-1} + \cdots + u^{k-1}}_{k \text{ раз}} = k u^{k-1}.$$

Комбинаторно:

выбрать одну из k компонент и сделать её специальной.

Причём u^{k-1} теперь считает только оставшиеся невыделенные компоненты.

То есть здесь производная сама по себе уже является оператором **«выбрать один множитель»**.

С x происходит то же самое, только x отмечает **атомы**.

Объект размера n несёт вес

$$x^n.$$

Формально

$$\frac{d}{dx} x^n = n x^{n-1}.$$

Почему n ?

Потому что можно выбрать один из n атомов:

выбрать атом

а x^{n-1} означает, что выбранный атом мы больше не считаем обычным атомом — он стал специальным $*$, то есть дыркой.

Для EGF это особенно красиво. Один объект размера n имеет вес

$$\frac{x^n}{n!}.$$

Дифференцируем:

$$\frac{d}{dx} \frac{x^n}{n!} = \frac{n x^{n-1}}{n!} = \frac{x^{n-1}}{(n-1)!}.$$

Здесь множитель n — это ровно число способов выбрать специальный атом, а факториалы автоматически перенормировались под оставшиеся $n-1$ обычных меток.

Это не случайность — EGF буквально **идеально согласована с операцией выделения атома**.

Но в теории species мы идём ещё дальше и вообще **определяем структурную производную независимо от рядов**:

$$F'[U] = F[U \sqcup \{*\}].$$

То есть:

F' -структура на обычных метках U — это F -структура, в которую добавлен один специальный атом $$.*

А затем доказываем:

$$\text{EGF}(F') = \frac{d}{dx} \text{EGF}(F).$$

То есть сначала существует **структурная операция**, а обычная производная ряда оказывается её числовым образом.

Это уже гораздо сильнее, чем «формулы совпали».

И правила дифференцирования тоже получают буквальный структурный смысл. Например,

$$(FG)' = F'G + FG'$$

означает:

дырка в паре структур находится либо в F -части, либо в G -части.

Цепное правило тоже означает: дырка проходит сначала во внешнюю структуру, затем в выбранную внутреннюю компоненту.

Мне нравится такой взгляд:

$$\text{species / структуры} \xrightarrow{\text{забываем структуру, считаем}} \text{производящие функции.}$$

При этом

$$\text{структурная производная} \longrightarrow \text{обычная производная ряда.}$$

На категориальном языке это можно назвать **декатегорификацией**: наверху у нас реальные структуры и биекции; внизу остаются только числа/ряды. Обычный calculus сохраняет след более богатой комбинаторной операции.

А для u ситуация чуть менее фундаментальная: если u — просто вспомогательный маркер статистики, то

$$\partial/\partial u$$

не является автоматически производной species по атомам. Но принцип тот же: **каждый фактор u соответствует одному отмечаемому компоненту**, поэтому производная выбирает один такой компонент.

Так что я бы удержал две формулы:

$$\frac{\partial}{\partial x} \approx \text{выделить атом и превратить его в } *$$

$$\frac{\partial}{\partial u} \approx \text{выделить одну из структур, которые считает } u.$$

И именно поэтому это дифференцирование, а не просто удачная терминология.

10/3/2026, 4:12:35 PM · gpt-5-6-thinking





Ощущение такое, что мы просто выделяем (выдаём флажок на помахать), но для атомов, выделенный атом мы условно считаем дыркой. Что там выделение, что здесь выделение. Единственно, почему член $a_{k,n} u^k x^n / n!$ несимметричен относительно u , x - это из-за $n!$ Неужели же дело в этом? Если нет, то не должно быть и разницы в подходах к u и x .

10/3/2026, 4:20:23 PM



Да — твоя интуиция здесь очень точная. **И по x , и по u мы в сущности делаем одно и то же: одну из учитываемых сущностей делаем специальной, “ставим на неё флажок”.**

А «дырка» — это просто удобный язык именно для специального атома.

Главное уточнение: надо различать $\partial/\partial x$ и **pointing** $x\partial/\partial x$.

Пусть

$$F(x, u) = \sum_{n,k} a_{n,k} u^k \frac{x^n}{n!}.$$

Тогда

$$\frac{\partial}{\partial u} u^k = k u^{k-1}.$$

Это означает: из k компонент одну выбрали и сделали **специальной**; она больше не учитывается множителем u , поэтому осталось u^{k-1} .

Совершенно аналогично

$$\frac{\partial}{\partial x} \frac{x^n}{n!} = \frac{x^{n-1}}{(n-1)!}.$$

Структурно: один из n обычных атомов превращён в специальный атом $*$, который больше не учитывается переменной x .

То есть параллель буквально такая:

$$\partial_x : \text{один обычный атом} \rightarrow \text{специальный атом } *$$

$$\partial_u : \text{одна обычная компонента} \rightarrow \text{специальная компонента.}$$

А если мы хотим **выделить, но продолжать считать её обычной**, то умножаем обратно на соответствующую переменную:

$$x\partial_x$$

— выделить один из n атомов, сохранив общий размер n ;

$$u\partial_u$$

— выделить одну из k компонент, сохранив общее число компонент k .

Вот здесь симметрия почти полная.

Почему же запись несимметрична:

$$u^k \frac{x^n}{n!} ?$$

Да: **по существу из-за того, что x и u играют разные роли.**

x считает **помеченные атомы**, поэтому используется экспоненциальная нормировка

$$\frac{x^n}{n!}.$$

u пока всего лишь считает некоторую статистику — количество блоков, циклов и т. п. Сами блоки не получают независимых меток $1, \dots, k$, поэтому мы пишем просто

$$u^k,$$

а не

$$\frac{u^k}{k!}.$$

Если бы мы ввели **второй сорт помеченных атомов** и считали его через

$$\frac{u^k}{k!},$$

то x и u стали бы совершенно симметричными. Это уже теория **многосортовых species**.

Так что твоя догадка:

«Неужели всё дело в $n!$?»

Почти да, но я бы сформулировал чуть глубже:

$n!$ — след того, что x считает настоящий сорт помеченных атомов,

тогда как

u пока лишь обычный маркер некоторой статистики структуры.

Очень хороший пример — наши разбиения:

$$\Pi(x, u) = e^{u(e^x - 1)}.$$

Здесь

$$\partial_u \Pi$$

означает «один блок сделать специальным», а

$$\partial_x \Pi$$

— «один атом сделать специальным».

Это действительно **одна и та же идея “выделения” на двух разных структурных уровнях**.

И я думаю, это гораздо лучший способ понимать частные производные, чем мой прежний контраст «по x дырка, по u выделение». Он был слишком резким. Дырка — просто специальный атом, который мы перестали считать среди обычных.

10/3/2026, 4:20:33 PM · gpt-5-6-thinking



"выделить, но продолжать считать её обычной" - неудачная формулировка, мне кажется. "выделить, но продолжать её считать" (!!)

10/3/2026, 4:24:20 PM



Да, согласен. Лучше именно так:

$$\partial_x$$

— выделить один атом и перестать учитывать его переменной x ;

$$x\partial_x$$

— выделить один атом, но продолжать его считать в размере.

И аналогично:

$$\partial_u$$

— выделить одну компоненту и перестать учитывать её множителем u ;

$$u\partial_u$$

— выделить одну компоненту, но продолжать её считать среди компонент.

Это намного точнее, чем «считать её обычной». Выделенная сущность остаётся специальной; мы лишь решаем, входит ли она по-прежнему в соответствующую статистику.

10/3/2026, 4:24:22 PM · gpt-5-6-thinking





Да! Продолжаем?

10/3/2026, 4:24:48 PM



Да. Продолжаем Листок 7.

Задача 2. Разбиения с маркером числа блоков

У нас

$$\Pi(x, u) = \exp(u(e^x - 1)).$$

Рассмотрим отдельно два оператора:

$$\frac{\partial}{\partial u} \quad \text{и} \quad u \frac{\partial}{\partial u}.$$

Объясни комбинаторно, что считает каждый из них.

Подсказка только терминологическая: u считает **число блоков** разбиения.

10/3/2026, 4:24:55 PM · gpt-5-6-thinking





Только что обсудили [если я правильно понял вопрос]: один блок выделяем и его не считаем с остальными. И, один блок выделяем, и его продолжаем считать с остальными.

10/3/2026, 4:26:48 PM



Да, именно так.

Для разбиения с k блоками:

$$\frac{\partial}{\partial u}$$

даёт ku^{k-1} : один блок выделен, остальные $k - 1$ продолжают учитываться маркером u .

А

$$u \frac{\partial}{\partial u}$$

даёт ku^k : один блок выделен, но общее число блоков по-прежнему считается как k .

Следующая задача будет чуть менее тавтологичной.

Задача 3

Для тех же разбиений сравни теперь:

$$\frac{\partial \Pi}{\partial x} \quad \text{и} \quad x \frac{\partial \Pi}{\partial x}.$$

Что комбинаторно делает каждый оператор, если x считает атомы?

10/3/2026, 4:26:51 PM · gpt-5-6-thinking



А есть разница между этой задачей и предыдущей? Один атом выделяем (заменяем дыркой/*) и этот атом в размере не считаем. И, считаем для пар (атом, структура с дыркой), то есть для структур, в которых один атом выделен (носит флажок, но входит в счёт) и тем самым отличает эту структуру от подобных ей.

10/3/2026, 4:31:33 PM



Да — по существу это **та же самая схема**, только u считает блоки, а x — атомы.

Твоя формулировка точная:

$$\frac{\partial \Pi}{\partial x}$$

— один атом выделили, превратили в специальный *, и **в размере по x его больше не считаем**.

А

$$x \frac{\partial \Pi}{\partial x}$$

— один атом выделили, но **размер по-прежнему считает его**. Это pointing:

структура с одним выделенным атомом.

Так что действительно:

$\partial_u \leftrightarrow$ выделить блок и не считать его по u

$u \partial_u \leftrightarrow$ выделить блок и продолжать считать его

и совершенно аналогично для x .

Задача 4

Теперь чуть интереснее. Что комбинаторно означают:

$$\frac{\partial^2 \Pi}{\partial u^2}$$

и

$$u^2 \frac{\partial^2 \Pi}{\partial u^2} ?$$

Важно: выделяются ли два блока упорядоченно или неупорядоченно?

10/3/2026, 4:31:38 PM · gpt-5-6-thinking



Я бы сказал неупорядочено. Выделяем два блока (красный флажок и синий флажок, то есть разные флажки?) и их не считаем. И, выделяем два блока (разные флажки?) и их считаем. Вопрос: разные ли флажки? я не уверен.

10/3/2026, 4:35:02 PM



Да — **флажки разные**. Это как раз важная тонкость.

Если у объекта k блоков, то

$$\frac{\partial^2}{\partial u^2} u^k = k(k-1)u^{k-2}.$$

Число

$$k(k-1)$$

считает **упорядоченный выбор двух различных блоков**:

- сначала выбираем блок с первым флажком;
- потом другой блок со вторым флажком.

То есть условно:

красный флажок + синий флажок

и перестановка цветов даёт другой выбор.

Поэтому

$$\frac{\partial^2 \Pi}{\partial u^2}$$

означает: два различных блока выделены **различимыми флажками**, и оба перестали учитываться степенью u .

А

$$u^2 \frac{\partial^2 \Pi}{\partial u^2}$$

— те же два различимо выделенных блока, но оба продолжают учитываться в числе блоков.

Если же мы хотим выбрать просто **неупорядоченную пару блоков**, одинаково выделенных, надо поделить на $2!$:

$$\frac{1}{2!} \frac{\partial^2 \Pi}{\partial u^2}$$

потому что каждая неупорядоченная пара была посчитана дважды.

То есть:

$$\partial_u^2 \leftrightarrow \text{два различных флажка}$$

а

$$\frac{1}{2!} \partial_u^2 \leftrightarrow \text{два одинаковых флажка.}$$

Это очень хороший принцип, и дальше он будет повторяться.

10/3/2026, 4:35:07 PM · gpt-5-6-thinking





Да, верно, $k(k-1)$ объясняет упорядоченность пары. Иначе бы было $C(k, 2) = k(k-1)/2$.

10/3/2026, 4:37:44 PM



Именно.

$$\partial_u^2$$

даёт **упорядоченную пару различных блоков**:

$$k(k-1).$$

А

$$\frac{1}{2!} \partial_u^2$$

даёт **неупорядоченную пару**:

$$\binom{k}{2}.$$

Обобщение сразу видно:

$$\frac{\partial^r}{\partial u^r}$$

выделяет r различных компонент с r различными флажками, а

$$\boxed{\frac{1}{r!} \frac{\partial^r}{\partial u^r}}$$

— выбирает неупорядоченное множество из r выделенных компонент.

Задача 5

Теперь пусть u считает число циклов в перестановке.

Если

$$P(x, u)$$

— соответствующая двухпеременная EGF, что должны комбинаторно означать:

$$u \frac{\partial P}{\partial u}$$

и

$$\frac{1}{2} u^2 \frac{\partial^2 P}{\partial u^2} ?$$

Пока не нужно знать явную формулу для $P(x, u)$.

10/3/2026, 4:37:46 PM · gpt-5-6-thinking





Выделяем один цикл, но считаем всё равно. И, выделяем два цикла, но неупорядоченно, и считаем всё равно. Мне нужен небольшой конкретный пример, что при этом меняется при счёте.

10/3/2026, 4:41:29 PM



Да. Возьмём совсем маленький пример: перестановки трёх элементов.

По числу циклов они распределяются так:

$$123 = (1)(2)(3)$$

— одна перестановка с 3 циклами;

три перестановки типа

$$(12)(3)$$

— с 2 циклами;

две перестановки типа

$$(123)$$

— с 1 циклом.

Поэтому, если пока отбросить общий множитель $x^3/3!$, соответствующий полином по u равен

$$u^3 + 3u^2 + 2u.$$

Теперь применим

$$u \frac{\partial}{\partial u}.$$

Получаем

$$3u^3 + 6u^2 + 2u.$$

Что произошло комбинаторно?

- у $(1)(2)(3)$ было 3 возможности выбрать выделенный цикл $\rightarrow$ вместо одного объекта получаем **3**;
- каждая из трёх перестановок с двумя циклами даёт по 2 выбора $\rightarrow 3 \cdot 2 = 6$;
- каждая 3-цикл-перестановка имеет только один цикл $\rightarrow 2 \cdot 1 = 2$.

То есть оператор

$$u \partial_u$$

буквально **размножает каждый объект с k циклами в k объектов — по одному на каждый выбор выделенного цикла.**

Причём степень u^k осталась прежней: мы цикл выделили, но продолжаем считать его среди k циклов.

Теперь два неупорядоченно выделенных цикла:

$$\frac{1}{2} u^2 \frac{\partial^2}{\partial u^2}.$$

Из

$$u^3 + 3u^2 + 2u$$

получаем



$$3u^3 + 3u^2.$$

И снова буквально:

- (1)(2)(3): выбрать 2 из 3 циклов можно

$$\binom{3}{2} = 3$$

способами;

- каждая перестановка с двумя циклами: выбрать оба можно одним способом; таких перестановок $3 \rightarrow$ ещё 3;
- перестановка с одним циклом не даёт ни одного варианта.

Итого 6 структур с двумя выделенными циклами.

Так что практический эффект очень простой:

исходный объект с k компонентами

превращается в

k объектов

при одном выделении, или в

$\binom{k}{2}$ объектов

при неупорядоченном выделении двух.

Вот что реально меняется при счёте: **одна структура начинает учитываться столько раз, сколько существует способов поставить требуемые флажки.**

10/3/2026, 4:41:38 PM · gpt-5-6-thinking



Ага, теперь понятно. Ассоциация: а нельзя ли как нибудь из этого получить теорию Burnside-Pólya? Мы, в общем, тоже считаем.

10/3/2026, 4:47:12 PM



Да — и не просто «можно»: **это один из естественных выходов из species прямо в Burnside-Pólya.**

До сих пор EGF помнит только число помеченных структур:

$$|F[n]|.$$

Но species F знает гораздо больше: группа перестановок меток S_n действует на $F[n]$. Для каждой перестановки σ можно спросить:

сколько F -структур остаётся неподвижными при σ ?

А это уже буквально территория Бернсайда.

Всю эту информацию собирают в **cycle index series** species:

$$Z_F(p_1, p_2, \dots) = \sum_{n \geq 0} \frac{1}{n!} \sum_{\sigma \in S_n} \text{fix}_F(\sigma) p_1^{c_1(\sigma)} p_2^{c_2(\sigma)} \dots,$$

где $c_k(\sigma)$ — число k -циклов перестановки σ .

И теперь чудо: и EGF, и OGF получаются из **одного и того же** Z_F .

Для помеченного счёта:

$$F_{\text{EGF}}(x) = Z_F(x, 0, 0, \dots).$$

Почему? Потому что выживает только тождественная перестановка, состоящая из одних 1-циклов.

А для непомеченных форм:

$$\tilde{F}(x) = Z_F(x, x^2, x^3, \dots).$$

Это уже прямо Burnside: мы усредняем число фиксированных структур по всем перестановкам, то есть считаем орбиты под действием S_n .

И наш любимый **SET** даёт идеальный пример. Его cycle index:

$$Z_{\text{SET}} = \exp\left(p_1 + \frac{p_2}{2} + \frac{p_3}{3} + \dots\right).$$

Теперь:

помеченный счёт:

$$p_1 = x, \quad p_{k>1} = 0$$

даёт

$$e^x.$$

А непомеченный:

$$p_k = x^k$$

даёт

$$\exp\left(x + \frac{x^2}{2} + \frac{x^3}{3} + \dots\right) = \exp(-\log(1-x)) = \frac{1}{1-x}.$$

То есть наши две функции

$$e^x \quad \text{и} \quad \frac{1}{1-x}$$

для SET — это две разные проекции **одного более богатого объекта**, cycle index.

И это прекрасно объясняет то, о чём мы недавно говорили:

$$\text{species} \rightarrow \text{действие } S_n \rightarrow \text{cycle index} \rightarrow \begin{cases} \text{EGF} \text{ — labeled} \\ \text{OGF} \text{ — unlabeled} \end{cases}$$

А **Pólya enumeration** появляется, когда вместо просто структур мы ещё раскрашиваем атомы и спрашиваем число раскрасок с точностью до симметрий. Тогда в cycle index подставляются generating functions цветов — это и есть знаменитая Pólya substitution.

Так что связь очень тесная:

$$\text{Burnside} \subset \text{cycle-index machinery of species}$$

а Pólya — следующий уровень, где фиксированные точки группового действия превращаются в систематическое перечисление раскрасок и составных структур.

И, по-моему, это отличный будущий листок: начать с SET, где мы уже знаем ответы e^x и $1/(1-x)$, и **вывести оба из одного cycle index через Burnside**. Это как раз свяжет твои нынешние вопросы об автоморфизмах, OGF/EGF и действиях групп в одну картину.

10/3/2026, 4:47:21 PM · gpt-5-6-thinking



Да, я как раз хотел попросить добавить листок в конце на эту тему. Хорошо, продолжаем.

10/3/2026, 4:49:42 PM



Да. Burnside-Pólya добавляем отдельным поздним листком.

Продолжаем **Листок 7**.

Задача 6. Перестановки с маркером числа циклов

Перестановку можно представить как множество циклов:

$$\text{PERM} = \text{SET}(\text{CYC}(X)).$$

Теперь введём маркер u , который считает число циклов.

Попробуй получить двухпеременную EGF $P(x, u)$.

Подсказка: один цикл даёт

$$-\log(1 - x),$$

а каждый цикл должен ещё получить вес u .

10/3/2026, 4:49:53 PM · gpt-5-6-thinking





$$\exp(-\log(1-x)) = 1/(1-x) = \sum (x^n), \text{ а нам надо } \sum (u^k x^n), k, n \geq 0$$

10/3/2026, 4:58:33 PM



Да — ты почти дошёл, но u надо вставить **до внешней экспоненты**, потому что вес u получает каждый цикл.

Один цикл:

$$\text{CYC}(X) \longleftrightarrow -\log(1-x).$$

Если каждый цикл получает дополнительный вес u , то:

$$u[-\log(1-x)].$$

А перестановка — SET таких циклов, поэтому

$$P(x, u) = \exp(-u \log(1-x)) = (1-x)^{-u}.$$

Твоя запись

$$\sum_{n, k \geq 0} u^k x^n$$

означала бы, что для каждой пары (n, k) существует ровно один объект. Но перестановок n элементов с k циклами вообще говоря много.

Правильно:

$$P(x, u) = \sum_{n, k \geq 0} c(n, k) u^k \frac{x^n}{n!},$$

где $c(n, k)$ — **беззнаковые числа Стирлинга первого рода**: число перестановок n элементов с k циклами.

Например, при $n = 3$:

$$2u + 3u^2 + u^3,$$

потому что есть:

2 перестановки с 1 циклом, 3 с 2 циклами, 1 с 3 циклами.

И при $u = 1$:

$$P(x, 1) = \frac{1}{1-x},$$

что соответствует $n!$ перестановкам:

$$n! \frac{x^n}{n!} = x^n.$$

Задача 7

Теперь найди

$$u \frac{\partial P}{\partial u}$$

для

$$P(x, u) = (1-x)^{-u},$$

и объясни **двумя способами**, что она считает:

1. непосредственно через «выделить один цикл, но продолжать его считать»;

2. по получившейся аналитической формуле.

10/3/2026, 4:58:39 PM · gpt-5-6-thinking



$-u \times \ln(1-x) \times P$, считает количество перестановок на n элементов, причём каждый цикл имеет вес u , причём учитывается что один цикл выделен - то есть, для k циклов количество дополнительно умножается на k . Вопрос: как это получить из формулы для производной? Не видно.

10/3/2026, 5:10:08 PM



Да — из закрытой формулы это действительно не видно, пока не прочитать множители комбинаторно. Есть два способа увидеть.

Во-первых, из самого ряда:

$$P(x, u) = \sum_{n, k} c(n, k) u^k \frac{x^n}{n!}.$$

Тогда

$$u \frac{\partial P}{\partial u} = \sum_{n, k} k c(n, k) u^k \frac{x^n}{n!}.$$

Вот здесь смысл совершенно прозрачен: каждую перестановку с k циклами мы считаем k раз — по числу способов выбрать один выделенный цикл.

А теперь закрытая формула:

$$P = (1 - x)^{-u}.$$

Получаем

$$u P_u = -u \log(1 - x) P.$$

Разберём множители:

$$-\log(1 - x)$$

— EGF **одного цикла**;

$$u[-\log(1 - x)]$$

— один цикл, причём он получает вес u , то есть **мы продолжаем учитывать его среди циклов**;

$$P$$

— произвольная перестановка на остальных метках, то есть множество остальных циклов.

Поэтому

$$u P_u = \underbrace{u \text{ CYC}(X)}_{\text{один выделенный цикл}} \cdot \underbrace{\text{PERM}}_{\text{все остальные циклы}}$$

и это буквально означает:

$$\boxed{\text{перестановка с одним выделенным циклом.}}$$

То есть закрытая аналитическая формула

$$-u \log(1 - x) P$$

на самом деле очень содержательна: производная «вытащила» из **SET** один элемент — один цикл — а остальные остались обычным **SET** циклов.

Это тот же общий шаблон:



$$\frac{d}{du} \exp(uA) = A \exp(uA)$$

комбинаторно значит:

из множества A -компонент выбрать одну особую A -компоненту, остальные оставить множеством.

A дополнительный множитель u возвращает выделенную компоненту в статистику числа компонент.

10/3/2026, 5:10:14 PM · gpt-5-6-thinking



Если так посмотреть, то понятно. Хорошо, продолжим.

10/3/2026, 5:13:33 PM



Хорошо. Тогда следующая задача — уже с содержательным результатом.

Задача 8. Сколько всего циклов во всех перестановках?

У нас

$$P(x, u) = (1 - x)^{-u},$$

и

$$u \frac{\partial P}{\partial u} = -u \log(1 - x) P.$$

Теперь положи

$$u = 1.$$

Что получится как EGF для **общего числа выделенных циклов во всех перестановках n элементов?**

А затем попробуй извлечь коэффициент при $x^n/n!$. Подсказка:

$$-\log(1 - x) = \sum_{m \geq 1} \frac{x^m}{m}, \quad \frac{1}{1 - x} = \sum_{r \geq 0} x^r.$$

Должна появиться знакомая сумма

$$1 + \frac{1}{2} + \cdots + \frac{1}{n}.$$

10/3/2026, 5:13:37 PM · gpt-5-6-thinking





Да. Перемножаем $\sum (x^n/n)$ и $\sum (x^n/r)$. Коэффициент при x^n это сумма коэффициентов при $x^{1/1} \times x^{(n-1)} + x^{2/2} \times x^{(n-2)} + \dots + x^{n/n} \times x^0$, то есть $1 + 1/2 + \dots + 1/n = H_n$. Этот коэффициент включает деление на $n!$. То есть, $a_n = H_n \times n!$. Это соответствует общему количеству всех циклов для перестановок n элементов.

10/3/2026, 8:11:55 PM



Да, именно.

$$u \frac{\partial P}{\partial u} \Big|_{u=1} = \frac{-\log(1-x)}{1-x} = \sum_{n \geq 1} H_n x^n.$$

Но это EGF, значит

$$\boxed{\text{общее число циклов во всех перестановках } S_n = n! H_n.}$$

И отсюда сразу следует ещё один приятный факт: среднее число циклов в случайной перестановке n элементов равно

$$\boxed{H_n}.$$

Это хороший пример того, как маркер u и производная по нему дают не просто формальное тождество, а реальную статистику класса объектов.

Листок 7 почти закончен. Я бы дал ещё одну последнюю задачу на второй момент:

если взять

$$u^2 \frac{\partial^2 P}{\partial u^2},$$

то что он считает комбинаторно, и какую статистику по числу циклов можно из него получить после подстановки $u = 1$?

10/3/2026, 8:11:59 PM · gpt-5-6-thinking





Комбинаторно мы фиксируем два цикла (если они есть), то есть после подстановки мы получим при x^n не общую сумму циклов, а общую сумму $k(k-1)$. В принципе, поскольку мы знаем среднее, это новое можно использовать для вычисления дисперсии.

10/3/2026, 8:17:08 PM



Да, именно. И здесь появляется очень полезное понятие: **факториальные моменты**.

Если K — число циклов случайной перестановки, то

$$u^2 \frac{\partial^2 P}{\partial u^2} \Big|_{u=1}$$

считает суммарно

$$K(K-1)$$

по всем перестановкам. Для

$$P(x, u) = (1-x)^{-u}$$

имеем

$$\frac{\partial^2 P}{\partial u^2} = [-\log(1-x)]^2 P,$$

поэтому при $u = 1$

$$u^2 P_{uu} \Big|_{u=1} = \frac{[-\log(1-x)]^2}{1-x}.$$

Коэффициент при x^n оказывается

$$H_n^2 - H_n^{(2)},$$

где

$$H_n^{(2)} = \sum_{j=1}^n \frac{1}{j^2}.$$

Следовательно

$$E[K(K-1)] = H_n^2 - H_n^{(2)}.$$

А поскольку

$$K^2 = K(K-1) + K,$$

то

$$E[K^2] = H_n^2 - H_n^{(2)} + H_n.$$

И наконец

$$\text{Var}(K) = H_n - H_n^{(2)}.$$

То есть твоя мысль абсолютно правильная: первая производная по маркеру дала среднее, вторая — второй факториальный момент, и вместе они дают дисперсию.

Причём теперь виден общий принцип:

$$(u \partial_u)^2$$

нужно использовать осторожно, а именно

$$u^r \frac{\partial^r}{\partial u^r}$$

естественно даёт падающий факториал

$$K^r = K(K-1) \cdots (K-r+1).$$

Это очень удобно именно потому, что комбинаторно означает: **выбрать r различных компонент с различными флажками**.

На этом я бы **Листок 7 завершил**. По-моему, здесь произошёл настоящий переход от пассивного к активному пониманию маркеров: u -производная теперь уже не выглядит чужой операцией.

Следующий по нашему плану — **Листок 8: composition/substitution, Bell polynomials и дорога к Faà di Bruno**.

10/3/2026, 8:17:18 PM · gpt-5-6-thinking



Хорошо. Эту тему мы ранее [практически] не разбирали и она мне не очень знакома, может перед листком сделать краткую обзорную лекцию, или включить пояснения в задачи?

10/3/2026, 8:20:34 PM



Я бы сделал **гибрид**: сначала очень короткая обзорная лекция на 5-10 минут, чтобы появилась карта местности, а затем листок, где новые детали будут вводиться ровно тогда, когда понадобятся. Для совсем новой темы это лучше, чем бросать тебя сразу в задачи.

Главная карта такая.

У нас уже были операции над species:

$$F + G, \quad FG, \quad \text{SET}(F), \quad \text{SEQ}(F), \quad \text{CYC}(F).$$

Теперь появляется ещё одна:

$$F \circ G$$

- **подстановка / композиция species.**

Интуитивно: берём F -структуру, но вместо каждого её «атома» помещаем целую G -структуру.

Однако в labeled species это надо сформулировать аккуратнее. Множество меток S сначала разбивается на блоки:

$$S = B_1 \sqcup \dots \sqcup B_k.$$

На каждом блоке B_i строится G -структура, а затем **сами блоки**, уже рассматриваемые как k объектов, организуются в F -структуру.

То есть:

$$F \circ G = F\text{-структура на множестве } G\text{-компонент.}$$

И наш старый пример уже был именно таким:

$$\Pi = \text{SET}(\text{SET}_{\geq 1}(X)).$$

Разбиение множества - это:

1. разбить метки на непустые блоки;
2. каждый блок есть $\text{SET}_{\geq 1}(X)$;
3. сами блоки образуют неупорядоченное множество SET .

Отсюда EGF:

$$\Pi(x) = \exp(e^x - 1).$$

Это не случайная алгебра:

$$\text{SET} \circ \text{SET}_{\geq 1} \longrightarrow \exp(e^x - 1).$$

То есть **композиции species соответствует композиция EGF**:

$$(F \circ G)(x) = F(G(x)),$$

при обычном условии $G(0) = 0$: внутренняя структура не должна быть пустой, иначе один внешний атом можно было бы заменять пустотой бесконечно многими способами.

Теперь откуда возникнут **Bell polynomials**.

Пусть мы хотим узнать коэффициент в

$$F(G(x)).$$

Например,

$$F(y) = \sum_k f_k \frac{y^k}{k!}, \quad G(x) = \sum_{j \geq 1} g_j \frac{x^j}{j!}.$$

Если внешняя F -структура имеет k компонентов, надо разбить n меток на k блоков разных размеров и на каждом построить G -структуру.

Вот эта задача

«всеми способами разбить n меток на k блоков размеров $1, 2, \dots$, с весом g_j для блока размера j »

и кодируется **частичными полиномами Белла**

$$B_{n,k}(g_1, g_2, \dots).$$

Поэтому схематически

$$[x^n/n!] F(G(x)) = \sum_k f_k B_{n,k}(g_1, g_2, \dots).$$

То есть Bell polynomials - не страшная новая сущность. Это просто **машина для учёта всех способов собрать n атомов из k внутренних компонент различных размеров.**

И отсюда почти автоматически возникает **Faà di Bruno**.

Мы уже знаем обычное правило цепочки:

$$(F \circ G)' = (F' \circ G) G'.$$

Если дифференцировать много раз, дырки могут распределяться по внутренним G -компонентам множеством способов:

- все попасть в одну;
- распределиться $2 + 1$;
- $1 + 1 + 1$;
- и т. д.

То есть снова появляются **разбиения множества производных на блоки**.

Например:

$$\frac{d^2}{dx^2} F(G) = F''(G)(G')^2 + F'(G)G'',$$

а для третьей:

$$F'''(G)(G')^3 + 3F''(G)G'G'' + F'(G)G'''.$$

Коэффициент 3 во втором слагаемом - уже маленький Bell-polynomial phenomenon: три способа выбрать, какие две из трёх дифференциаций попали вместе.

Поэтому большая формула Faà di Bruno - это в сущности:

$$\text{многократное правило цепочки} = \text{разбиения множества} = \text{Bell polynomials}.$$

И это очень хорошо ложится на species: не надо сначала запоминать ужасную формулу, а потом пытаться понять её. Мы попробуем **получить её из комбинаторики разбиений**.

Я бы построил Листок 8 примерно из 9 задач: сначала две задачи на саму подстановку $F \circ G$, затем разбиения и $\Pi = \text{SET} \circ \text{SET}_{\geq 1}$, затем коэффициенты и первые Bell polynomials, и только в конце - 2-я и 3-я производные композиции как

подготовка к Faà di Bruno. Большую общую формулу сначала вообще не будем давать.

10/3/2026, 8:20:39 PM · gpt-5-6-thinking



Ты это говоришь уже не в первый раз: "при обычном условии $(G(0)=0)$: внутренняя структура не должна быть пустой, иначе один внешний атом можно было бы заменять пустотой бесконечно многими способами.", но я так и не понял. Можно привести какой-нибудь простой пример или объяснить иначе?

10/3/2026, 8:29:51 PM



Да — лучше всего увидеть это на совсем конкретном контрпримере.

Возьмём

$$F = \text{SEQ}, \quad G = 1 + X.$$

$G = 1 + X$ означает: внутренняя структура может быть либо пустой, либо состоять из одного атома. То есть

$$G(0) = 1 \neq 0.$$

Теперь попробуем построить

$$\text{SEQ}(G).$$

Пусть у нас всего один настоящий атом a . Если пустые G -структуры разрешены, то один и тот же атом можно окружать сколько угодно пустыми компонентами:

$$\begin{aligned} &[a], \\ &[\emptyset, a], \quad [a, \emptyset], \\ &[\emptyset, \emptyset, a], \quad [\emptyset, a, \emptyset], \quad [a, \emptyset, \emptyset], \end{aligned}$$

и так далее.

Все эти структуры имеют **размер 1**, потому что пустые компоненты атомов не содержат.

Получается:

уже структур размера 1 бесконечно много.

Вот в чём проблема.

То же самое видно алгебраически. Для **SEQ**

$$F(y) = \frac{1}{1-y} = \sum_{k \geq 0} y^k.$$

Подставляем

$$\begin{aligned} G(x) &= 1 + x : \\ F(G(x)) &= \sum_{k \geq 0} (1+x)^k. \end{aligned}$$

Посмотрим хотя бы на коэффициент при x . В $(1+x)^k$ он равен k , поэтому получаем

$$0 + 1 + 2 + 3 + \dots,$$

то есть коэффициент вообще не является конечным числом.

А если

$$G(0) = 0,$$

то каждая внутренняя G -компонента обязана содержать хотя бы один атом. Поэтому на n атомах таких компонент может быть максимум n .

Например, если

$$G = X + X^2 + \dots,$$

то структура размера n не может содержать $n + 1$ внутренних G -компонент: для каждой нужен хотя бы один атом. Поэтому условие

$$G(0) = 0$$

можно читать очень просто:

каждая внутренняя компонента должна что-нибудь “съесть” из имеющихся атомов.

Тогда при фиксированном размере n нельзя незаметно добавить сколько угодно пустых компонент.

И, пожалуй, моя прежняя формулировка «один внешний атом можно заменять пустотой бесконечно многими способами» была неудачной. Точнее сказать так:

если внутренняя компонента может быть пустой, то можно добавлять сколько угодно таких компонент, не меняя размер объе

Именно это ломает обычную композицию производящих функций.

10/3/2026, 8:30:02 PM · gpt-5-6-thinking





Понятно. Хорошо, можно продолжать.

10/3/2026, 8:33:38 PM



Тогда начинаем **Листок 8: композиция species**. Будем идти постепенно, с пояснениями по мере надобности.

Задача 1

Рассмотрим

$$F = \text{SEQ}, \quad G = \text{SET}_{\geq 1}(X).$$

То есть

$$F \circ G = \text{SEQ}(\text{SET}_{\geq 1}(X)).$$

Пусть множество меток

$$S = \{1, 2, 3\}.$$

Вопрос: что представляет собой одна такая структура на S ?

Попробуй сначала описать словами, а затем перечислить или хотя бы посчитать все структуры на $\{1, 2, 3\}$.

Подсказка: сначала надо разбить S на непустые блоки, а потом вспомнить, что внешний **SEQ** делает с этими блоками.

10/3/2026, 8:33:49 PM · gpt-5-6-thinking



У меня есть довольно фундаментальное непонимание (надо все мои текущие непонимания как-то собрать вместе): возьмем список (SEQ) как тип данных. Есть только одна форма для каждого размера (n-ка, кортеж длины n). Если мы помечаем элементы, у нас есть n! вариантов для размера n - все перестановки. Дубликаты не допускаются. У меня есть подозрение, что дубликаты не допускаются только потому, что у нас такой тип данных - без дубликатов (что явно не проговаривалось) и, в принципе, можно завести тип "список с дубликатами" и считать по другому. [Или я чего-то не понимаю и отсутствие дубликатов принципиальный момент (?)]. Далее, сейчас у нас вместо нумерованных атомов есть блоки, и отсутствие дубликатов становится не таким очевидным. Два блока с одинаковым содержимым - это дубликаты (пояснение: с точки зрения практически любого языка программирования, помеченный атом - это синглтон, но два объекта с одинаковым содержимым - это два разных объекта.)

10/4/2026, 10:39:24 AM



Да — здесь действительно смешались **два разных понятия: атом как объект с идентичностью и значение, которое он хранит**. Это фундаментальный узел.

В species отсутствие «дубликатов» у SEQ не является специальным свойством типа SEQ. Причина глубже:

species строит структуру НА данном множестве меток U .

Если

$$U = \{1, 2, 3\},$$

то $SEQ[U]$ — это все линейные порядки **этих трёх элементов**, причём каждый элемент множества U используется ровно один раз:

$$(1, 2, 3), (1, 3, 2), \dots$$

Отсюда 3!.

Нельзя получить

$$(1, 1, 2),$$

потому что это уже не структура на множестве $\{1, 2, 3\}$: метка 1 использована дважды, а 3 вообще исчезла.

То есть species здесь ближе не к обычному программному `List<T>`, куда можно несколько раз положить одно и то же значение, а к:

«возьми данное конечное множество объектов и организуй **все его элементы** в такую-то структуру».

Это принципиальная часть определения species, а не особенность SEQ.

Теперь к твоей программистской интуиции. Она совершенно уместна, но надо разделить:

identity $\neq$ value.

Допустим, есть три различных атома

$$a, b, c,$$

но у a и b одинаковое значение `"red"`.

Тогда в списке вполне может быть

$$(a, b, c),$$

и по значениям это выглядит как

(red, red, blue).

Повторяющиеся значения разрешены. Просто a и b остаются двумя различными атомами.

В обычной species-теории метки 1, 2, 3 играют роль именно **identity**, а не payload.

Если мы хотим дополнительно присваивать им значения/цвета, это ещё один слой структуры.

Теперь композиция — здесь твой вопрос особенно важен.

Пусть

$$F = \text{SEQ}, \quad G = \text{SET}_{\geq 1}(X).$$

Для $U = \{1, 2, 3\}$ сначала берётся **разбиение множества** U на блоки.

Например:

$$\{\{1, 3\}, \{2\}\}.$$

Блоки сами являются элементами нового множества:

$$\pi = \{ \{1, 3\}, \{2\} \}.$$

А внешний **SEQ** задаёт порядок на **этом множестве блоков**:

$$(\{1, 3\}, \{2\})$$

или

$$(\{2\}, \{1, 3\}).$$

Теперь твой вопрос:

могут ли быть два блока с одинаковым содержимым?

Нет — но **не потому что** **SEQ** **запрещает дубликаты**.

Потому что блоки являются блоками **разбиения одного множества**. Они должны быть непусты, попарно не пересекаться и вместе давать U .

Поэтому нельзя иметь

$$\{\{1, 2\}, \{1, 2\}\},$$

потому что атомы 1, 2 использовались бы дважды.

Но совершенно можно иметь два блока **одинакового типа/формы**, например

$$\{\{1, 2\}, \{3, 4\}\}.$$

Оба блока имеют размер 2, и на каждом из них **SET₂**-структура выглядит одинаково с точностью до переименования меток. Однако это два разных объекта:

$$\{1, 2\} \neq \{3, 4\}.$$

Вот тут твоя аналогия с языками программирования очень хороша:

```
object A = {1, 2}
object B = {3, 4}
```

Они могут иметь одинаковую форму и даже одинаковые payload-значения, но это разные компоненты.

Поэтому я бы сформулировал главный принцип так:

В species атомы не выбираются “с возвратом”.

Мы начинаем с фиксированного множества атомов U , и структура организует каждый из них ровно один раз.

А «дубликаты» в программистском смысле относятся уже не к меткам, а к **значениям, которыми атомы могут быть дополнительно украшены**.

Именно это, мне кажется, было скрытым местом, которое раньше стоило проговорить явно.

Для нашей исходной задачи это значит, что `SEQ(SET_{\geq 1}(X))` — это просто **упорядоченное разбиение множества**: разбили $\{1, 2, 3\}$ на блоки и затем упорядочили блоки.

10/4/2026, 10:39:37 AM · gpt-5-6-thinking



Хорошо, посмотрим на новую структуру: у нас есть одно разбиение с тремя блоками $A1:\{1\}$, $A2:\{2\}$, $A3:\{3\}$ - длина 3, 3 разбиения с двумя блоками $B1:\{1\}, \{2, 3\}$, $B2, B3$, и одно разбиение с одним блоком $C:\{1, 2, 3\}$. Теперь нас интересуют только атомы, в которые превратились блоки. $A1, A2, A3$ нам дают 6 списков длины 3, $B1, B2, B3$ дают 6 списков длины 2, и C - один список длины 1. Итого (через EGF): $1/1! \times x + 6/2! \times x^2 + 6/6! \times x^3 = x + 3x^2 + x^3$. Или я всё неправильно понял.

10/4/2026, 12:01:28 PM



Ты понял **структуру почти совершенно правильно**; ошибка только в последнем шаге — в том, **что считает переменная x** . И это как раз очень важный момент.

Для множества исходных меток

$$U = \{1, 2, 3\}$$

ты правильно насчитал:

- разбиение $1 + 1 + 1$: $3! = 6$ порядков блоков;
- три разбиения типа $1 + 2$, каждое даёт $2!$ порядка: $3 \cdot 2 = 6$;
- разбиение 3 : 1.

Итого

$$6 + 6 + 1 = 13$$

структур.

Но все эти 13 структур имеют **размер 3**, а не размеры 3, 2, 1.

Почему? Потому что размер композиции по-прежнему считается по **исходным атомам** 1, 2, 3. Блоки становятся «атомами» только локально — для внешней **seq**. Они не становятся новыми атомами, которые считает x .

Поэтому вклад всех этих структур в EGF:

$$13 \frac{x^3}{3!}.$$

Вот это различие стоит зафиксировать:

блоки = элементы, на которых строится внешняя F -структура;

но

x продолжает считать исходные атомы внутри всех блоков.

Если же ты **специально хочешь считать число блоков**, тогда вводим второй маркер, скажем u . И твой подсчёт превращается в совершенно правильный полином:

$$u + 6u^2 + 6u^3$$

для структур на фиксированном множестве $\{1, 2, 3\}$.

А в полной двухпеременной EGF соответствующий член будет

$$(u + 6u^2 + 6u^3) \frac{x^3}{3!}.$$

Это, по-моему, ещё один фундаментальный пункт для нашего списка непониманий:



При композиции блок временно играет роль «атома» для внешней структуры, но **не становится атомом размера 1 для общей производящей функции**. Общий размер остаётся числом исходных X -атомов.

И можно проверить всё аналитически:

$$G(x) = e^x - 1, \quad F(y) = \frac{1}{1-y},$$

поэтому

$$F(G(x)) = \frac{1}{1-(e^x-1)} = \frac{1}{2-e^x}.$$

Начало ряда:

$$1 + x + \frac{3}{2}x^2 + \frac{13}{6}x^3 + \dots$$

и действительно

$$\frac{13}{6}x^3 = 13\frac{x^3}{3!}.$$

Так что комбинаторную часть ты сделал правильно; споткнулись именно о **два уровня размера: число исходных атомов и число внешних блоков**.

10/4/2026, 12:01:44 PM · gpt-5-6-thinking



Приблизительно понял: строим структуру на $3x$ уникальных атомах (можно считать, что значение атома - это его id и есть, или считать, что у нас три уникальных значения - неважно каких.) Используем эту структуру в другой структуре. Надо найти сколько получается вариантов (с точностью до автоморфизмов) в этой сложной структуре. Но атомов у нас по-прежнему 3, так что всё идёт в x^3 .

10/4/2026, 12:11:15 PM



Да — почти. Единственное, что я бы поправил: фраза «с точностью до автоморфизмов» здесь немного опасна.

Мы строим сложную структуру на фиксированном множестве трёх атомов

$$U = \{1, 2, 3\}.$$

Внутренняя структура G организует эти атомы в блоки; затем внешняя F организует уже блоки. Но **носителем всей конструкции по-прежнему остаётся U** . Поэтому любая такая структура имеет размер 3 и идёт в

$$\frac{x^3}{3!}.$$

А коэффициент перед ним — число **различных помеченных структур на $\{1, 2, 3\}$** . Здесь это 13.

Аutomорфизмы возникают, если мы решим считать через непомеченные формы. В нашем примере формы особенно хорошо видны как композиции числа 3:

$$(3), \quad (1, 2), \quad (2, 1), \quad (1, 1, 1).$$

Они дают соответственно

$$1, \quad 3, \quad 3, \quad 6$$

помеченных структур, итого 13. Например, форма $(1, 2)$ имеет автоморфизм, переставляющий два атома внутри двухэлементного блока, поэтому

$$\frac{3!}{2!} = 3.$$

А форма $(1, 1, 1)$ уже полностью упорядочила три атома, автоморфизмов нет:

$$3! = 6.$$

Так что твоя основная формулировка верна:

внутренние блоки становятся объектами для внешней структуры, но исходных атомов всё равно три

и потому **всё сидит при x^3** .

Это, пожалуй, стоит занести в наш будущий список «мест, где species ломают программистскую интуицию».

10/4/2026, 12:11:25 PM · gpt-5-6-thinking





SEQ(SET()) использует автоморфизмы на уровне SET [{1,2} = {2,1}]. У SEQ автоморфизмов нет. Вот если бы было: SET(SET()) - у нас бы были автоморфизмы на обоих уровнях. Правильно?

10/4/2026, 12:16:55 PM



Да — именно так, с одной важной оговоркой: лучше писать

$$\text{SET}(\text{SET}_{\geq 1}(X)),$$

чтобы внутренние блоки не были пустыми.

Для

$$\text{SEQ}(\text{SET}_{\geq 1}(X))$$

симметрии есть **внутри каждого блока**:

$$\{1, 2\} = \{2, 1\}.$$

Но внешний SEQ фиксирует порядок блоков, поэтому менять блоки местами нельзя.

Например,

$$(\{1, 2\}, \{3, 4\})$$

имеет внутренние автоморфизмы

$$1 \leftrightarrow 2, \quad 3 \leftrightarrow 4,$$

то есть группу размера

$$2! \cdot 2! = 4.$$

Но обменять целиком первый и второй блок нельзя, потому что

$$(\{1, 2\}, \{3, 4\}) \neq (\{3, 4\}, \{1, 2\}).$$

А вот для

$$\text{SET}(\text{SET}_{\geq 1}(X))$$

внешний уровень тоже неупорядочен:

$$\{\{1, 2\}, \{3, 4\}\}.$$

Теперь можно:

1. переставлять элементы внутри каждого блока;
2. переставлять **сами блоки одинакового размера**.

Поэтому для формы $2 + 2$:

$$|\text{Aut}| = (2!)^2 \cdot 2! = 8.$$

Вообще если в разбиении есть m_j блоков размера j , то

$$|\text{Aut}| = \prod_{j \geq 1} (j!)^{m_j} m_j!$$

— $j!$ за внутреннюю симметрию каждого блока и $m_j!$ за перестановку одинаковых по размеру блоков.

Так что твоя формулировка очень хороша:

$$\text{SEQ}(\text{SET}): \text{симметрии внутри блоков}$$



SET(SET): симметрии и внутри блоков, и между блоками

И это как раз тот мост к Burnside-Pólya, который мы потом используем.

10/4/2026, 12:17:08 PM · gpt-5-6-thinking



Хорошо. Я не уверен, что всё понимаю, но продолжим. Для этого листка есть смысл идти очень медленно и с подробными разъяснениями, чтобы я привык.

10/4/2026, 12:20:06 PM



Да. Для этого листка лучше **сильно снизить темп** и не вводить Bell polynomials, пока сама композиция species не станет почти механической.

Предлагаю такой режим: один новый смысловой шаг за задачу, и после каждого — разбор того, **что именно считает формула**.

Сейчас закрепим только уже знакомый пример

$$\text{SEQ}(\text{SET}_{\geq 1}(X)).$$

У внутреннего вида

$$G = \text{SET}_{\geq 1}(X)$$

EGF

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

Внешний **SEQ** имеет

$$F(y) = 1 + y + y^2 + y^3 + \dots$$

После подстановки:

$$F(G(x)) = 1 + G(x) + G(x)^2 + G(x)^3 + \dots$$

И здесь есть очень конкретный смысл:

$$G(x)^k = \text{структуры ровно с } k \text{ упорядоченными блоками.}$$

Потому что произведение G^k означает k **6**-структур, а позиции $1, \dots, k$ делают их упорядоченными.

Следующая маленькая задача

Возьмём только коэффициент при x^3 .

Нужно отдельно найти вклад:

$$G, \quad G^2, \quad G^3.$$

То есть попробуй получить три числа:

$$[x^3]G, \quad [x^3]G^2, \quad [x^3]G^3,$$

а затем для каждого объяснить, почему после умножения на $3!$ мы должны получить соответственно число структур на $\{1, 2, 3\}$ с

$$1, \quad 2, \quad 3$$

упорядоченными блоками.

Это ровно тот же подсчёт $1 + 6 + 6 = 13$, но теперь посмотрим, **как EGF сама его производит**.

10/4/2026, 12:20:16 PM · gpt-5-6-thinking





$G = x + x^2/2 + x^3/6 + \dots$
 $G^2 = e^{2x} - 2e^x + 1 = 1 + 2x + 4x^2/2 + 8x^3/6 - 2 - 2x - 2x^2/2 - 2x^3/6 + 1$ [+ члены более высокого порядка] $= x^2 + x^3 + \dots$
 $G^3 = x^3 + \dots$
 Итого: $(1+6+6)/6 \times x^3$

10/4/2026, 12:37:21 PM



Да, всё правильно. Только в строке для G^2 , конечно, подразумевалось

$$G^2 = (e^x - 1)^2 = e^{2x} - 2e^x + 1.$$

И ты получил именно нужное:

$$\begin{aligned}
 G &= x + \frac{x^2}{2} + \frac{x^3}{6} + \dots, \\
 G^2 &= x^2 + x^3 + \dots, \\
 G^3 &= x^3 + \dots.
 \end{aligned}$$

Поэтому при x^3 :

$$[x^3]G = \frac{1}{6}, \quad [x^3]G^2 = 1, \quad [x^3]G^3 = 1.$$

Поскольку это EGF, умножаем каждое на $3! = 6$:

$$1, \quad 6, \quad 6.$$

И здесь особенно полезно видеть соответствие буквально:

$$\begin{aligned}
 \underbrace{G}_{1 \text{ блок}} &\longrightarrow 1, \\
 \underbrace{G^2}_{2 \text{ упорядоченных блока}} &\longrightarrow 6, \\
 \underbrace{G^3}_{3 \text{ упорядоченных блока}} &\longrightarrow 6.
 \end{aligned}$$

Итого

$$\boxed{13}.$$

То есть ты сейчас одним и тем же объектом посчитал дважды: сначала непосредственно через разбиения $\{1, 2, 3\}$, а теперь через композицию EGF. Это именно тот мост, который нам нужен.

Теперь сделаем **одно маленькое изменение**, чтобы увидеть роль внешней структуры.

Задача 2

Заменим внешний SEQ на SET:

$$\text{SET}(\text{SET}_{\geq 1}(X)).$$

Это уже обычное разбиение множества.

На уровне EGF:

$$\exp(G(x)) = 1 + G(x) + \frac{G(x)^2}{2!} + \frac{G(x)^3}{3!} + \dots$$

Используя только что найденные

$$[x^3]G = \frac{1}{6}, \quad [x^3]G^2 = 1, \quad [x^3]G^3 = 1,$$

найди коэффициент при x^3 , а затем переведи его обратно в число структур на $\{1, 2, 3\}$.

И главное: попробуй словами объяснить, **почему появились делители $2!$ и $3!$** по сравнению с SEQ.

10/4/2026, 12:37:32 PM · gpt-5-6-thinking



$1/6 + 1/2 + 1/6 = 1/6 + 3/6 + 1/6 = 5/6$, то есть 5 неизоморфных структур. Делители появляются потому, что порядок не важен - то есть из-за автоморфизмов второго уровня.

10/4/2026, 12:41:01 PM



Почти всё правильно. Арифметика:

$$\frac{1}{6} + \frac{1}{2} + \frac{1}{6} = \frac{5}{6},$$

поэтому число структур на фиксированном множестве $\{1, 2, 3\}$ равно

$$3! \cdot \frac{5}{6} = 5.$$

Но я бы поправил одно слово: это не **5 неизоморфных**, а **5 помеченных разбиений** множества $\{1, 2, 3\}$:

$$\begin{aligned} &\{123\}, \\ &\{1\}\{23\}, \quad \{2\}\{13\}, \quad \{3\}\{12\}, \\ &\{1\}\{2\}\{3\}. \end{aligned}$$

Неизоморфных форм здесь только три:

$$3, \quad 2 + 1, \quad 1 + 1 + 1.$$

А насчёт делителей $2!, 3!$: идея у тебя правильная — внешний **SET** убирает порядок блоков. Я лишь осторожнее употреблял бы слово «автоморфизмы». В

$$\frac{G^k}{k!}$$

деление на $k!$ прежде всего говорит:

G^k временно нумерует k компонент позициями $1, \dots, k$, а **SET** забывает этот порядок.

Например,

$$(\{1\}, \{2, 3\}) \quad \text{и} \quad (\{2, 3\}, \{1\})$$

в G^2 различны, но в **SET(G)** это одна структура

$$\{\{1\}, \{2, 3\}\}.$$

Так что можно говорить о **симметрии второго уровня**, но не всякое такое деление буквально есть размер группы автоморфизмов конкретной структуры.

И теперь очень маленький следующий шаг. Добавим маркер u числа блоков:

$$\exp(uG(x))$$

где $G = e^x - 1$.

Используя уже найденные коэффициенты при x^3 , попробуй получить коэффициент при $x^3/3!$ как полином по u .

Он должен непосредственно показать числа разбиений $\{1, 2, 3\}$ на 1, 2, 3 блока.

10/4/2026, 12:41:10 PM · gpt-5-6-thinking





Давай обсудим терминологию. Для меня множество - это кортеж с забытым порядком, то есть, добавляем группу перестановок (автоморфизмов) S_n , и на неё "делим". В математике, обычно, множество базовое понятие и пара определяется как $\{a, \{a, b\}\}$, но у нас, явно, не тот случай. Коэффициенты будут $u/6, u^2/2, u^3/6$, или $(u+3u^2+u^3)/6$, то есть, без $/6, u+3u^2+u^3$.

10/4/2026, 1:02:36 PM



Да, так гораздо лучше сформулировать. В нашем контексте твой взгляд

$$\text{SET}_k = \text{SEQ}_k / S_k$$

вполне рабочий: сначала берём k пронумерованных позиций, а затем **забываем их порядок**, факторизуя по действию S_k .

Я бы только развёл два употребления слова «автоморфизм»:

- S_k — **группа симметрий/перестановок позиций**, по которой мы факторизуем;
- «группа автоморфизмов конкретной непомеченной структуры» — уже стабилизатор этой структуры при перестановках её атомов.

Они тесно связаны, но это не буквально одно и то же понятие во всех ситуациях. Поэтому твоё «добавляем S_k и делим» я буду принимать; просто иногда буду говорить «**факторизуем по симметрии S_k** », чтобы не возникало двусмысленности.

И да, это вовсе не вопрос оснований теории множеств. SET здесь — **комбинаторный конструктор**. Мы вполне можем мыслить его как «кортеж, у которого забыли порядок». Это для нашей работы даже полезнее.

С коэффициентами у тебя всё правильно:

$$e^{uG} = 1 + uG + \frac{u^2 G^2}{2!} + \frac{u^3 G^3}{3!} + \dots$$

При x^3 :

$$\frac{u}{6} + \frac{u^2}{2} + \frac{u^3}{6} = \frac{u + 3u^2 + u^3}{6}.$$

То есть в EGF:

$$(u + 3u^2 + u^3) \frac{x^3}{3!}.$$

И полином

$$u + 3u^2 + u^3$$

говорит буквально:

- 1 разбиение на 1 блок;
- 3 разбиения на 2 блока;
- 1 разбиение на 3 блока.

Это ровно

$$S(3,1)u + S(3,2)u^2 + S(3,3)u^3.$$

Так мы почти незаметно снова пришли к числам Стирлинга второго рода.

Следующий шаг я бы сделал очень медленным: не вводить ещё Bell polynomials, а сначала посмотреть, **как общий коэффициент $G(x)^k/k!$ кодирует разбиение n атомов на k блоков определённых размеров**. Это как раз место, где Bell polynomials потом возникнут естественно.

10/4/2026, 1:02:49 PM · gpt-5-6-thinking



Хорошо. Продолжаем.

10/4/2026, 2:10:24 PM



Тогда следующий шаг - **один конкретный коэффициент**, без общей формулы.

Задача 3

Оставим

$$G(x) = e^x - 1$$

и рассмотрим

$$\frac{G(x)^2}{2!}.$$

Комбинаторно это означает:

разбиение множества на ровно два неупорядоченных непустых блока.

Возьмём $n = 4$, то есть множество

$$\{1, 2, 3, 4\}.$$

Есть только два возможных типа размеров блоков:

$$1 + 3 \quad \text{и} \quad 2 + 2.$$

Попробуй посчитать отдельно:

1. сколько разбиений типа $1 + 3$;
2. сколько типа $2 + 2$;
3. сколько всего.

А затем проверь то же самое через коэффициент

$$[x^4] \frac{(e^x - 1)^2}{2}.$$

После умножения этого коэффициента на $4!$ должно получиться то же число.

Здесь особенно интересно увидеть, **где именно появляется дополнительное деление на 2 для типа $2 + 2$** , а почему для $1 + 3$ фактически такого эффекта нет.

10/4/2026, 2:10:33 PM · gpt-5-6-thinking





Разбиений типа 1+3: $4 = C(4, 1)$; типа 2+2: $6 = C(4, 2)$, итого: 10. Коэффициент при x^4 $7/24$. Не сходится.

10/4/2026, 2:16:14 PM



Вот как раз хорошая ошибка: она показывает именно ту симметрию, которую мы хотели поймать.

Для типа $1 + 3$ всё верно:

$$\binom{4}{1} = 4.$$

Выбор одноэлементного блока автоматически определяет трёхэлементный, и перепутать их нельзя: размеры разные.

А для $2 + 2$:

$$\binom{4}{2} = 6$$

считает каждое разбиение **дважды**. Например,

$$\{1, 2\} \mid \{3, 4\}$$

получается и когда мы выбрали $\{1, 2\}$, и когда выбрали $\{3, 4\}$.

Поэтому

$$\frac{1}{2} \binom{4}{2} = 3.$$

Итого:

$$4 + 3 = 7.$$

Это ровно совпадает с

$$[x^4] \frac{(e^x - 1)^2}{2!} = \frac{7}{24},$$

потому что

$$4! \cdot \frac{7}{24} = 7.$$

Ключевой момент здесь такой:

если размеры блоков различны, они уже различимы по размеру

поэтому для $1 + 3$ дополнительного деления нет.

А при $2 + 2$ два блока одного типа, и перестановка блоков ничего не меняет:

$$A \mid B = B \mid A,$$

поэтому возникает фактор $2!$.

Это уже почти зародыш общей формулы: если есть m_j блоков размера j , то появляется деление на

$$m_j!.$$

Именно из таких факторов потом вырастут Bell polynomials.

10/4/2026, 2:16:21 PM · gpt-5-6-thinking





А, верно. Достаточно выбрать пару к 1 (2, 3, или 4).

10/4/2026, 2:17:22 PM



Да, именно. Для разбиений типа $2 + 2$ можно избежать двойного счёта так: зафиксировать атом 1 и выбрать, кто с ним в паре.

Варианты:

$$\{1, 2\} \mid \{3, 4\}, \quad \{1, 3\} \mid \{2, 4\}, \quad \{1, 4\} \mid \{2, 3\}.$$

Итого 3.

Это хороший комбинаторный способ увидеть тот же фактор $1/2!$, не вводя его формально.

Следующий маленький шаг предлагаю сделать на $n = 5$: посчитать число разбиений типа

$$2 + 3.$$

Здесь размеры блоков различны, поэтому попробуй сначала просто выбрать двухэлементный блок. Сколько получится?

10/4/2026, 2:17:27 PM · gpt-5-6-thinking





Если 1 слева, то 4 варианта. Если 1 справа, то $C(4,2) = 6$. Итого: 10. Или, просто выбираем 2 из 5: $C(5, 2) = 10$.

10/4/2026, 2:20:34 PM



Да, именно:

$$\binom{5}{2} = 10.$$

И здесь важно, почему **не делим ещё на 2!**: блоки размеров 2 и 3 различимы уже по размеру. Выбрав двухэлементный блок, мы однозначно определили трёхэлементный.

Теперь следующий шаг — как раз случай, где появится повторяющийся размер.

Задача 4

Разобьём

$$\{1, 2, 3, 4, 5\}$$

на блоки размеров

$$1 + 2 + 2.$$

Посчитай число таких разбиений.

Удобно идти так:

1. выбрать одноэлементный блок;
2. оставшиеся четыре элемента разбить на две **неупорядоченные** пары.

Попробуй получить число сначала комбинаторно. Затем посмотрим, как ровно тот же множитель возникает из EGF.

10/4/2026, 2:20:41 PM · gpt-5-6-thinking





Выбрать одноэлементный блок: 5. Умножить на 3 (это мы уже проходили).
Итого: 15.

10/4/2026, 2:22:31 PM



Да:

$$5 \cdot 3 = \boxed{15}.$$

Теперь посмотрим, как **точно то же самое возникает внутри EGF**. Это полезно, потому что здесь уже начинает просматриваться общая формула.

Мы смотрим на

$$\frac{G(x)^3}{3!}, \quad G(x) = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots,$$

потому что у нас ровно три блока.

Для типа $1 + 2 + 2$ из трёх множителей G надо взять

$$x, \quad \frac{x^2}{2!}, \quad \frac{x^2}{2!}.$$

Одноэлементный блок можно выбрать из трёх позиций G , поэтому внутри G^3 получаем

$$3 \cdot x \left(\frac{x^2}{2!} \right)^2 = \frac{3}{4} x^5.$$

Теперь внешний **SET** делит на $3!$:

$$\frac{1}{3!} \cdot \frac{3}{4} x^5 = \frac{1}{8} x^5.$$

Это EGF, поэтому число структур:

$$5! \cdot \frac{1}{8} = \boxed{15}.$$

Но особенно интересно переписать коэффициент так:

$$\frac{3}{3!} \frac{1}{(2!)^2} = \frac{1}{2!} \frac{1}{(2!)^2}.$$

Почему?

- $(2!)^2$ - порядок забывается **внутри каждого из двух двухэлементных блоков**;
- ещё один $2!$ - забывается порядок **между двумя одинаковыми по размеру блоками**;
- никакого дополнительного деления для одноэлементного блока нет: он единственный.

В итоге

$$\boxed{\frac{5!}{1! (2!)^2 2!} = 15.}$$

Вот уже почти общая закономерность. Если имеется m_j блоков размера j , то соответствующее число разбиений будет содержать знаменатель

$$\boxed{(j!)^{m_j} m_j!}.$$

Но пока не будем превращать это в формулу.

Следующая маленькая проверка: как ты ожидаешь посчитать разбиения 6 элементов типа

$$2 + 2 + 2?$$

Здесь все три блока одинакового размера, поэтому симметрия второго уровня должна проявиться особенно ясно.

10/4/2026, 2:22:41 PM · gpt-5-6-thinking



Просто из соображений симметрии: $6!/(2!)^3/3! = 15$. Комбинаторно: выбираем пару для 1 (5 вариантов); 2+2 - это 3 варианта. $3 \times 5 = 15$. Или $G^3/6$ (три блока), три множителя $x^2/2 = x^6/8$, внешний SET делит на $3!$, итого: $6!/2!^3/3!$. То же значение.

10/4/2026, 4:03:58 PM



Да, всё три способа совпали — и это как раз знак, что механизм уже начинает становиться естественным.

Для $2 + 2 + 2$:

$$\frac{6!}{(2!)^3 3!} = 15.$$

Здесь особенно чисто видны два уровня симметрии:

$$(2!)^3$$

— забываем порядок внутри каждой пары, а

$$3!$$

— забываем порядок трёх пар между собой.

Комбинаторный подсчёт

$$5 \cdot 3 = 15$$

— тот же самый факторинг симметрий, только без формул.

Теперь можно сделать **один осторожный шаг к Bell polynomials**.

Пусть m_j означает: сколько у нас блоков размера j . Например, для

$$1 + 2 + 2$$

имеем

$$m_1 = 1, \quad m_2 = 2,$$

а для

$$2 + 2 + 2$$

$$m_2 = 3.$$

Тогда обязательно

$$\sum_j j m_j = n,$$

потому что всего n атомов, и

$$\sum_j m_j = k,$$

если всего k блоков.

А число разбиений данного типа должно быть

$$\frac{n!}{\prod_j (j!)^{m_j} m_j!}.$$

Мы эту формулу уже фактически вывели на примерах.

Задача 5

Проверь её на типе

$$1 + 1 + 2 + 3$$

для $n = 7$.

То есть здесь

$$m_1 = 2, \quad m_2 = 1, \quad m_3 = 1.$$

Попробуй:

1. получить число по общей формуле;
2. объяснить каждый множитель в знаменателе комбинаторно.

После этого Bell polynomial появится буквально как «сумма этой формулы по всем допустимым наборам m_j ».

10/4/2026, 4:04:07 PM · gpt-5-6-thinking



1. Формула: $7!/(1!^2 \times 2!)/(2!^1 \times 1!)/(3!^1 \times 1!) = 7!/(2 \times 2 \times 6) = 210$
2. $j!^{m_j}$ - это внутренние симметрии блока (с учётом количества блоков этого размера - m_j и размера блока - j). $m_j!$ - это внешняя симметрия, одинаковые блоки можно переставлять местами, а их m_j для размера j .

10/4/2026, 4:12:00 PM



Да, всё верно:

$$\frac{7!}{(1!)^2 2! (2!)^1 1! (3!)^1 1!} = \frac{5040}{24} = \boxed{210}.$$

И объяснение знаменателя тоже правильное. Только одно терминологическое уточнение: $m_j!$ переставляет не «одинаковые блоки» в смысле одинакового содержимого — содержимое у них разное, — а **блоки одинакового размера j** . Например, два одноэлементных блока $\{1\}, \{4\}$ различны как множества, но во внешнем SET нет «первого» и «второго» одноэлементного блока.

Теперь мы готовы ввести Bell polynomial почти без новой идеи.

Пусть

$$G(x) = \sum_{j \geq 1} g_j \frac{x^j}{j!}.$$

Здесь g_j — число возможных G -структур на j заданных метках.

Хотим построить ровно k неупорядоченных G -блоков:

$$\frac{G(x)^k}{k!}.$$

Для фиксированного набора

$$(m_1, m_2, \dots),$$

где

$$\sum_j m_j = k, \quad \sum_j j m_j = n,$$

вклад равен

$$n! \prod_j \frac{1}{m_j!} \left(\frac{g_j}{j!} \right)^{m_j}.$$

Почему появился $g_j^{m_j}$? Потому что теперь на каждом блоке размера j может быть не одна структура, как у SET, а g_j разных G -структур.

И вот **Bell polynomial** — это просто сумма этих вкладов по всем возможным типам блоков:

$$B_{n,k}(g_1, g_2, \dots) = n! \sum_{\substack{\sum m_j = k \\ \sum j m_j = n}} \prod_j \frac{1}{m_j!} \left(\frac{g_j}{j!} \right)^{m_j}.$$

Причём

$$\frac{G(x)^k}{k!} = \sum_{n \geq k} B_{n,k}(g_1, g_2, \dots) \frac{x^n}{n!}.$$

То есть никакой новой магии: $B_{n,k}$ просто говорит:

«Сколько способов разбить n меток на k неупорядоченных блоков и на каждом блоке построить G -структуру?»

В нашем старом случае

$$G = \text{SET}_{\geq 1}(X),$$

на любом непустом блоке есть ровно одна структура, поэтому

$$g_1 = g_2 = g_3 = \dots = 1.$$

Тогда

$$B_{n,k}(1, 1, \dots) = S(n, k)$$

— обычные числа Стирлинга второго рода.

Задача 6

Возьмём $n = 5$, $k = 3$, пока всё ещё $g_j = 1$.

Какие возможны только **два типа размеров блоков**?

После этого посчитаем их вклады и тем самым получим $B_{5,3}(1, 1, \dots) = S(5, 3)$.

10/4/2026, 4:12:12 PM · gpt-5-6-thinking



3,1,1 и 2,2,1

10/4/2026, 4:16:19 PM



Да, ровно эти два типа:

$$3 + 1 + 1 \quad \text{и} \quad 2 + 2 + 1.$$

Теперь посчитаем их по нашей формуле.

Для $3 + 1 + 1$:

$$m_1 = 2, \quad m_3 = 1,$$

поэтому

$$\frac{5!}{(1!)^2 2! (3!)^1 1!} = \frac{120}{2 \cdot 6} = 10.$$

Для $2 + 2 + 1$:

$$m_1 = 1, \quad m_2 = 2,$$

поэтому

$$\frac{5!}{(1!)^1 1! (2!)^2 2!} = \frac{120}{4 \cdot 2} = 15.$$

Итого

$$10 + 15 = 25.$$

Значит,

$$B_{5,3}(1, 1, \dots) = S(5, 3) = 25.$$

Это хороший момент: **Bell polynomial** здесь просто суммирует по типам разбиения.

Следующий шаг я бы сделал совсем маленький: оставить $n = 5$, $k = 3$, но взять не все $g_j = 1$, а, например,

$$g_1 = 1, \quad g_2 = 2, \quad g_3 = 1.$$

То есть на двухэлементном блоке теперь есть **две возможные внутренние G -структуры**.

Тогда попробуй понять, во сколько раз изменятся вклады типов

$$3 + 1 + 1 \quad \text{и} \quad 2 + 2 + 1.$$

Здесь впервые станет видно, зачем в Bell polynomial стоят степени $g_j^{m_j}$.

10/4/2026, 4:16:21 PM · gpt-5-6-thinking





Каждой двухэлементной структуре соответствует множитель 2, то есть вес разбиения $2+2+1$ увеличится в 4 раза.

10/4/2026, 4:19:12 PM



Да, именно.

Тип $3 + 1 + 1$ вообще не меняется, потому что двухэлементных блоков там нет:

$$10.$$

А в типе $2 + 2 + 1$ есть два блока размера 2, и на каждом теперь по $g_2 = 2$ возможных внутренних структур. Поэтому появляется множитель

$$2^2 = 4.$$

Было

$$15,$$

становится

$$15 \cdot 4 = 60.$$

Итого

$$\boxed{10 + 60 = 70}.$$

То есть здесь уже буквально видно, откуда в Bell polynomial берётся

$$g_j^{m_j}.$$

Если блоков размера j ровно m_j , и на каждом таком блоке есть g_j возможных внутренних структур, то независимый выбор на каждом блоке даёт

$$\boxed{g_j^{m_j}}.$$

Следующий маленький шаг я бы сделал уже на самой композиции $F(G(x))$: Bell polynomial считает внутренний слой при фиксированном числе блоков k , а коэффициент f_k внешней species скажет, сколько способов эти k блоков организовать снаружи. Это будет последний мост перед Faà di Bruno.

10/4/2026, 4:19:14 PM · gpt-5-6-thinking



Хорошо.

10/4/2026, 4:20:10 PM



Тогда теперь соединяем **внутренний** и **внешний** уровни.

Пусть

$$F(y) = \sum_{k \geq 0} f_k \frac{y^k}{k!}, \quad G(x) = \sum_{j \geq 1} g_j \frac{x^j}{j!}.$$

Мы уже выяснили, что

$$B_{n,k}(g_1, g_2, \dots)$$

считает способы разбить n исходных атомов на k неупорядоченных G -блоков, построив на каждом блоке G -структуру.

Теперь эти k блоков становятся k «атомами» для внешней species F . На фиксированном множестве из k таких блоков имеется

$$f_k$$

возможных F -структур.

Поэтому просто умножаем:

$$a_n = \sum_{k=0}^n f_k B_{n,k}(g_1, g_2, \dots)$$

и это коэффициенты композиции

$$F(G(x)) = \sum_{n \geq 0} a_n \frac{x^n}{n!}.$$

Задача 7 — проверка на нашем старом примере

Возьмём

$$G = \text{SET}_{\geq 1}(X),$$

поэтому

$$g_j = 1$$

и

$$B_{n,k}(1, 1, \dots) = S(n, k).$$

А снаружи возьмём

$$F = \text{SEQ}.$$

Сколько здесь f_k , то есть сколько **SEQ**-структур имеется на фиксированном множестве из k блоков?

После этого подставь $n = 3$ в

$$a_3 = \sum_k f_k S(3, k)$$

и проверь, что снова получаются наши 13.

10/4/2026, 4:20:22 PM · gpt-5-6-thinking



$$f_k = k!, a_3 = 1 \times 1 + 2 \times 3 + 6 \times 1 = 13$$

10/4/2026, 4:24:33 PM



Да, совершенно верно:

$$f_k = k!,$$

потому что на фиксированном множестве из k блоков `SEQ` задаёт один из $k!$ линейных порядков.

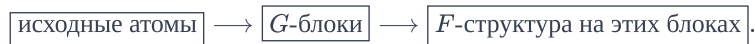
И

$$a_3 = 1! S(3, 1) + 2! S(3, 2) + 3! S(3, 3)$$

даёт

$$1 \cdot 1 + 2 \cdot 3 + 6 \cdot 1 = \boxed{13}.$$

То есть теперь у нас есть достаточно ясная схема:



Bell polynomial считает первый переход при фиксированном числе блоков k , а f_k — второй.

Теперь можно очень осторожно перейти к Faà di Bruno.

Задача 8

Пусть

$$H(x) = F(G(x)).$$

Мы уже знаем:

$$H'(x) = F'(G(x)) G'(x).$$

Теперь выделим **два различных атома** — скажем, красный и синий.

Комбинаторно возможны ровно два принципиальных случая:

1. красный и синий попали в **один и тот же G -блок**;
2. они попали в **два разных G -блока**.

Попробуй по этим двум случаям восстановить формулу для

$$H''(x).$$

Не надо пока механически дифференцировать. Именно из двух комбинаторных случаев.

10/4/2026, 4:24:42 PM · gpt-5-6-thinking





Один и тот же блок: $F'(G)G''$. В F только один блок затронут, зато дважды. В G выделены два элемента (заполнены две дырки).

Плюс:

Два разных блока: $F''(G)G'^2$. В F отмечены два блока (заполнены две дырки с помощью двух блоков G , в каждом из которых выделено по одному элементу [заполнены по одной дырке в каждом]).

10/4/2026, 4:32:10 PM



Да, именно так:

$$H''(x) = F'(G(x)) G''(x) + F''(G(x))(G'(x))^2$$

И твоя комбинаторная расшифровка правильная.

В первом случае два выделенных атома оказываются внутри одного и того же внутреннего G -блока. Поэтому во внешней F -структуре выделен один блок:

$$F'(G),$$

а внутри него два выделенных атома:

$$G''.$$

Во втором случае выделенные атомы находятся в двух разных G -блоках. Тогда во внешней структуре выделены два разных блока:

$$F''(G),$$

и в каждом из этих двух блоков выделено по одному атому:

$$(G')^2.$$

То есть уже здесь правило второй производной — это просто разбиение множества двух флажков:

$$\{1, 2\}$$

либо как один блок

$$\{1, 2\},$$

либо как два блока

$$\{1\}\{2\}.$$

И вот это — самая важная идея перед Faà di Bruno:

слагаемые соответствуют разбиениям множества выделенных атомов.

Для третьей производной будут уже три типа разбиений:

$$3, \quad 2 + 1, \quad 1 + 1 + 1.$$

Это и будет следующая задача.

10/4/2026, 4:32:15 PM · gpt-5-6-thinking





Хорошо. Поехали.

10/4/2026, 4:32:59 PM



Тогда **Задача 9: третья производная как разбиение трёх флажков.**

Пусть снова

$$H(x) = F(G(x)).$$

Теперь у нас три различных выделенных атома: скажем, красный, синий и зелёный.

Есть ровно три типа того, как они могут распределиться по внутренним G -блокам:

$$3, \quad 2 + 1, \quad 1 + 1 + 1.$$

Попробуй для каждого типа написать соответствующее слагаемое.

То есть нужно восстановить

$$H'''(x)$$

чисто комбинаторно.

Особенно важно объяснить, откуда в типе

$$2 + 1$$

появится числовой коэффициент.

10/4/2026, 4:33:02 PM · gpt-5-6-thinking





Три компонента: $F'''(G)G'^3$ - затронуты три разных блока, по одной дырке в каждом. Всё симметрично.
 $3 \times F''(G) \times G' \times G''$ - затронуто два блока, один из них дважды, для каждой комбинации три варианта из-за внешней асимметрии. $F'(G)G'''$ - затронут только один блок, но трижды. В этом блоке отмечены 3 элемента. Всё симметрично.

10/4/2026, 4:42:02 PM



Да, формула получилась совершенно правильно:

$$H''' = F'''(G)(G')^3 + 3F''(G)G'G'' + F'(G)G'''.$$

Только я бы чуть поправил объяснение коэффициента 3. Это не столько «внешняя асимметрия», сколько число разбиений трёх **различимых флажков** типа $2 + 1$.

Пусть флажки a, b, c . Возможны:

$$\{a, b\} | \{c\}, \quad \{a, c\} | \{b\}, \quad \{b, c\} | \{a\}.$$

Именно поэтому коэффициент 3.

Для типа $1 + 1 + 1$ существует ровно одно разбиение множества флажков:

$$\{a\} | \{b\} | \{c\},$$

отсюда коэффициент 1, хотя сами три блока различимы тем, **какой флажок в них находится**. Аналогично для типа 3 имеется одно разбиение:

$$\{a, b, c\}.$$

И теперь можно увидеть Faà di Bruno почти целиком:

$$n\text{-я производная } F(G) \leftrightarrow \text{разбиения множества из } n \text{ флажков.}$$

Каждый блок разбиения означает: «эти производные попали в одну и ту же внутреннюю G -компоненту». Если блок имеет размер j , он даёт $G^{(j)}$. Если всего блоков k , внешний слой даёт $F^{(k)}(G)$.

Например, для некоторого разбиения размеров

$$3 + 2 + 2 + 1$$

возникнет множитель

$$F^{(4)}(G) G'''(G'')^2 G'.$$

А Bell polynomial как раз суммирует такие слагаемые по всем разбиениям заданного типа.

Я бы на этом **Листок 8 закончил**: композиция $\rightarrow$ блоки $\rightarrow$ Bell polynomials $\rightarrow$ многократное правило цепочки. Причём большую формулу Faà di Bruno мы фактически поняли до того, как её записали.

10/4/2026, 4:42:13 PM · gpt-5-6-thinking





Для меня это был довольно сложный листок, но, думаю, это просто моя особенность - в принципе, я понимаю, что ничего сложного в нём нет. Хорошо, листок 9 (возможно, с предисловием).

10/4/2026, 4:44:55 PM



Да, и я бы не списывал это на твою «особенность». Листок 8 действительно был концептуально плотным: там одновременно работали два уровня структуры, EGF-нормировка, разбиения и композиция. Арифметика простая, но объект всё время надо держать в двух масштабах.

Для **Листка 9** предисловие точно нужно, потому что мы специально отложили одну неприятную двусмысленность слова «дырка».

Листок 9. Атомная дырка, рекурсивная дырка и zipper

Есть три близких, но разных понятия.

1. Обычная производная species

$$F'[U] = F[U \sqcup \{*\}].$$

Здесь * заменяет **один атом того сорта, который считает x** .

То есть

$$\frac{\partial}{\partial X}$$

означает: найти один атом X и сделать его специальным.

2. Производная по рекурсивному месту

Если тип задан рекурсивно, например

$$T = \Phi(X, T),$$

можно сначала временно написать

$$\Phi(X, Y)$$

и спросить:

$$\frac{\partial \Phi}{\partial Y}.$$

Здесь Y — не атом размера, а **место, куда вставляется целая рекурсивная структура T** .

Поэтому такая производная означает:

один слой конструкции с дыркой вместо одного рекурсивного поддерева.

Это уже другая производная.

3. Zipper

Чтобы сфокусироваться на поддереве глубоко внутри структуры, одного слоя недостаточно. Надо сохранить **всю цепочку слоёв от фокуса до корня**.

Это и есть context/zipper:

фокус + информация, достаточная для восстановления всего объекта.

Начнём не с деревьев, а со списков — там различие особенно видно.

Задача 1. Список: две разные дырки

Пусть

$$L = 1 + XL.$$

Возьмём конкретный список

$$[a, b, c, d].$$

(a) Атомная производная

Мы уже знаем:

$$L' = L^2.$$

Если выделить атом c , объект можно записать как

$$[a, b] \square [d].$$

Почему здесь нужны **два списка**?

(b) Теперь рекурсивная дырка

Временно запишем конструктор списка как

$$\Phi(X, Y) = 1 + XY, \quad L = \Phi(X, L).$$

Тогда

$$\frac{\partial \Phi}{\partial Y} = X.$$

Что означает этот X как **один слой контекста**?

Например, для

$$[a, b, c, d]$$

если фокусом является целый хвост

$$[c, d],$$

какую информацию надо сохранить, чтобы потом восстановить исходный список?

Здесь пока главное — почувствовать разницу между:

«дырка вместо атома c »

и

«дырка вместо целого подсписка $[c, d]$ ».

10/4/2026, 4:45:06 PM · gpt-5-6-thinking





Не уверен, что я понял твои вопросы. Давай попробуем разобраться. Обычная производная по атому: просто заменяем один атом на специальный атом (*), и в расчётах этот атом более не участвует. Зато он меняет структуру и тем влияет на общий счёт. Например, для списков * может стоять в любой из n позиций, и, стало быть, увеличивает количество вариантов для одной и той же последовательности из $n-1$ атомов в n раз. Зачем нам два списка? В общем-то, незачем. Мы можем список со * представить и по-другому. Производная по рекурсивному месту: ничего не понял, раньше мы о таком не говорили. В случае списка мы, похоже, удаляем рекурсивный хвост, а X - это начальный элемент списка. В случае деревьев Каталана $X+Y^2$, мы получаем $2Y$: да/нетхдерево Каталана (??). В случае zipper.. я как-то не думал, что zipper принципиально отличается от производной по атому (у меня сложилось впечатление, что их рассматривают как одно и то же), но, в принципе, это структура (одна из), которая хранит достаточно информации, чтобы восстановить исходную (или отредактировать), начиная с нашей дырки/выделенного атома.

10/4/2026, 5:33:57 PM



Да. И здесь твоя реакция показывает, что я слишком резко ввёл новую терминологию. **Производную по рекурсивному месту мы раньше действительно не разбирали.** А насчёт zipper твоя прежняя интуиция была во многом правильной.

Разделим вещи аккуратно.

Обычная производная species $F'(X)$ - ровно как ты сказал: один X -атом заменяется специальным *, который больше не считается размером. Представление результата не уникально. Для списка

$$L' = L^2$$

два списка - это лишь удобное кодирование:

$$[a, b, *, d, e] \leftrightarrow ([a, b], [d, e]).$$

Ты совершенно прав: **«зачем два списка?» - принципиально незачем.** Они просто особенно удобны, потому что позволяют восстановить список.

Теперь совершенно новая вещь. Пусть рекурсивный тип задан не сразу как L , а через один шаг конструктора:

$$L = \Phi(X, L), \quad \Phi(X, Y) = 1 + XY.$$

Здесь Y означает: «сюда вставляется ещё один целый список».

Если формально продифференцировать **конструктор** по этому месту Y ,

$$\frac{\partial \Phi}{\partial Y} = X,$$

то мы не дифференцируем сам готовый тип L . Мы спрашиваем:

Как выглядит **один родительский слой**, если вместо одного рекурсивного потомка Y сделать дырку?

Для списка один слой выглядит так:

$$a :: \square.$$

Нужно сохранить только голову a ; дырка стоит вместо хвоста.

Поэтому для

$$[a, b, c, d]$$

и фокуса на целом хвосте

$$[c, d]$$

контекст равен

$$[a, b, \square].$$

Но он состоит уже из **двух однослойных контекстов**:

$$a :: \square, \quad b :: \square.$$

То есть $\partial\Phi/\partial Y = X$ даёт не полный zipper, а **один кадр / frame**. Цепочка таких кадров даёт полный путь от фокуса к корню.

И с деревом Каталана ты сразу попал точно в цель. Если

$$B = X + B^2$$

и временно

$$\Phi(X, Y) = X + Y^2,$$

то

$$\frac{\partial\Phi}{\partial Y} = 2Y.$$

Твоё «да/нет × дерево Каталана» - практически именно это. Лучше назвать:

лево/право × соседнее поддерево.

Один frame:

- дырка была слева → запомнить правое поддерево;
- дырка была справа → запомнить левое поддерево.

После подстановки $Y = B$:

$$\boxed{2B}.$$

А чтобы дырка могла находиться на произвольной глубине, нужна последовательность таких frames:

$$\boxed{\text{SEQ}(2B)}.$$

И вот здесь происходит красивое совпадение с тем, что мы уже получили раньше:

$$B' = \text{SEQ}(2B).$$

Так что твоя интуиция «производная и zipper - почти одно и то же» **не была ошибочной**. Именно на этом основан известный лозунг «derivative of a datatype = one-hole contexts». Просто надо уточнять: **по чему мы дифференцируем и что именно считается дыркой**.

На списках различие видно особенно хорошо:

$$\text{атомная дырка в } [a, b, c, d] : \quad [a, b] \square [d],$$

поэтому контекст содержит и то, что слева, и то, что справа;

а

$$\text{дырка вместо рекурсивного хвоста } [c, d] : \quad [a, b, \square],$$

поэтому нужен только префикс.

Это, думаю, и есть правильное предисловие к Листку 9.

И я начну вести наш маленький реестр концептуальных узлов: сейчас там уже явно есть **identity vs value**, **исходные атомы vs блоки внешней структуры**, и теперь **атомная дырка vs рекурсивная дырка/frame vs полный zipper**.

Следующую задачу я бы сделал буквально на одном списке: взять $[a, b, c, d]$ и отдельно построить контекст для фокуса на атоме c и для фокуса на подсписке $[c, d]$.

10/4/2026, 5:34:12 PM · gpt-5-6-thinking





Контекст на атоме с это, например, два списка [a, b] и [d]. Контекст для подписка [c, d] - это два фрейма a:: и b::. Скажем, список [a::, b::].

10/4/2026, 5:39:47 PM



Да, именно. Только я бы чуть уточнил представление второго контекста.

Для атома с:

$$[a, b, c, d] \longrightarrow ([a, b], [d]).$$

Это атомная дырка:

$$[a, b] \square [d].$$

Для подписка [c, d] действительно получаем два фрейма:

$$a :: \square, \quad b :: \square.$$

Их удобно хранить как стек

$$[a ::, b ::].$$

Только при восстановлении они применяются **в обратном направлении**:

$$[c, d] \xrightarrow{b::} [b, c, d] \xrightarrow{a::} [a, b, c, d].$$

То есть полный рекурсивный контекст здесь можно считать либо префиксом [a, b], либо стеком однослойных фреймов. Это две эквивалентные кодировки.

И теперь уже видно различие очень чисто:

$$\boxed{\text{атомная дырка: } L \times L}$$

потому что надо помнить обе стороны атома,

а

$$\boxed{\text{рекурсивная дырка: } \text{SEQ}(X)}$$

потому что путь от подписка к корню состоит из последовательности фреймов X .

Следующий шаг можно сделать на бинарном дереве Каталана, где каждый фрейм уже не просто X , а

$$\boxed{2B}$$

— «с какой стороны дырка + соседнее поддерево».

10/4/2026, 5:39:56 PM · gpt-5-6-thinking



Да, я понимаю.

10/4/2026, 5:41:24 PM



Тогда переходим к дереву Каталана.

Задача 2. Один frame бинарного дерева

Пусть

$$B = X + B^2,$$

то есть дерево либо лист X , либо внутренний узел с левым и правым поддеревом.

Для одного рекурсивного шага пишем

$$\Phi(X, Y) = X + Y^2.$$

Тогда

$$\frac{\partial \Phi}{\partial Y} = 2Y.$$

Попробуй теперь расшифровать $2Y$ совершенно конкретно.

Представь узел

$$\text{Node}(L, R).$$

Если фокусом является поддерево L , какую информацию должен хранить один frame?

А если фокусом является R ?

Нужно получить два типа frame, которые вместе и дают $2B$.

10/4/2026, 5:41:28 PM · gpt-5-6-thinking





L: (мы пришли слева, R), для R - наоборот.

10/4/2026, 5:42:43 PM



Да, ровно так.

Для узла

$\text{Node}(L, R)$

один frame хранит:

- если фокус был в L : «пришли слева» + сохранить R ;
- если фокус был в R : «пришли справа» + сохранить L .

То есть два типа:

$(\text{Left}, R), \quad (\text{Right}, L).$

Именно это и есть

$2B$.

Теперь следующий маленький шаг.

Если фокус находится не прямо под корнем, а глубже, например путь от корня к фокусу такой:

$\text{left} \rightarrow \text{right} \rightarrow \text{left},$

то сколько frames должен хранить полный контекст?

И как бы ты записал этот zipper схематически?

10/4/2026, 5:44:06 PM · gpt-5-6-thinking



З, как последовательность фреймов. Я думаю, это всё довольно ясно (в отличие от предыдущего листка), и можно ускорить процесс.

10/4/2026, 5:45:46 PM



Да. Тогда оставшуюся часть листка 9 можно сильно ускорить и сразу выйти на главное.

Для рекурсивного типа

$$T = \Phi(X, T)$$

обозначим

$$A = \Phi_Y(X, T).$$

Это **один frame**: один шаг от фокуса к родителю. Полный рекурсивный контекст C - последовательность таких frames:

$$C = 1 + AC,$$

то есть

$$C = \text{SEQ}(A).$$

Теперь продифференцируем само рекурсивное уравнение по настоящему атому X :

$$T' = \Phi_X + \Phi_Y T' = \Phi_X + AT'.$$

Отсюда структурно:

$$T' = C \Phi_X.$$

Вот это, по-моему, главный результат всего листка.

Он сразу объясняет нашу прежнюю путаницу.

Для списка

$$L = 1 + XL, \quad \Phi(X, Y) = 1 + XY.$$

Имеем

$$\Phi_Y = X,$$

поэтому рекурсивный контекст

$$C = \text{SEQ}(X) = L.$$

Это просто префикс перед сфокусированным хвостом.

Но

$$\Phi_X = Y,$$

а после $Y = L$:

$$\Phi_X = L.$$

Поэтому

$$L' = C L = L^2.$$

То есть атомная дырка требует двух вещей: **контекст до того рекурсивного слоя** и **остаток списка после выбранного атома**.

А для дерева Каталана

$$B = X + B^2, \quad \Phi(X, Y) = X + Y^2$$

имеем

$$\Phi_Y = 2Y \Rightarrow A = 2B,$$

поэтому

$$C = \text{SEQ}(2B).$$

Но теперь

$$\Phi_X = 1.$$

Следовательно

$$\boxed{B' = C}.$$

Вот почему именно для наших каталановских деревьев у нас создалось впечатление

«производная = zipreg-контекст».

Там это действительно так! Просто это **не универсальная истина**: дополнительный множитель Φ_X случайно равен 1.

Ускоренная задача

Теперь проверим, что это действительно понято, на типе

$$D = X + XD^2,$$

где **атомами считаются все вершины**: и лист, и внутренний узел содержит X .

Запиши

$$\Phi(X, Y) = X + XY^2$$

и найди без подробных вычислений:

$$\Phi_Y, \quad C, \quad \Phi_X, \quad D'.$$

И главное - объясни словами, **почему теперь атомная производная D' уже не равна одному только рекурсивному контексту C .**

10/4/2026, 5:45:59 PM · gpt-5-6-thinking



$\Phi_Y = 2XY$. $A = 2XD$. $C = \text{SEQ}(2XD)$. $\Phi_X = 1 + Y^2 = 1 + D^2$. $D' = C \times \Phi_X = \text{SEQ}(2XD) \times (1 + D^2)$. В отличие от деревьев Каталана у нас возможно `Maybe()` монада, `1+` непустое начало, а именно, два поддерева нашей дырки $(1 + D^2)$. А потом мы восстанавливаем постепенно, фрейм за фреймом: влево/вправо, атом, альтернативное поддерево: $2XD$.

10/4/2026, 5:59:55 PM



Да, всё правильно. И здесь как раз хорошо видно различие между **локальной частью вокруг выбранного атома** и **путём к корню**.

$$\Phi_Y = 2XY, \quad A = 2XD, \quad C = \text{SEQ}(2XD),$$

$$\Phi_X = 1 + Y^2 \Rightarrow 1 + D^2,$$

поэтому

$$D' = \text{SEQ}(2XD)(1 + D^2).$$

Только я бы чуть поправил словесную интерпретацию $1 + D^2$. Это не столько «непустое начало», сколько **то, что непосредственно окружает выбранный атом**. Есть две возможности:

1

— выбранный атом был листом;

или

D^2

— выбранный атом был внутренней вершиной, и тогда под ним надо сохранить два поддерева.

Твоя программистская ассоциация вполне точна:

$$1 + D^2 \simeq \text{Maybe}(D \times D).$$

После этого действительно идём вверх по последовательности frames. Каждый

$2XD$

хранит:

лево/право + атом родителя + соседнее поддерево.

То есть zipper для выделенной вершины здесь можно мысленно разложить так:

$$\underbrace{\text{локальное содержимое вокруг фокуса}}_{1 + D^2} + \underbrace{\text{стек frames до корня}}_{\text{SEQ}(2XD)}$$

(«+» здесь словесное, формально произведение).

И это, пожалуй, хороший итог Листка 9:

$$T' = \underbrace{\text{SEQ}(\Phi_Y)}_{\text{путь/zipper context}} \cdot \underbrace{\Phi_X}_{\text{что остаётся вокруг выбранного атома}}$$

для рекурсивного уравнения $T = \Phi(X, T)$, в наших простых случаях.

На этом Листок 9 я бы считал освоенным. Следующий по плану — **Листок 10: производные регулярных выражений по Бржозовскому**. Там слово «производная» означает уже не дырку, а остаточный язык после чтения символа — то есть будет полезная проверка, насколько общий сам алгебраический способ мышления.

10/4/2026, 6:00:06 PM · gpt-5-6-thinking





Очень хороший листок (9)! Крайне эстетично и проясняет мозги! ОК, листок 10.

10/4/2026, 6:02:18 PM



Тогда **Листок 10: производные Бржозовского**. Здесь действительно будет другая производная, но алгебраический стиль мышления очень похож.

Короткое предисловие

Пусть регулярное выражение R задаёт язык $L(R)$. Производная Бржозовского по букве a определяется так:

$$D_a L = \{ w : aw \in L \}.$$

То есть мы спрашиваем:

Если я уже прочитал букву a , какие продолжения w ещё могут привести к слову нашего языка?

Например,

$$L = ab^* = \{a, ab, abb, abbb, \dots\}.$$

После чтения a остаётся

$$D_a(ab^*) = b^*.$$

А после чтения b :

$$D_b(ab^*) = \varnothing,$$

потому что ни одно слово языка не начинается с b .

Это **не дырка** и не выделенный атом. Это скорее:

остаточная задача после того, как часть входа уже съедена.

И здесь сразу возникает связь с автоматами: если мы прочли слово u , то состояние автомата можно мыслить как

$$D_u(R).$$

Два префикса ведут в одно состояние, если после них остаётся один и тот же остаточный язык.

Ещё понадобится одно булево свойство:

$$\text{nullable}(R)$$

- допускает ли R пустое слово ε .

Задача 1

Возьмём

$$R = (a + b)b.$$

То есть язык

$$L(R) = \{ab, bb\}.$$

Найди непосредственно из определения:

$$D_a(R) \quad \text{и} \quad D_b(R).$$

Пока никаких правил дифференцирования - просто: «прочитал первую букву; что может остаться?»

10/4/2026, 6:02:32 PM · gpt-5-6-thinking



б и б

10/4/2026, 6:28:58 PM



Да, верно:

$$D_a((a+b)b) = b, \quad D_b((a+b)b) = b.$$

Это уже хороший пример того, что **разные прочитанные префиксы могут приводить к одному и тому же остаточному языку**.

Следующий шаг.

Задача 2

Возьмём

$$R = a^*b.$$

Язык:

$$\{b, ab, aab, aaab, \dots\}.$$

Найди непосредственно:

$$D_a(R) \quad \text{и} \quad D_b(R).$$

И заодно скажи, является ли R nullable, то есть принадлежит ли ϵ языку R .

10/4/2026, 6:29:01 PM · gpt-5-6-thinking





$D_a(R) = a^*b$, $D_b(R) = \varepsilon$, $\text{nullable}(R) = \text{false}$

10/4/2026, 8:48:19 PM



Да, всё верно:

$$D_a(a^*b) = a^*b,$$

потому что после одной прочитанной a задача остаётся той же самой.

$$D_b(a^*b) = \varepsilon,$$

потому что если первой буквой сразу была b , то слово уже полностью завершено.

И

$$\text{nullable}(a^*b) = \text{false}.$$

Теперь можно вывести первое настоящее правило.

Для суммы языков:

$$D_a(R + S) = D_a(R) + D_a(S).$$

Это очевидно: слово aw принадлежит $R + S$, если оно принадлежит либо R , либо S .

А вот для конкатенации появляется первый интересный момент:

$$D_a(RS) = D_a(R)S + \begin{cases} D_a(S), & \text{если } R \text{ nullable,} \\ \emptyset, & \text{иначе.} \end{cases}$$

Почему? Потому что прочитанная буква a обычно съедается внутри R . Но если R умеет быть пустым, то она может оказаться уже первой буквой S .

Задача 3

Возьми

$$R = a^*, \quad S = b.$$

То есть снова $RS = a^*b$.

Попробуй теперь получить

$$D_a(a^*b)$$

не по смыслу языка, а именно по правилу произведения.

Здесь важно заметить, что a^* nullable.

10/4/2026, 8:48:25 PM · gpt-5-6-thinking





$$D_a(\text{ast } b) = D_a(\text{ast})b + D_a(b) = \text{ast } b + \text{пусто} = \text{ast } b$$

10/4/2026, 8:59:28 PM



Да, именно:

$$D_a(a^*b) = D_a(a^*)b + D_a(b).$$

Поскольку

$$D_a(a^*) = a^*, \quad D_a(b) = \emptyset,$$

получаем

$$\boxed{D_a(a^*b) = a^*b}.$$

Теперь очень похожая проверка, где второй член уже **не исчезнет**.

Возьми

$$R = a^*, \quad S = a + b.$$

То есть

$$a^*(a + b).$$

Найди

$$D_a(a^*(a + b))$$

по правилу произведения и упрости результат.

10/4/2026, 8:59:34 PM · gpt-5-6-thinking



$$Da(ast(a+b)) = Da(ast)(a+b) + Da(a+b) = ast(a+b) + \varepsilon = ast+ast\ b$$

10/4/2026, 9:04:02 PM



Да, всё верно.

$$D_a(a^*(a+b)) = a^*(a+b) + \varepsilon.$$

И твоё упрощение тоже правильное:

$$a^*(a+b) + \varepsilon = a^*a + a^*b + \varepsilon = \boxed{a^* + a^*b}.$$

Здесь как раз сработал второй член правила произведения: поскольку a^* nullable, прочитанная a могла принадлежать либо самому a^* , либо уже следующему $(a+b)$.

Теперь последнее базовое правило:

$$\boxed{D_a(R^*) = D_a(R)R^*}.$$

Смысл простой: если слово из R^* начинается с a , то a съедается в первом непустом экземпляре R ; после остатка этого экземпляра могут идти ещё сколько угодно R .

Задача 4

Пусть

$$R = (ab)^*.$$

Найди

$$D_a(R) \quad \text{и} \quad D_b(R).$$

Здесь уже попробуй пользоваться правилом для звезды.

10/4/2026, 9:04:20 PM · gpt-5-6-thinking



$b(ab)^*$ и
Пустое множество

10/4/2026, 9:05:46 PM



Да, оба ответа верны:

$$D_a((ab)^*) = b(ab)^*,$$

потому что

$$D_a(ab) = b,$$

а значит по правилу звезды

$$D_a((ab)^*) = D_a(ab)(ab)^* = b(ab)^*.$$

И

$$D_b((ab)^*) = \emptyset,$$

потому что ab не может начинаться с b :

$$D_b(ab) = \emptyset.$$

Теперь можно сделать следующий красивый шаг: увидеть, как из производных возникает автомат.

Для

$$R = (ab)^*$$

после чтения a мы попадаем в состояние

$$b(ab)^*.$$

После этого чтение b должно вернуть нас обратно в

$$(ab)^*.$$

А неправильная буква ведёт в $\emptyset$.

Задача 5

Найди все производные, которые реально возникают из $R = (ab)^*$ при чтении любых слов, и опиши переходы по буквам a, b .

И отдельно: какие из этих состояний являются принимающими? Подсказка: состояние принимающее тогда и только тогда, когда соответствующий остаточный язык nullable.

10/4/2026, 9:05:51 PM · gpt-5-6-thinking





$b(ab)^*$,
 $(ab)^*$ - принимающее
 Φ (пустое множество)

10/4/2026, 9:08:42 PM



Да. Это ровно три состояния:

$$A = (ab)^*, \quad B = b(ab)^*, \quad \emptyset.$$

Переходы:

$$\begin{aligned} A &\xrightarrow{a} B, & A &\xrightarrow{b} \emptyset, \\ B &\xrightarrow{b} A, & B &\xrightarrow{a} \emptyset, \\ \emptyset &\xrightarrow{a} \emptyset, & \emptyset &\xrightarrow{b} \emptyset. \end{aligned}$$

Принимающее только

$$A = (ab)^*,$$

потому что оно nullable. B и $\emptyset$ — нет.

И здесь появляется важная идея:

состояние DFA = остаточный язык после уже прочитанного префикса.

То есть производные не просто «похожи на автомат» — они буквально **строят его состояния**.

Задача 6

Возьмём чуть менее тривиальное выражение

$$R = (a + b)^*ab.$$

Это язык всех слов над $\{a, b\}$, оканчивающихся на ab .

Найди сначала только две производные:

$$D_a(R), \quad D_b(R).$$

Не надо пока строить весь автомат. Здесь интересно увидеть, как правило произведения работает из-за nullable у $(a + b)^*$.

10/4/2026, 9:08:50 PM · gpt-5-6-thinking



Da = R+b, Db = R

10/4/2026, 9:11:34 PM



Да, оба ответа правильны:

$$D_a(R) = R + b, \quad D_b(R) = R,$$

где

$$R = (a + b)^* ab.$$

Разбор первого:

$$D_a((a + b)^* ab) = D_a((a + b)^*) ab + D_a(ab).$$

Поскольку

$$D_a((a + b)^*) = (a + b)^*, \quad D_a(ab) = b,$$

получаем

$$\boxed{D_a(R) = R + b}.$$

А для b :

$$D_b(R) = D_b((a + b)^*) ab + D_b(ab) = R + \emptyset = \boxed{R}.$$

И здесь уже начинает проявляться смысл состояний: после прочитанной a мы «на полпути» к окончанию ab , поэтому появляется дополнительная возможность b .

Задача 7

Теперь возьмём новое состояние

$$Q = R + b.$$

Найди

$$D_a(Q), \quad D_b(Q),$$

и отметь, nullable ли получившиеся состояния.

После этого у нас почти автоматически получится весь минимальный автомат для слов, оканчивающихся на ab .

10/4/2026, 9:11:47 PM · gpt-5-6-thinking





Da(Q) = Q, Db(Q) = R + ε = R

10/4/2026, 9:14:07 PM



Почти: первая производная верна,

$$D_a(Q) = D_a(R + b) = D_a(R) + D_a(b) = Q + \emptyset = \boxed{Q}.$$

А вот во второй есть важная ошибка:

$$D_b(Q) = D_b(R) + D_b(b) = R + \varepsilon.$$

И

$$\boxed{R + \varepsilon \neq R},$$

потому что $R = (a + b)^*ab$ **не содержит пустого слова**, а $R + \varepsilon$ содержит. Обозначим новое состояние

$$P = R + \varepsilon.$$

Это как раз принимающее состояние: прочитанное до сих пор слово оканчивается на ab .

Теперь маленькое завершение автомата: найди

$$D_a(P), \quad D_b(P).$$

После этого у нас будут все три состояния R, Q, P .

10/4/2026, 9:14:19 PM · gpt-5-6-thinking



Da(P) = Q, Db(P) = R

10/4/2026, 9:15:39 PM



Да, верно:

$$D_a(P) = Q, \quad D_b(P) = R,$$

потому что $D_a(\varepsilon) = D_b(\varepsilon) = \emptyset$.

И теперь весь автомат замкнулся на трёх состояниях:

$$R = (a + b)^*ab, \quad Q = R + b, \quad P = R + \varepsilon.$$

Переходы:

	a	b
R	Q	R
Q	Q	P
P	Q	R

Принимающее состояние только P , потому что только оно nullable.

Это очень красиво читается без regex:

- R : пока полезного суффикса нет;
- Q : текущий суффикс — a ;
- P : текущий суффикс — ab , то есть слово сейчас принимается.

Задача 8

Теперь производная уже не по одной букве, а по слову. Определим

$$D_{uv}(R) = D_v(D_u(R)).$$

Для нашего языка слов, оканчивающихся на ab , найди состояния после префиксов

$$a, \quad aa, \quad aba, \quad abab.$$

И затем сформулируй своими словами: **что вообще означает $D_w(R)$ для произвольного уже прочитанного слова w ?**

10/4/2026, 9:15:46 PM · gpt-5-6-thinking



Q, Q, Q, P. Состояние соответствующее w (?)

10/4/2026, 9:20:44 PM



Да:

$$a \rightarrow Q, \quad aa \rightarrow Q, \quad aba \rightarrow Q, \quad abab \rightarrow P.$$

И формулировку стоит сделать чуть точнее:

$$D_w(R) = \{ z : wz \in L(R) \}.$$

То есть $D_w(R)$ — не просто «состояние, соответствующее w », а

остаточный язык после того, как префикс w уже прочитан.

Именно этот остаточный язык мы и считаем состоянием автомата.

Например, после $w = aba$ мы находимся в Q , потому что текущее слово заканчивается на a : чтобы попасть в язык, можно дописать b , либо ещё что-то, а потом закончить на ab .

А после $w = abab$ мы в P , и P nullable: можно ничего больше не дописывать, потому что $abab$ уже оканчивается на ab .

Отсюда очень красивый критерий:

$$w \in L(R) \iff D_w(R) \text{ nullable.}$$

Это, пожалуй, центральная формула всего листка 10.

Следующий шаг я бы сделал концептуальный: два слова u, v приводят в одно состояние тогда и только тогда, когда

$$D_u(R) = D_v(R),$$

то есть после них допустимы **ровно те же продолжения**. Это уже прямой мост к отношению Майхилла—Нероуда и минимальности автомата.

10/4/2026, 9:20:50 PM · gpt-5-6-thinking



Может не стоит уходить в теорию автоматов. Это отдельная тема. Мой вопрос про производные регулярных выражений был связан с производными типов данных и перечислениями. Пока особых связей я не вижу. Если так, то тему можно пропустить.

10/4/2026, 9:24:40 PM



Да, я согласен. Здесь связь оказалась гораздо слабее, чем я первоначально надеялся использовать педагогически. Производная типа/species и производная Бржозовского называются «производными» не случайно: обе подчиняются похожей алгебре — сумма, произведение, цепочка структур и т. п. Но смысл принципиально различный:

F' — структура с одной выделенной/вынутой атомной позицией;

$D_a R$ — остаточный язык после поглощения символа a .

Первая непосредственно связана с **контекстами, zipper'ами и перечислением**. Вторая — прежде всего с **распознаванием языков и автоматами**. Чтобы получить между ними действительно содержательный мост, пришлось бы уйти довольно далеко в алгебру типов/формальных языков, а для нашей нынешней линии это боковая ветвь.

Так что **Листок 10 закрываем здесь**. Мы получили достаточно, чтобы знать: «да, существует ещё один дифференциальный calculus для регулярных выражений, но это другая производная».

Я бы теперь немного изменил первоначальный план. Отдельный листок про перестановки и циклы нам тоже почти не нужен: маркеры циклов, $P(x, u) = (1 - x)^{-u}$, производные по u , среднее H_n и дисперсию мы уже хорошо разобрали в Листке 7.

Поэтому следующий полезный этап — **синтетический листок**: я буду давать комбинаторный класс без подсказки, а ты будешь сам решать, нужны ли `SET`, `SEQ`, `CYC`, композиция, маркер, производная/pointing и OGF или EGF. Это проверит, сложилась ли вся система в единый язык. А уже после него — обещанный отдельный листок **Burnside-Pólya / cycle index**, который соединит наши разговоры об автоморфизмах, labeled/unlabeled и факторизации по симметриям.

То есть я предлагаю: **следующий — синтез**, затем **Burnside-Pólya**.

10/4/2026, 9:24:52 PM · gpt-5-6-thinking





То есть, осталось два листка? Можешь дать общий список листков (пройденных и оставшихся)? Просто номер и тема.

10/4/2026, 9:27:21 PM



Да, по текущему плану осталось **два листка**.

1. **Списки: рекурсивные типы, производящие функции и производная**
2. **Деревья Каталана: OGF, EGF и производная**
3. **Помеченные корневые деревья**
4. **Разбиения множества: Bell, Stirling и производная**
5. **Endofunctions**
6. **Повторение и перенос идей: species, SET/SEQ/CYC, производные**
7. **Маркеры и частные производные**
8. **Композиция species, Bell polynomials, Faà di Bruno**
9. **Атомная дырка, рекурсивная дырка и zipper**
10. **Производные регулярных выражений Бржозовского** - сокращённый листок
11. **Синтез: самостоятельно распознавать конструкции и выбирать инструменты** - остался
12. **Burnside-Pólya и cycle index series** - остался

Отдельный первоначально намечавшийся листок про **перестановки и циклы** фактически растворился в листке 7, поэтому отдельным номером его уже нет.

10/4/2026, 9:27:31 PM · gpt-5-6-thinking





Хорошо. Тогда, если можешь, собери те ключевые моменты, которые я не понимал в силу того, что явно они не были проговорены. Первый такой момент, если я не ошибаюсь, это то, что мы считаем атомы (как в $T = X + T^2$). Один из последних: разница между id и value. Их всего несколько (4-6).

10/4/2026, 9:35:51 PM



Да. Я бы выделил **шесть** таких моментов. Это именно не «трудные теоремы», а скрытые соглашения языка species, которые я поначалу недостаточно явно проговаривал.

1. **X означает выбранный нами атом размера.**

В

$$T = X + T^2$$

X - не «метка» и не просто переменная: это то, что мы решили считать единицей размера. Для каталановского дерева это были листья. Можно выбрать другую атомизацию - например, считать все вершины, - и тогда изменятся и GF, и смысл производной. Отсюда же важное: «забыть метки» не значит положить $X = 1$. X продолжает учитывать размер.

2. **Species строится на фиксированном множестве атомов, и каждый атом используется ровно один раз.**

Если $U = \{1, 2, 3\}$, то $\text{SEQ}[U]$ - это $3!$ порядков именно этих трёх объектов. Нельзя получить $(1, 1, 2)$. Это не специальное свойство SEQ ; это базовая модель species:

взяли множество U и организуем все его элементы.

Я слишком долго пользовался этим молча.

3. **Identity и value - разные вещи.**

Метки 1, 2, 3 - это прежде всего **идентичности объектов**, а не их значения. Два разных атома вполне могут иметь одинаковый payload:

$$a \neq b, \quad \text{value}(a) = \text{value}(b) = \text{red}.$$

Поэтому «дубликаты значений» допустимы; нельзя лишь дважды использовать один и тот же атом. Это сняло твоё программистское недоумение насчёт списков и блоков.

4. **При произведении и композиции исходное множество атомов разбивается; новые компоненты не создают новый размер.**

Для произведения

$$(FG)[U]$$

сначала

$$U = A \sqcup B,$$

затем строятся $F[A]$ и $G[B]$.

Для композиции $F \circ G$ множество U разбивается на блоки, на каждом строится G -структура, а блоки становятся объектами для внешней F -структуры.

Но это «атомы внешнего уровня» лишь образно. Если исходно было три атома, **весь объект всё равно имеет размер 3**, даже если получилось один, два или три блока. Поэтому всё идёт при x^3 , а число блоков при желании надо считать отдельным маркером u .

5. **SET можно понимать как «упорядочили, затем забыли порядок», но надо различать два вида симметрии.**

Рабочая картинка:

$$\text{SET}_k \sim \text{SEQ}_k / S_k.$$

Поэтому возникают $1/k!$. В $\text{SET}(\text{SET})$ симметрия есть и внутри блоков, и между блоками одинакового типа; в $\text{SEQ}(\text{SET})$ внешний порядок сохраняется.

Но полезно различать:

- группу S_k , **по которой мы факторизуем позиции**;
- группу автоморфизмов **конкретной структуры**, то есть её стабилизатор.

Именно здесь же появляется формула

$$\# \text{labelings of shape} = \frac{n!}{|\text{Aut}|},$$

и EGF-вес формы $1/|\text{Aut}|$.

6. Производная имеет жёсткое определение, а “дырка/zipper” - представление; кроме того, есть два разных вида дырки.

Обычная производная species:

$$F'[U] = F[U \sqcup \{*\}],$$

то есть один **атом** X стал специальным и перестал учитываться размером. Например,

$$L' = L^2$$

- не определение производной списка, а удобная кодировка списка с одной атомной дыркой.

Совсем другая операция возникает для рекурсивного конструктора

$$T = \Phi(X, T).$$

Производная

$$\Phi_Y$$

описывает **один frame - дырку вместо целого рекурсивного поддерева**. Последовательность frames даёт zipper-контекст. А обычная атомная производная раскладывается как

$$T' = \text{SEQ}(\Phi_Y) \Phi_X$$

в наших простых рекурсивных случаях. Это наконец объяснило, почему иногда «производная = zipper», а иногда нет.

Я бы сказал, что именно эти шесть составляют наш маленький «**словарь скрытых соглашений species**». После того как они были произнесены вслух, значительная часть прежних загадок действительно перестала быть загадочной.

10/4/2026, 9:36:04 PM · gpt-5-6-thinking



7. Суть разницы между OGF и EGF. Как работает счёт и EGF на метках. Собственно, ты говоришь об этом в 4, но не совсем чётко.

10/4/2026, 9:41:24 PM



Да. Это действительно отдельный пункт, и, пожалуй, один из самых важных.

7. **OGF и EGF отличаются не “тем, что одно для форм, другое для меток”, а способом учитывать распределение меток.**

Если

$$A(x) = \sum a_n x^n$$

— OGF, то коэффициент a_n просто считает объекты размера n .

Если

$$A(x) = \sum a_n \frac{x^n}{n!}$$

— EGF, то a_n — число структур на **фиксированном множестве из n различных меток**.

Деление на $n!$ нужно не ради красоты: оно делает произведение EGF совместимым с разбиением множества меток между компонентами.

Для

$$H = FG$$

на n метках надо выбрать, какие k меток пойдут в F :

$$\binom{n}{k}.$$

Поэтому

$$h_n = \sum_k \binom{n}{k} f_k g_{n-k}.$$

А в EGF эта биномиальная величина автоматически возникает из

$$\frac{x^k}{k!} \frac{x^{n-k}}{(n-k)!} = \binom{n}{k} \frac{x^n}{n!}.$$

То есть

EGF встроено умеет распределять различные метки между компонентами.

Для непомеченных форм такой выбор меток отсутствует, поэтому часто естественнее OGF.

И ещё один важный нюанс: одна непомеченная форма A размера n даёт

$$\frac{n!}{|\text{Aut } A|}$$

различных помеченных структур. После деления на $n!$ её вклад в EGF равен

$$\frac{1}{|\text{Aut } A|}.$$

Поэтому EGF можно также понимать как сумму по непомеченным формам с весом $1/|\text{Aut } A|$.

И да, это лучше держать отдельно от пункта 4. Пункт 4 — **как устроена композиция и что остаётся атомами размера**. Пункт 7 — **почему именно EGF правильно считает labeled-конструкции и откуда берутся**

биномиальные коэффициенты и веса автоморфизмов.

10/4/2026, 9:41:27 PM · gpt-5-6-thinking



Хорошо. Продолжим. Листок 11.

10/8/2026, 11:30:24 AM



Итак, **Листок 11. Синтез.**

Здесь идея другая: я не буду заранее говорить, какой именно инструмент нужен. Надо самому распознать структуру:

SET, SEQ, CYS, произведение, композиция, маркер, pointing, производная, OGF или EGF.

Будем идти по одной задаче.

Задача 1

Есть класс структур:

конечное множество непустых линейно упорядоченных блоков на помеченных атомах.

То есть исходное множество меток разбивается на непустые блоки, внутри каждого блока задан линейный порядок, а сами блоки между собой не упорядочены.

Нужно:

1. записать species;
2. получить EGF;
3. объяснить, почему именно EGF здесь естественна;
4. сказать, что будет считать дополнительный маркер u , если поставить его на каждый внешний блок.

Пока без вычисления коэффициентов.

10/8/2026, 11:30:32 AM · gpt-5-6-thinking





1. $\text{SET}(\text{SEQ}(X))$, 2. $\exp(1/(1-x))$, 3. EGF естественна, поскольку наши атомы помечены, и, при подсчёте, важны не только форма, но и метки. 4. Маркер считает блоки (точнее, придает вес и каждому блоку).

10/8/2026, 11:34:40 AM



Почти всё верно; одна важная поправка в пп. 1-2.

Поскольку блоки **непустые**, надо писать

$$\text{SET}(\text{SEQ}_{\geq 1}(X))$$

а не $\text{SET}(\text{SEQ}(X))$.

Для непустой последовательности:

$$\text{SEQ}_{\geq 1}(X) = X + X^2 + X^3 + \dots,$$

и её EGF

$$\frac{x}{1-x}.$$

Поэтому весь класс имеет EGF

$$\exp\left(\frac{x}{1-x}\right).$$

Если бы мы взяли $\text{SEQ}(X)$, туда входила бы пустая последовательность, и внешний SET позволял бы добавлять сколько угодно пустых блоков — ровно та проблема с $G(0) \neq 0$, которую мы недавно разбирали.

П. 3 у тебя по существу верен. Я бы уточнил: EGF естественна не просто потому, что «метки важны», а потому, что при разбиении исходного множества меток между блоками возникают биномиальные/мультиномиальные коэффициенты, и EGF их автоматически кодирует.

П. 4 тоже верен. С маркером u :

$$\exp\left(u \frac{x}{1-x}\right)$$

и степень u^k отмечает структуры с k блоками.

Задача 2

Теперь похожий, но другой класс:

помеченные атомы разбиты на непустые циклически упорядоченные блоки; сами блоки не упорядочены.

Нужно снова определить species и EGF. Здесь главный вопрос: чем заменить внутренний $\text{SEQ}_{\geq 1}$?

10/8/2026, 11:34:46 AM · gpt-5-6-thinking





Да, я пропустил слово непустые. 1 $\text{SET}(\text{CYC}(X), \geq 1)$, 2 $\exp(-\log(1-x)) = 1/(1-x)$

10/8/2026, 11:38:42 AM



Да. И здесь даже проще: $\text{CYC}(X)$ уже по определению непустой, так что достаточно

$$\boxed{\text{SET}(\text{CYC}(X))}.$$

EGF:

$$\text{CYC}(X) \longleftrightarrow -\log(1-x),$$

поэтому

$$\text{SET}(\text{CYC}(X)) \longleftrightarrow \exp(-\log(1-x)) = \boxed{\frac{1}{1-x}}.$$

И мы узнали старого знакомого: это просто **перестановки**, потому что любая перестановка единственным образом раскладывается в неупорядоченное множество непересекающихся циклов.

Задача 3

Теперь класс:

*разбиение помеченного множества на непустые блоки, причём **один блок выделен**.*

Нужно:

1. записать species;
2. получить EGF;
3. связать её с производной по маркеру u в

$$\Pi(x, u) = \exp(u(e^x - 1)).$$

Здесь попробуй решить сразу двумя взглядами: «выделенный блок + все остальные блоки» и через $\partial/\partial u$.

10/8/2026, 11:38:50 AM · gpt-5-6-thinking



1. Как я понимаю, имеется в виду множество множеств. $\text{SET}(X, \geq 1) \times \text{SET}(\text{SET}(X, \geq 1), \geq 0)$, 2. $(\exp(x)-1) \times \exp(\exp(x)-1)$, 3 $\Pi'(x, y)$, $u=1 = \exp(\exp(x)-1) \times (\exp(x)-1)$ - то же самое. В самом деле, производная по u - это выделение одного блока из множества непустых множеств/блоков.

10/8/2026, 11:45:52 AM



Да, всё правильно.

В species-нотации я бы записал чуть компактнее:

$$\text{SET}_{\geq 1}(X) \text{ SET}(\text{SET}_{\geq 1}(X))$$

— первый множитель есть выделенный блок, второй — множество всех остальных блоков, которое может быть пустым.

EGF:

$$(e^x - 1) \exp(e^x - 1).$$

А через маркер:

$$\begin{aligned} \Pi(x, u) &= \exp(u(e^x - 1)), \\ \frac{\partial \Pi}{\partial u} &= (e^x - 1) \exp(u(e^x - 1)), \end{aligned}$$

поэтому

$$\Pi_u(x, u)|_{u=1} = (e^x - 1) \exp(e^x - 1).$$

То есть два описания действительно совпали:

$$\text{выделенный блок} + \text{остальные блоки}$$

и

$$\partial_u = \text{выделить один из блоков}.$$

Задача 4

Теперь очень похожая формулировка, но выделяем **не блок, а один атом**:

разбиение помеченного множества, в котором один атом выделен.

Нужно получить EGF двумя способами:

1. непосредственно: что происходит с блоком, содержащим выделенный атом, и что происходит с остальными блоками;
2. через обычную производную $\Pi'(x)$ и pointing.

Главное здесь — не спутать

$$\Pi'(x)$$

с

$$x\Pi'(x).$$

Какой из них соответствует выделенному **обычному атому, который продолжает учитываться в размере?**

10/8/2026, 11:46:02 AM · gpt-5-6-thinking





1. Исходно у нас $\text{SET}(\text{SET}(X, \geq 1))$. Мы выделяем атом, то есть, мы выделяем блок, а в нём атом, который продолжает учитываться в размере, то есть дырку надо заполнить. Получаем: $\text{SET}(\text{SET}(X, \geq 1) \times \text{SET}(X, \geq 1) \times X)$. EGF [формальный перевод]: $\exp(\exp(x)-1) \times (\exp(x)-1) \times x$ 2. Производная Π' и pointing: $x\Pi'(x) = x \times \exp(\exp(x)-1) \times (\exp(x)-1)$. $x\Pi'(x)$ - выделение атома, который продолжает учитываться в размере.

10/8/2026, 11:52:42 AM



Почти, но здесь как раз важная ошибка: блок, содержащий выделенный атом, после выделения распадается не на

$$X \times \text{SET}_{\geq 1}(X),$$

а на

$$X \times \text{SET}(X).$$

Почему? Потому что выделенный атом может быть **единственным** в своём блоке. После того как мы вынесли его отдельно, остальных атомов в этом блоке может быть $0, 1, 2, \dots$

Поэтому прямое описание:

$$X \cdot \text{SET}(X) \cdot \text{SET}(\text{SET}_{\geq 1}(X))$$

То есть:

- X — выделенный атом, который продолжаем считать;
- $\text{SET}(X)$ — остальные атомы его блока, возможно ни одного;
- $\text{SET}(\text{SET}_{\geq 1}(X))$ — все остальные блоки.

EGF:

$$x e^x \exp(e^x - 1).$$

Теперь через производную:

$$\Pi(x) = \exp(e^x - 1),$$

$$\Pi'(x) = e^x \exp(e^x - 1),$$

а pointing:

$$x\Pi'(x) = x e^x \exp(e^x - 1).$$

Совпало.

И это хороший маленький урок:

$$\partial_x$$

не «выделяет блок, затем атом в нём». Она выделяет атом напрямую; а когда мы смотрим на его блок, **остаток блока может быть пустым**.

10/8/2026, 11:52:50 AM · gpt-5-6-thinking





Да, и я неправильно посчитал производную, иначе бы поправил.

10/8/2026, 11:55:29 AM



Да, именно. Если бы сразу продифференцировать

$$\Pi(x) = \exp(e^x - 1),$$

то

$$\Pi'(x) = e^x \exp(e^x - 1)$$

сразу подсказало бы, что «остаток выделенного блока» должен давать $e^x = \text{SET}(X)$, а не $e^x - 1$.

Хорошая самопроверка: **аналитическая производная и структурное описание должны совпасть**. Если не совпадают, почти наверняка где-то неверно учтена пустая возможность.

Задача 5

Теперь класс:

корневое помеченное дерево, в котором дополнительно выделена ещё одна вершина; она может совпадать с корнем.

Напомню:

$$T = X \cdot \text{SET}(T), \quad T(x) = xe^{T(x)}.$$

Нужно:

1. записать species/операцию, которая соответствует «ещё одна выделенная вершина»;
2. получить EGF;
3. сказать, сколько таких структур на n метках, если обычных корневых деревьев n^{n-1} .

Здесь должно всплыть pointing.

10/8/2026, 11:55:41 AM · gpt-5-6-thinking





1. Выделенная вершина является корневой (n вариантов): $X \times \text{SET}(T)$. Плюс, выделенная вершина где-то в другом месте и мы выбираем дырку в поддереве: $X^2 \times T'(X) \times \text{SET}(T)$ 2. Дифференцируем и умножаем на x : $x \times T'(x) = x \times \exp(T) + x^2 \times \exp(T) \times T'(x)$. 3 n^n

10/8/2026, 12:05:53 PM



Да, по существу всё правильно. Особенно хорошо, что ты не просто написал pointing XT' , а разложил его по положению выделенной вершины.

Только одна маленькая поправка к словам: если дополнительная выделенная вершина **совпадает с уже заданным корнем**, то для фиксированного корневого дерева это **один** вариант, не n . n вариантов — это общее число возможных положений дополнительной отметки.

Структурно ты получил:

$$XT' = X \text{SET}(T) + X^2 T' \text{SET}(T).$$

Первое слагаемое: выделена корневая вершина.

Второе: выделенная вершина находится в одном из поддеревьев; мы выбираем это поддерево с дыркой T' , а остальные дети корня остаются обычным $\text{SET}(T)$.

Аналитически из

$$T = x e^T$$

имеем

$$T' = e^T + x e^T T',$$

поэтому

$$xT' = x e^T + x^2 e^T T'.$$

И по счёту:

$$n^{n-1} \times n = \boxed{n^n}.$$

То есть да: XT' — корневое помеченное дерево с ещё одной выделенной вершиной, возможно совпадающей с корнем.

Задача 6

Теперь без подсказки инструмента:

помеченное множество разбито на циклы, и **один цикл выделен**.

Нужно:

1. записать species;
2. получить EGF;
3. связать это с маркером u числа циклов в

$$P(x, u) = (1 - x)^{-u}.$$

Здесь должно получиться то же самое двумя путями.

10/8/2026, 12:06:06 PM · gpt-5-6-thinking





1. $CYC(X) \times SET(CYC(X))$ 2. Формально: $-\log(1-x)/(1-x)$, 3. $dP/du, u=1$: $-\log(1-x)/(1-x)$ - то же самое.

10/8/2026, 12:12:30 PM



Да, всё правильно:

$$CYC(X) \cdot SET(CYC(X))$$

— один выделенный цикл плюс неупорядоченное множество остальных циклов.

EGF:

$$-\log(1-x) \cdot \frac{1}{1-x}.$$

И через маркер:

$$P(x, u) = (1-x)^{-u},$$

$$\frac{\partial P}{\partial u} = -\log(1-x)(1-x)^{-u},$$

поэтому при $u = 1$:

$$P_u|_{u=1} = \frac{-\log(1-x)}{1-x}.$$

Точно совпало.

Задача 7

Теперь специально проверим различие **OGF vs EGF**.

Полное плоское бинарное дерево:

$$B = X + B^2,$$

где атомами считаются листья.

Для **непомеченных форм** мы раньше получили OGF

$$B(x) = x + B(x)^2.$$

Теперь представим, что все листья имеют различные метки.

1. Какую производящую функцию естественно использовать теперь?
2. Какое структурное уравнение она удовлетворяет?
3. Получится ли формально та же функция $B(x)$?
4. Если коэффициент при x^n в старой OGF равен C_{n-1} , сколько теперь **помеченных** деревьев на фиксированном наборе из n меток?

Здесь небольшой подвох: формула ряда останется той же, а смысл коэффициентов — нет.

10/8/2026, 12:12:40 PM · gpt-5-6-thinking





1 EGF, 2 то же самое, 3 формально да, но теперь все коэффициенты ряда умножаются и делятся на $n!$, то есть счёт меняется - умножается на $n!$. 4. Из 3: $n! \times C_{n-1}$

10/8/2026, 12:17:38 PM



Да, всё верно.

1. Для помеченных листьев естественна **EGF**.
2. Структурное уравнение species остаётся тем же:

$$B = X + B^2.$$

3. И формально ряд действительно удовлетворяет тому же уравнению:

$$B(x) = x + B(x)^2.$$

Поэтому как формальный ряд он совпадает со старой каталановской OGF.

Но смысл коэффициента другой:

- в OGF коэффициент при x^n — число помеченных форм;
- в EGF коэффициент при $x^n/n!$ — число помеченных структур.

То есть если

$$[x^n]B(x) = C_{n-1},$$

то в labeled-интерпретации

$$\frac{b_n}{n!} = C_{n-1},$$

откуда

$$b_n = n! C_{n-1}.$$

Это как раз один из тех случаев, где **одна и та же формальная функция скрывает два разных комбинаторных смысла**.

Задача 8

Последняя содержательная задача этого листка.

Есть класс помеченных структур:

неупорядоченное множество непустых линейных списков, причём в каждом списке один атом выделен.

Попробуй сам решить без подсказки:

1. species;
2. EGF;
3. если внутри каждого списка выделенный атом продолжает учитываться в размере, какой оператор/конструкция здесь естественно появляется?

Здесь надо совместить **SET**, **SEQ** и pointing.

10/8/2026, 12:17:46 PM · gpt-5-6-thinking



1. Исходная структура (без выделения): $\text{SET}(\text{SEQ}(X, \geq 1))$. С выделением (выбираем список, место и атом): $\text{SET}(\text{SEQ}(X, \geq 1)) \times \text{SEQ}'(X) \times X$ или $\text{SET}(\text{SEQ}(X), \geq 1) \times \text{SEQ}(X) \times \text{SEQ}(X) \times X$.
2. EGF: $[\exp(1/(1-x)-1)]' \times x = \exp(x/(1-x))/(1-x)^2 \times x$ [или формально из 1 - то же самое.]
3. Pointing - см выше: находим производную исходной структуры по X и умножаем на X .

10/8/2026, 12:28:28 PM



Здесь как раз обнаружилась **важная неоднозначность**, и мой текст задачи её спровоцировал.

Я имел в виду:

в каждом списке выделен один атом.

А ты решил другую, тоже совершенно естественную задачу:

во всём множестве списков выделен **один атом**, а значит автоматически выделен и один список.

Для твоей интерпретации решение по существу правильное:

$$F = \text{SET}(\text{SEQ}_{\geq 1}(X)),$$

и pointing всей структуры даёт

$$XF'$$

с EGF

$$x \frac{d}{dx} \exp\left(\frac{x}{1-x}\right) = \frac{x}{(1-x)^2} \exp\left(\frac{x}{1-x}\right).$$

Структурно это

$$\text{SET}(\text{SEQ}_{\geq 1}(X)) \cdot X \cdot \text{SEQ}(X)^2$$

— остальные списки плюс один особый список с выделенным атомом.

Но моя задуманная задача была другой. Сначала сделаем **один pointed list**:

$$A = \text{SEQ}_{\geq 1}(X).$$

В нём один атом выделен и продолжает считаться:

$$XA'$$

Поскольку

$$A' = \text{SEQ}(X)^2,$$

получаем

$$P = X \text{SEQ}(X)^2.$$

Его EGF:

$$P(x) = \frac{x}{(1-x)^2}.$$

А теперь берём **неупорядоченное множество таких уже-pointed списков**:



$$\text{SET}(X \text{ SEQ}(X)^2).$$

Следовательно EGF:

$$\exp\left(\frac{x}{(1-x)^2}\right).$$

Разница очень содержательная:

$$X \frac{d}{dx} \text{SET}(A)$$

= выделить **один атом во всей большой структуре**;

тогда как

$$\text{SET}(X A')$$

= **в каждой компоненте** A независимо выделить по одному атому.

И это, пожалуй, отличный финальный вопрос Листка 11: порядок операций `SET` и pointing имеет совершенно разный комбинаторный смысл.

10/8/2026, 12:28:40 PM · gpt-5-6-thinking



Да, это я промахнулся- читать надо внимательней: ((

10/8/2026, 12:30:15 PM



Да, тут именно чтение условия, не математика. Причём твоя интерпретация была вполне содержательной и дала полезное сравнение двух разных операций, так что промах даже оказался педагогически удачным.

Я бы на этом **Листок 11 завершил**. По ощущениям, синтез уже работает: ты уверенно распознаёшь `SET/SEQ/CYC`, маркеры, pointing, производные и различаешь OGF/EGF. Остались в основном локальные оговорки вроде непустоты и точного уровня, на котором ставится отметка.

Следующий и последний — **Листок 12: Burnside-Pólya и cycle index series**. Там как раз соберём в одну картину симметрии, автоморфизмы, labeled/unlabeled и переход от species к подсчёту орбит.

10/8/2026, 12:30:20 PM · gpt-5-6-thinking



Хорошо. Продолжим.

10/8/2026, 12:30:37 PM



Тогда начинаем **Листок 12. Burnside-Pólya и cycle index series**.

Здесь я бы снова дал короткое предисловие, потому что это новый язык, хотя почти все ингредиенты нам уже знакомы.

Главная идея Бернсайда:

$$\# \text{орбит} = \frac{1}{|G|} \sum_{g \in G} \# \text{Fix}(g)$$

То есть, чтобы посчитать объекты **с точностью до симметрии**, мы усредняем число объектов, неподвижных при каждой симметрии.

Cycle index просто запоминает чуть больше: **какого циклического типа была перестановка**, которая что-то фиксирует.

Для $\sigma \in S_n$ обозначим через $c_j(\sigma)$ число j -циклов. Тогда ей соответствует моном

$$p_1^{c_1} p_2^{c_2} p_3^{c_3} \cdots$$

Например, в S_3 :

- тождество $(1)(2)(3)$ даёт p_1^3 ;
- транспозиция $(12)(3)$ даёт $p_1 p_2$;
- 3-цикл (123) даёт p_3 .

Задача 1. Самый маленький cycle index

Возьмём species **SET**.

На фиксированном множестве $\{1, 2, 3\}$ существует ровно одна **SET**-структура, и **любая** перестановка меток её сохраняет.

В S_3 есть:

1

тождественная перестановка,

3

транспозиции,

2

трёхцикла.

Поэтому степень 3 в cycle index должна получиться усреднением по этим шести перестановкам.

Попробуй сам записать:

$$Z_{\text{SET},3}(p_1, p_2, p_3) = ?$$

То есть просто соответствующий многочлен с коэффициентом $1/3!$.

10/8/2026, 12:30:52 PM · gpt-5-6-thinking



$$1/6 \times (p_1^3 + 3 \times p_1 \times p_2 + 2 \times p_3)$$

10/8/2026, 12:49:15 PM



Да, совершенно верно:

$$Z_{\text{SET},3} = \frac{1}{6} (p_1^3 + 3p_1p_2 + 2p_3) .$$

Теперь сразу увидим, зачем эта штука нужна.

Задача 2. Из одного cycle index получить EGF и OGF

Сделай две разные подстановки.

(a) Помеченный счёт: положи

$$p_1 = x, \quad p_2 = p_3 = \dots = 0.$$

Что получится из нашего $Z_{\text{SET},3}$?

(b) Непомеченный счёт: положи

$$p_1 = x, \quad p_2 = x^2, \quad p_3 = x^3.$$

Что получится теперь?

Попробуй затем объяснить, почему в первом случае выживает только тождественная перестановка, а во втором все шесть перестановок дают вклад x^3 .

10/8/2026, 12:49:25 PM · gpt-5-6-thinking





а) $1/6 x^3$, б) x^3 ; в первом случае нетождественные перестановки обнуляются, поскольку содержат хотя бы один цикл длиной > 1 ; все мономы гомогенны если вместо p_n подставить x^n (в результате получается произведение степеней, соответствующих длинам цикла по всем циклам, то есть, по всем n элементам, то есть x^n)

10/8/2026, 12:54:08 PM



Да, всё именно так. Особенно хороша последняя формулировка:

$$p_1^{c_1} p_2^{c_2} \dots \xrightarrow{p_j = x^j} x^{c_1 + 2c_2 + 3c_3 + \dots} = x^n,$$

потому что сумма длин всех циклов равна n .

То есть две подстановки действительно извлекают из одного более богатого объекта две разные информации:

$$\begin{aligned} Z_F(x, 0, 0, \dots) &\rightsquigarrow \text{labeled EGF,} \\ Z_F(x, x^2, x^3, \dots) &\rightsquigarrow \text{unlabeled OGF.} \end{aligned}$$

Для **SET** мы получили на уровне $n = 3$:

$$\frac{x^3}{3!} \quad \text{и} \quad x^3,$$

как и должны: одна **SET**-структура на фиксированных трёх метках и одна непомеченная форма размера 3.

Задача 3. Получить весь cycle index **SET**

Теперь попробуем вывести

$$Z_{\text{SET}} = \exp \left(p_1 + \frac{p_2}{2} + \frac{p_3}{3} + \dots \right),$$

а не просто принять формулу.

Пусть перестановка имеет

$$m_1 \text{ циклов длины } 1, \quad m_2 \text{ циклов длины } 2, \quad \dots$$

так что

$$\sum_j j m_j = n.$$

Мы уже встречали похожую формулу. Число перестановок такого циклического типа равно

$$\frac{n!}{\prod_j j^{m_j} m_j!}.$$

В cycle index мы ещё делим на $n!$.

Вопрос: какой вклад даст один фиксированный набор

$$(m_1, m_2, \dots)$$

после этого деления?

А затем попробуй увидеть, почему суммирование по всем $m_j \geq 0$ должно разложиться в произведение экспонент.

10/8/2026, 12:54:19 PM · gpt-5-6-thinking



$1/\prod (j^{m_j} m_j!)$, не вижу без записи, а записать здесь сложно :(

10/8/2026, 12:59:36 PM



Да, первый шаг правильный:

$$\frac{1}{\prod_j j^{m_j} m_j!}.$$

С учётом монома это вклад

$$\prod_j \frac{p_j^{m_j}}{j^{m_j} m_j!}.$$

А дальше как раз можно увидеть без тяжёлой записи. Мы суммируем **независимо по каждому** $m_j = 0, 1, 2, \dots$, поэтому сумма по всем наборам $(m_1, m_2, \dots)$ распадается в произведение:

$$\prod_{j \geq 1} \left(\sum_{m_j \geq 0} \frac{1}{m_j!} \left(\frac{p_j}{j} \right)^{m_j} \right).$$

Но внутренняя сумма — просто экспонента:

$$\sum_{m \geq 0} \frac{(p_j/j)^m}{m!} = \exp \left(\frac{p_j}{j} \right).$$

Значит,

$$Z_{\text{SET}} = \prod_{j \geq 1} \exp \left(\frac{p_j}{j} \right) = \exp \left(p_1 + \frac{p_2}{2} + \frac{p_3}{3} + \dots \right).$$

То есть ключевая идея здесь та же, что у нас уже была много раз:

независимый выбор количества компонентов каждого типа $\Rightarrow$ произведение экспоненциальных рядов.

Если хочешь, следующий шаг можно сделать очень конкретным: из этой общей формулы снова получить сразу и e^x , и $1/(1-x)$, но уже не для $n = 3$, а целиком.

10/8/2026, 12:59:38 PM · gpt-5-6-thinking





Мне переход от суммы произведений к произведению сумм очевидным не кажется.

10/8/2026, 1:01:45 PM



Да, тут я слишком быстро проскочил. Это не специфическое свойство cycle index — это просто многократное раскрытие скобок.

Сначала всего два независимых индекса. Пусть

$$a_{m_1}, \quad b_{m_2}.$$

Тогда

$$\sum_{m_1 \geq 0} \sum_{m_2 \geq 0} a_{m_1} b_{m_2}$$

означает

$$a_0 b_0 + a_0 b_1 + a_0 b_2 + \dots + a_1 b_0 + a_1 b_1 + a_1 b_2 + \dots + \dots$$

А если взять

$$\left(\sum_{m_1 \geq 0} a_{m_1} \right) \left(\sum_{m_2 \geq 0} b_{m_2} \right),$$

то при раскрытии скобок получится **ровно та же таблица всех пар** (m_1, m_2) :

$$\begin{aligned} & (a_0 + a_1 + a_2 + \dots)(b_0 + b_1 + b_2 + \dots) \\ &= a_0 b_0 + a_0 b_1 + \dots + a_1 b_0 + a_1 b_1 + \dots \end{aligned}$$

Поэтому

$$\boxed{\sum_{m_1, m_2 \geq 0} a_{m_1} b_{m_2} = \left(\sum_{m_1 \geq 0} a_{m_1} \right) \left(\sum_{m_2 \geq 0} b_{m_2} \right).}$$

Теперь у нас

$$a_{m_1} = \frac{p_1^{m_1}}{1^{m_1} m_1!}, \quad b_{m_2} = \frac{p_2^{m_2}}{2^{m_2} m_2!},$$

и дальше аналогично для $m_3, m_4, \dots$

Например, только для циклов длин 1 и 2:

$$\sum_{m_1, m_2 \geq 0} \frac{p_1^{m_1}}{m_1!} \frac{(p_2/2)^{m_2}}{m_2!}$$

равно

$$\left(1 + p_1 + \frac{p_1^2}{2!} + \dots \right) \left(1 + \frac{p_2}{2} + \frac{(p_2/2)^2}{2!} + \dots \right).$$

Каждый выбор одного члена из первой скобки и одного из второй соответствует ровно одной паре

$$(m_1, m_2).$$

С тремя длинами циклов:

$$\sum_{m_1, m_2, m_3} a_{m_1} b_{m_2} c_{m_3} = \left(\sum a_{m_1} \right) \left(\sum b_{m_2} \right) \left(\sum c_{m_3} \right).$$

То есть в нашем случае

$$\sum_{m_1, m_2, \dots} \prod_j \frac{(p_j/j)^{m_j}}{m_j!}$$

превращается в

$$\prod_j \left(\sum_{m_j \geq 0} \frac{(p_j/j)^{m_j}}{m_j!} \right).$$

А каждая внутренняя сумма есть

$$e^{p_j/j}.$$

Поэтому

$$Z_{\text{SET}} = \prod_j e^{p_j/j} = \exp \left(\sum_j \frac{p_j}{j} \right).$$

Смысл перехода можно запомнить так:

суммируем по всем независимым наборам $(m_1, m_2, \dots)$ $\iff$ для каждого j независимо выбираем m_j .

И произведение сумм как раз кодирует этот независимый выбор.

10/8/2026, 1:01:52 PM · gpt-5-6-thinking



Хорошо, давай сделаем подстановки и попробуем получить из Z_{set} что-нибудь интересное. Те две подстановки что у нас были дают, соответственно: $\exp(x)$ и $\exp(-\log(1-x))$, то есть $\text{SET}()$ и $\text{SEQ}()$

10/8/2026, 1:08:40 PM



Да — только одно важное уточнение:

$$Z_{\text{SET}}(x, 0, 0, \dots) = e^x$$

— это **EGF помеченного** SET , а

$$Z_{\text{SET}}(x, x^2, x^3, \dots) = \exp\left(x + \frac{x^2}{2} + \dots\right) = \frac{1}{1-x}$$

— **OGF непомеченного** SET .

Формально $1/(1-x)$ совпадает с производящей функцией SEQ , но это не значит, что $\text{SET}=\text{SEQ}$: просто у непомеченного множества есть ровно одна форма каждого размера, как и у непомеченного списка в нашем прежнем смысле формы.

А теперь сделаем действительно интересную подстановку — собственно Pólya.

Возьмём пока только множества размера 3:

$$Z_{\text{SET},3} = \frac{1}{6}(p_1^3 + 3p_1p_2 + 2p_3).$$

Пусть каждый атом можно покрасить либо в красный цвет с весом r , либо в синий с весом b .

Для j -цикла подставляем

$$p_j = r^j + b^j.$$

Почему? Если раскраска должна остаться неподвижной при j -цикле, все j элементов этого цикла обязаны иметь один цвет. Поэтому цикл либо весь красный — вес r^j , либо весь синий — b^j .

Получаем

$$\frac{1}{6} [(r+b)^3 + 3(r+b)(r^2+b^2) + 2(r^3+b^3)].$$

Попробуй упростить этот многочлен.

Должно получиться что-то очень простое, где каждый моном непосредственно соответствует одному типу раскраски трёх **неразличимых с точностью до перестановки** атомов.

10/8/2026, 1:09:15 PM · gpt-5-6-thinking





$$(r+b)(r^2+b^2) = r^3 + r^2b + rb^2 + b^3$$

10/8/2026, 3:34:06 PM



Да. Тогда весь числитель:

$$\begin{aligned}(r+b)^3 &= r^3 + 3r^2b + 3rb^2 + b^3, \\ 3(r+b)(r^2+b^2) &= 3r^3 + 3r^2b + 3rb^2 + 3b^3, \\ 2(r^3+b^3) &= 2r^3 + 2b^3.\end{aligned}$$

Складываем:

$$6r^3 + 6r^2b + 6rb^2 + 6b^3.$$

После деления на 6:

$$r^3 + r^2b + rb^2 + b^3.$$

И это буквально четыре орбиты раскрасок множества из трёх элементов:

$$RRR, \quad RRB, \quad RBB, \quad BBB.$$

То есть после факторизации по всем перестановкам уже неважно, **какие именно** два элемента красные: остаётся только число красных и синих.

Вот здесь Pólya уже начинает делать что-то содержательное: подстановка

$$p_j \mapsto r^j + b^j$$

превращает информацию о циклических типах симметрий в счёт раскрасок с точностью до этих симметрий.

Следующий пример я бы сделал чуть интереснее, потому что для SET результат почти очевиден. Возьмём **4 вершины на окружности с точностью до вращений**. Для группы C_4 cycle index:

$$\frac{1}{4} (p_1^4 + p_2^2 + 2p_4).$$

И подставим снова

$$p_j = r^j + b^j.$$

Тогда впервые окажется, что раскрасок с двумя красными и двумя синими будет **не одна орбита, а две**: соседние красные и чередующиеся красные. Это уже настоящий эффект геометрической симметрии.

10/8/2026, 3:34:18 PM · gpt-5-6-thinking





И что получается после подстановки?

10/8/2026, 3:37:32 PM



Получается:

$$\frac{1}{4} [(r+b)^4 + (r^2+b^2)^2 + 2(r^4+b^4)].$$

Раскрываем:

$$(r+b)^4 = r^4 + 4r^3b + 6r^2b^2 + 4rb^3 + b^4,$$
$$(r^2+b^2)^2 = r^4 + 2r^2b^2 + b^4.$$

Поэтому числитель равен

$$4r^4 + 4r^3b + 8r^2b^2 + 4rb^3 + 4b^4,$$

и после деления на 4:

$$r^4 + r^3b + 2r^2b^2 + rb^3 + b^4.$$

Смысл коэффициентов:

- r^4 : все красные — 1 орбита;
- r^3b : три красных, один синий — 1;
- r^2b^2 : **2 орбиты**;
- rb^3 : одна красная, три синих — 1;
- b^4 : все синие — 1.

Две орбиты при $2+2$ — это как раз:

$RRBB$

с двумя соседними красными и

$RBRB$

с чередующимися цветами.

Итого при двух цветах с точностью до **вращений** квадрата:

$$1 + 1 + 2 + 1 + 1 = \boxed{6}$$

неэквивалентных раскрасок.

10/8/2026, 3:37:38 PM · gpt-5-6-thinking



У меня сложилось впечатление, что Z_{set} появилось несколько немотивировано (просто догадка), а потом мы начали подставлять и получать интересные подсчёты. Переход от формулы Burnside к Z_{set} (точнее к мономам) был очень коротким и загадочным.

10/8/2026, 3:42:25 PM



Да — согласен. Я слишком быстро сделал главный концептуальный переход. В результате Z_{SET} выглядел как удачная формула, которую потом начали «магически» подставлять. На самом деле его лучше **вывести прямо из Burnside как Burnside с запоминающими переменными**.

Начнём заново с самого перехода.

Формула Burnside для действия группы G на объектах:

$$\#(\text{орбит}) = \frac{1}{|G|} \sum_{\sigma \in G} \# \text{Fix}(\sigma).$$

Если мы считаем структуры на n метках с точностью до переименования, группа — S_n .

Но Burnside в таком виде делает одну неприятную вещь: он складывает вместе все перестановки и **забывает, какого они были типа**.

А нам хотелось бы сохранить эту информацию.

Возьмём перестановку

$$\sigma = (12)(345)(6).$$

Она состоит из:

- одного 1-цикла;
- одного 2-цикла;
- одного 3-цикла.

То есть

$$c_1 = 1, \quad c_2 = 1, \quad c_3 = 1.$$

Мы просто придумываем формальный «паспорт» этой перестановки:

$$p_1 p_2 p_3.$$

Если, например,

$$\sigma = (12)(34)(56),$$

то паспорт:

$$p_2^3.$$

То есть вообще:

$$\sigma \mapsto p_1^{c_1(\sigma)} p_2^{c_2(\sigma)} p_3^{c_3(\sigma)} \dots$$

Это пока **никакая не теорема**. Мы просто кодируем циклический тип перестановки мономом.

Теперь берём Burnside, но вместо того чтобы сразу забыть, какая перестановка дала вклад, прикрепляем к её вкладу этот паспорт:

$$Z_{F,n} = \frac{1}{n!} \sum_{\sigma \in S_n} \text{Fix}_F(\sigma) p_1^{c_1(\sigma)} p_2^{c_2(\sigma)} \dots$$

Вот это и есть n -я часть cycle index species F .

То есть cycle index — это буквально:

Burnside average, но с сохранённым циклическим типом каждой симметрии.

Для **SET** всё особенно просто: на данном множестве существует одна SET-структура, и любая перестановка её сохраняет. Поэтому

$$\text{Fix}_{\text{SET}}(\sigma) = 1.$$

Значит

$$Z_{\text{SET},n} = \frac{1}{n!} \sum_{\sigma \in S_n} p_1^{c_1(\sigma)} p_2^{c_2(\sigma)} \dots$$

Для $n = 3$ мы именно поэтому получили

$$\frac{1}{6}(p_1^3 + 3p_1p_2 + 2p_3).$$

Не догадались — просто выписали шесть элементов S_3 и заменили каждый его циклическим паспортом.

Теперь становится понятно, **зачем вообще сохранять этот паспорт**. Возьмём раскраски.

Пусть есть красный и синий цвета. Возьмём одну конкретную перестановку, например

$$\sigma = (12)(345).$$

Какие раскраски она фиксирует?

Чтобы раскраска не изменилась после перестановки, все элементы одного цикла должны иметь одинаковый цвет.

Для 2-цикла:

$$(12)$$

есть два варианта:

- оба красные — вес r^2 ;
- оба синие — вес b^2 .

Итого:

$$r^2 + b^2.$$

Для 3-цикла:

$$r^3 + b^3.$$

Поскольку циклы независимы, общий вклад:

$$(r^2 + b^2)(r^3 + b^3).$$

Но паспорт нашей перестановки был

$$p_2p_3.$$

Значит получить число/вес фиксированных раскрасок можно просто подстановкой

$$p_j \mapsto r^j + b^j.$$

Вот откуда берётся подстановка Pólya. Она тоже не магическая.

Получается такая цепочка:

перестановка σ

↓

её циклы

↓

$$p_1^{c_1} p_2^{c_2} \dots$$

$$\downarrow \quad p_j \mapsto r^j + b^j$$

фиксированные этой σ раскраски.

А затем Burnside усредняет это по всем σ .

И только **после этого** имеет смысл заметить: раз этот многочлен с p_j такой полезный, давайте сохраним его один раз и потом делать разные подстановки. Это и есть cycle index.

Теперь становятся мотивированными и наши две прежние подстановки.

Если положить

$$p_1 = x, \quad p_{j>1} = 0,$$

то мы намеренно убиваем все перестановки, кроме identity. Остаётся

$$\frac{|F[n]|}{n!} x^n,$$

то есть labeled EGF.

Если же положить

$$p_j = x^j,$$

то любой моном циклического типа превращается в

$$x^{1c_1+2c_2+3c_3+\dots} = x^n.$$

Поэтому коэффициент становится

$$\frac{1}{n!} \sum_{\sigma \in S_n} \text{Fix}_F(\sigma),$$

а по Burnside это число орбит, то есть число непомеченных структур. Получаем OGF.

Так что общая картина гораздо лучше выглядит так:

Burnside $\longrightarrow$ не забывать циклический тип $\longrightarrow Z_F$
--

а уже потом

$$Z_F \longrightarrow \begin{cases} \text{EGF} & p_1 = x, p_{j>1} = 0, \\ \text{OGF} & p_j = x^j, \\ \text{Pólya colorings} & p_j = r^j + b^j + \dots \end{cases}$$

И вот **в таком порядке** cycle index, по-моему, перестаёт выглядеть придуманным из воздуха.

Я бы даже считал это ещё одним пунктом нашего списка «скрытых вещей, которые надо было явно проговорить»: Z_F — не новая загадочная generating function, а **усовершенствованная формула Burnside, которая вместо немедленного усреднения сначала запоминает циклическую структуру каждой симметрии.**

10/8/2026, 3:42:43 PM · gpt-5-6-thinking