Разбор задачи «Возводим в квадрат»

Автор задачи:	Михаил Иванов
Подготовка тестов и решений:	Михаил Иванов
Автор разбора:	Михаил Иванов

Тесты с маленьким k можно было пройти, заметив, что k-я с конца цифра числа может быть получена из его остатка от деления на 10^k (потому что этот остаток — это как раз последние k цифр числа). А для того, чтобы найти этот остаток, можно найти остаток от деления N на 10^k — это число из k единиц, которое мы назовём N_k — возвести его в квадрат и у полученного числа длины $2k-1$ взять k-ю цифру (неважно, слева или справа). Таким образом, требуется лишь, чтобы N_k^2, лежащее в пределах от 10^{2k-1} до $1{,}5 \cdot 10^{2k-1}$, влезало в ваш целочисленный тип. Если вы пользуетесь только 32-разрядными числами, то решение сработает при $k \leqslant 5$ и наберёт четыре балла; если 64-разрядным, то $k \leqslant 10$ и 14 баллов; если же вы воспользовались 128-разрядным типом, доступным в системе на языке C++, то вам доступно $k \leqslant 20$ и 34 балла. Кроме того, можно было воспользоваться длинной арифметикой, которая сразу позволяет решить задачу для многотысячных k; например, длинная арифметика Python позволяла набрать таким способом 74 балла. В этом решении может быть полезна такая формула: $N_k = \frac{10^k-1}{9}$.

Для того, чтобы решить задачу для действительно больших k, уже было необходимо заметить, как устроены цифры числа N^2. Выясняется, что самая правая цифра в нём 1, а дальше идёт цикл 234567890 длины 9, то есть в самом числе цифры устроены так:

$$N^2 = \ldots 0987654320987654320987654321.$$

Поэтому можно было отдельно рассмотреть случай $k = 1$, а затем взять остаток от деления на 9 и в зависимости от него вывести фиксированный ответ.

Как же доказать эту закономерность? Мы предложим идею доказательства, а детали оставим в качестве упражнения. Нам надо понять, как устроены последние k цифр числа N_k^2, где N_k — число из k единиц. Представим его в виде суммы $10^{k-1}N_k + 10^{k-2}N_k + \ldots + N_k$. Нетрудно доказать по индукции, что если взять первые i слагаемых этой суммы и сложить, то последние k цифр этого числа выглядят так: сначала последние i цифр последовательности

$$\ldots 0987654320987654320987654321,$$

а затем $k - i$ нулей.

Вашему вниманию предлагается также такое упражнение: а как находить k-ю цифру не с конца, а с начала числа?

Разбор задачи «Мало частей»

Автор задачи:	Михаил Иванов
Подготовка тестов и решений:	Владислав Макаров
Автор разбора:	Владислав Макаров

Заметим, что любой подотрезок хорошей строки — тоже хорошая строка. Поэтому в качестве первой строки (то есть строки, содержащий первый символ s) разбиения «наиболее выгодно» взять строку наибольшей возможной длины.

Формально говоря, среди всех разбиений с наименьшим количеством частей есть хотя бы одно, в котором первая строка имеет наибольшую возможную длину. Действительно, возьмём любое разбиение на хорошие строки, продлим в нём первую строку до максимума и сократим (или даже удалим) те части разбиения, на которые мы при этом «заехали». Получим разбиение на хорошее строки, в котором не больше частей, чем было раньше, а первая строка имеет наибольшую возможную длину.

Таким образом, задачу можно решать жадно: сделаем первую часть разбиения настолько длинной, насколько сможем, потом вторую, и так далее. Этот алгоритм можно реализовать за линейное от длины s время, если уметь понимать, останется ли текущая строка разбиения хорошей, если добавить к ней какой-то символ. Для этого достаточно хранить её длину и множество символов, которые в ней когда-либо встречались (это множество имеет размер не больше 3, так как строка хорошая).

Есть и другие, более сложные, подходы.

Разбор задачи «Две сумки»

Автор задачи:	Иван Казменко
Подготовка тестов и решений:	Иван Казменко
Автор разбора:	Иван Казменко

Решение подзадачи с $n \leqslant 10$: перебор за $O(2^n \cdot n)$. Можно перебрать все 2^n возможных подмножеств вещей, которые следует положить в первую сумку, положить остальные вещи во вторую сумку, после чего проверить, что все ограничения выполнены.

Решение подзадачи с $n \leqslant 100$: рюкзак за $O(n \cdot x)$. Будем добавлять вещи по одной и каждую класть в какую-нибудь сумку. Кроме номера рассматриваемой вещи k, нам нужно следить за двумя числами: объёмом вещей v в первой сумке и весом вещей w во второй сумке. Один из них сделаем параметром, а другой — значением функции. Получается, что нам нужно вычислять $f(k, v) = \min w$: если мы разложили по сумкам первые k вещей и положили в первую сумку вещи суммарного объёма v, какой минимальный суммарный вес w могут иметь вещи, попавшие во вторую сумку? При добавлении вещи k с объёмом v_k и весом w_k мы либо кладём её в первую сумку, либо во вторую: $f(k, v) = \min\{f(k-1, v-v_k), f(k-1, v) + w_k\}$. Для восстановления ответа сохраним, какой из двух переходов произошёл.

В последней подзадаче такое решение может потребовать слишком много памяти: с параметрами $0 \leqslant k \leqslant 500$ и $0 \leqslant v \leqslant 250\,000$ получается $\approx 125\,000\,000$ состояний, для каждого из которых удобно было бы хранить два числа — минимальный вес и номер перехода для восстановления ответа. В принципе эту информацию как раз можно уложить в четыре байта.

Если памяти не хватило, можно заметить, что значения $f(k, \cdot)$ зависят только от $f(k-1, \cdot)$. Поэтому можно хранить f только для текущего и предыдущего значений k. Таблицу для восстановления ответа так сжать не удастся, зато для каждого состояния можно хранить один байт или даже один бит.

А ещё можно восстанавливать ответ, не сохраняя переходы: в каждом состоянии по пути будем смотреть на значения и заново выяснять, какое из двух возможных состояний могло быть предыдущим. Такой способ требует написать больше кода в обмен на небольшую экономию памяти.

Разбор задачи «Гнём скобки»

Автор задачи:	Владислав Макаров
Подготовка тестов и решений:	Владислав Макаров
Автор разбора:	Владислав Макаров

Для решения первой подгруппы достаточно было перебрать итоговую последовательность. Даже самая наивная реализация будет работать достаточно быстро.

Вторая подгруппа решается динамическим программированием за $O(n^3)$. А именно, $d(\ell, r)$ — минимальное количество энергии, которое нужно потратить, чтобы подотрезок $[\ell, r)$ строки стал правильной скобочной последовательностью (ПСП). По определению динамики ответом будет $d(0, 2n)$ (напоминаю, что n — это *половина* длины строки).

Эта динамика пользуется рекурсивным определением ПСП. Базой динамики будут условия $d(\ell, \ell) = 0$ для $0 \leqslant \ell \leqslant 2n$ (пустая строка и так являвется ПСП). Состояния обрабатываются в порядке возрастания длины отрезка $[\ell, r)$, то есть $r - \ell$.

Заметим, что нас интересуют только состояния с чётным $r - \ell$, так как остальные отрезки нельзя превратить в ПСП просто из-за чётности. Для перехода динамики заметим, что, по рекурсивному определению ПСП, превратить подотрезок $[l, r)$ в ПСП можно тремя способами:

1. Для какого-то m ($\ell < m < r$) превратить $[l, m)$ в ПСП и $[m, r)$ в ПСП.
2. Превратить l-й символ в «`[`», $(r-1)$-й в «`]`» и превратить подотрезок $[\ell+1, r-1)$ в ПСП.
3. Превратить l-й символ в «`(`», $(r-1)$-й в «`)`» и превратить подотрезок $[\ell+1, r-1)$ в ПСП.

Следовательно, каждое $d(\ell, r)$ можно посчитать за $O(r-\ell) = O(n)$ времени, самое «дорогое» — перебор m.

Третья и последняя подгруппа требовала учёта специфики конкретной задачи. Во-первых, поскольку переворачивать скобки можно бесплатно, для нас нет никакой разницы между закрывающей и открывающей скобками одного типа, они полностью взаимозаменяемы. Получается, что задачу можно переформулировать следующим образом: есть строка из символов «`a`» (соответствует круглым скобкам) и «`b`» (соответствует квадратным скобкам), требуется получить из неё *хорошую* строку минимальным количеством изменений символов. Хорошие строки определяются так:

1. Пустая строка — хорошая.
2. Если s — хорошая, то `a`s`a` и `b`s`b` — тоже хорошие.
3. Если s и t хорошие, то st — тоже хорошая.
4. Если из правил 1–3 не следует, что строка — хорошая, то она такой и не является.

Индукцией по определению хорошей строки несложно доказать, что строка хорошая тогда и только тогда, когда следующий алгоритм со стеком даёт на выходе пустой стек:

1. Изначально стек пуст.
2. Обрабатываем символы строки по одному, пусть текущий символ — c.
3. Если стек непуст и на вершине лежит c, то снимем c со стека (и ничего не добавим). Если же стек пуст или на вершине лежит не c, то добавим c на стек (ничего при этом не сняв).

Действительно, случай 3 в определении хорошей строки соответствуют тому, что стек оказался пуст в какой-то момент работы алгоритма выше (кроме самого начала работы, когда не был обработан ни один символ, и самого конца, когда были обработаны все символы). Случай 2 соответствует тому, что стек никогда не становился пустым, и поэтому первый символ строки «аннигилировался» с последним.

Итак, мы можем проверять строку на хорошесть с помощью такого алгоритма. Оказывается, мы можем найти таким образом и ответ на задачу! Давайте сделаем следующее: применим к строке, которую мы хотим переделать в хорошую, алгоритм выше. Так как она не обязана быть хорошей, то стек в конце алгоритма мог остаться непустым. Однако, в нём точно чётное количество элементов (так как символы «аннигилируются» парами) и эти символы чередуются (иначе бы они «аннигилировались» при добавлении). Не умаляя общности, в стеке $2k$ элементов и он выглядит как `ab...ab`. Посмотрим на все буквы «`b`» в стеке, найдём их позиции в исходной строке и поменяем *именно эти* «`b`»-шки на «`a`»-шки. Тогда, когда мы раньше добавляли одну из *именно этих* «`b`»-шек на стек, мы теперь будем снимать со стека одну «`a`»-шку. При этом никакого влияния на будущее состояние стека это не окажет, так как стек будет становиться пустым после обработки каждой изменённой буквы. Таким образом, мы потратили всего k единиц энергии, где $2k$ — итоговый размер стека.

Почему нельзя потратить меньше энергии? Утверждается, что изменение одного символа в строке не поможет изменить размер стека больше чем на 2 (следовательно, таких изменений нужно хотя бы k), при этом не только в конце работы алгоритма, а *вообще* в любой момент времени. Точнее, можно доказать индукцией по номеру символа, что в любой момент времени стеки для исходной строки и строки с одним изменённым символом соотносятся одним из трёх следующих способов:

1. либо они просто равны,
2. либо они оба имеют размер 1,
3. либо их размеры отличаются ровно на 2, и один из них — префикс другого.

Код получается очень короткий: просто проход со стеком.

Замечание: в этой задаче помогают базовые познания в теории групп, но, как видно выше, можно было обойтись и без них. Приведённое выше объяснение не самое простое, но содержит несколько фактов, которые могут оказаться полезными и в других задачах.

Разбор задачи «Телепатия с числом»

Автор задачи:	Владислав Макаров
Подготовка тестов и решений:	Иван Казменко
Автор разбора:	Иван Казменко

В первой подзадаче диапазон возможных чисел очень велик: можно заранее выбрать какое-нибудь случайное число и надеяться, что в тестах оно не будет дано.

Во второй подзадаче, наоборот, возможных входных данных довольно мало. Можно явно сконструировать решение для всех возможных наборов.

Решим задачу в общем случае: если $2k + 1 \leqslant n$, существует точное решение.

Рассмотрим строку d из n скобок $d_1, d_2, \ldots, d_n$: скобка d_i открывающая, если зрители написали на доске число i, или закрывающая, если число i зрители не написали. Например, при $n = 10$ и множестве чисел $\{2, 6, 7, 10\}$ строка d равна «)()))(())(».

Наша задача — добавить одно число на доску. В терминах нашей строки d это означает замену одной закрывающей скобки на открывающую. Хочется выбрать закрывающую скобку так, чтобы и её позицию, и позицию получившейся открывающей скобки можно было восстановить однозначно.

Вот как это сделать. Будем заменять парные скобки на точки: если есть подстрока вида (s), где s состоит только из точек, то эти две скобки заменяются на точки. В нашем примере получим «)()))(())(» → «)()))(..)(» → «)()))...(» → «)..)....(». Порядок действий может быть другим, но получится то же самое.

Что получится в итоге? Учтём, что закрывающих скобок было больше, чем открывающих. А ещё заметим, что открывающая скобка не может остаться левее закрывающей. Итак, кроме точек, останется несколько закрывающих скобок (хотя бы одна), а за ними несколько открывающих (их может и не быть).

Какую закрывающую скобку можно однозначно найти в этой строке? Выберем последнюю (самую правую) из оставшихся. Заменим её на открывающую (то есть допишем число, соответствующее её позиции, на доску). В примере получится, что это число 5. Строка d поменяется так: «)()))(())(» → «)()))((())(». Более того, строка с удалёнными парами скобок тоже поменяется ровно в этом же символе: «)..)....(» → «)..)....(». Заметим, что заменённая скобка — теперь первая (самая левая) из оставшихся открывающих!

Итак, вот решение задачи. Получив числа в первый раз, за Васю, напишем строку d, заменим парные скобки на точки, найдём самую правую закрывающую скобку (пусть она находится на позиции i) и добавим на доску число i. Получив числа во второй раз, за Петю, напишем строку d, заменим парные скобки на точки, найдём самую левую открывающую

скобку (пусть она находится на позиции i) и скажем, что i — это и есть число, добавленное Васей.

Аккуратная реализация этого решения позволяет выполнить все действия за $O(n)$. В реализации можно даже не выписывать скобочную последовательность — достаточно следить за балансом, то есть разностью количества открывающих и закрывающих скобок.

Разбор задачи «Парк и дома»

Авторы задачи: Гуго Хадвигер, Отфрид Чеонг, Мира Ли, Глеб Ненашев, Вячеслав Соколов
Подготовка тестов и решений: Михаил Иванов
Автор разбора: Михаил Иванов

Первая идея — попробовать выпуклые многоугольники. Если использовать треугольники, можно добиться шести касаний, а вот с любым параллелограммом можно добиться уже восьми. Увы, для большего числа касаний нужна более тонкая настройка: в 1957 году Гуго Хадвигер доказал[1], что выпуклыми фигурами больше восьми касаний не добиться.

Также восемь касаний получается и с некоторыми другими фигурами, имеющими что-то общее с параллелограммами, например, с клеточным крестиком. Последние два примера набирают 25 баллов.

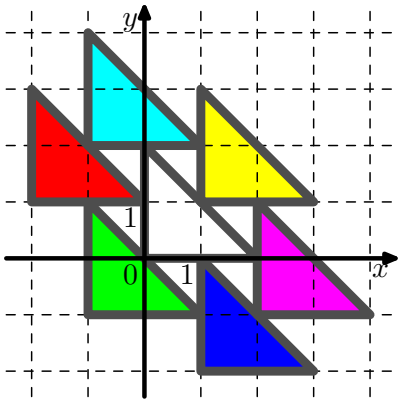

Рис. 1: Треугольник, $n = 6$

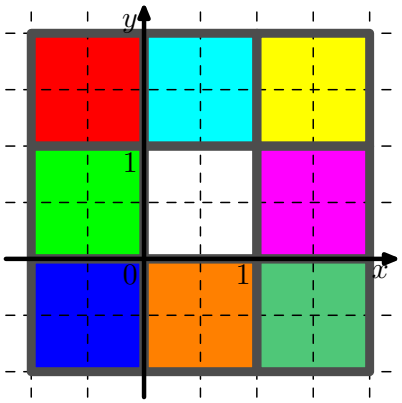

Рис. 2: Квадрат, $n = 8$

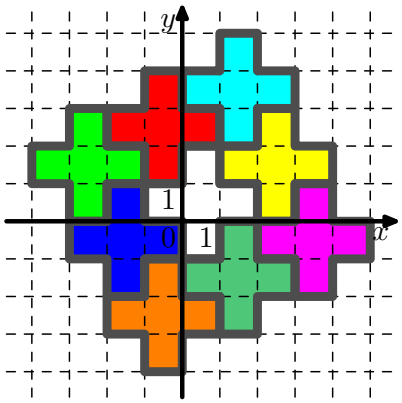

Рис. 3: Крест, $n = 8$

Отфрид Чеонг и Мира Ли нашли[2] клетчатый пример, который работает для любого n и состоит из экспоненциального количества клеточек. Его устройство основано на свойствах двоичной записи чисел. А именно, определим функцию ctz:

$$\mathrm{ctz}(x) = \max\{n \in \mathbb{N}_0 \mid x \text{ кратно } 2^n\},$$

тогда надо рассмотреть ломаную из 2^{n-1} горизонтальных и 2^{n-1} вертикальных отрезков, причём i-й горизонтальный отрезок равен $1 + \mathrm{ctz}(i)$, а i-й вертикальный отрезок — $n - 1$. Сдвинув эту фигурку на векторы $\left(i - 1 + (n-1)\left(2^{n-1} - 2^{n-1-i}\right), n - 2i + 2^n - 2^{n-i}\right)$, получим устраивающую нас систему.

Если рассмотреть ровно такую конструкцию из статьи, то она наберёт всего 35 баллов — при $n > 10$ в ней слишком много сторон. Однако можно заметить, что наш парк центрально симметричен, а все дома располагаются лишь с одной стороны от него. Сделав то же самое

[1] Hugo Hadwiger, "Über Treffanzahlen bei translationsgleichen Eikörpern", Arch. Math. 8 (1957), 212–213
[2] Otfried Cheong, Mira Lee, "The Hadwiger Number of Jordan Regions is Unbounded", Discrete Comput Geom 37, 497–501 (2007)

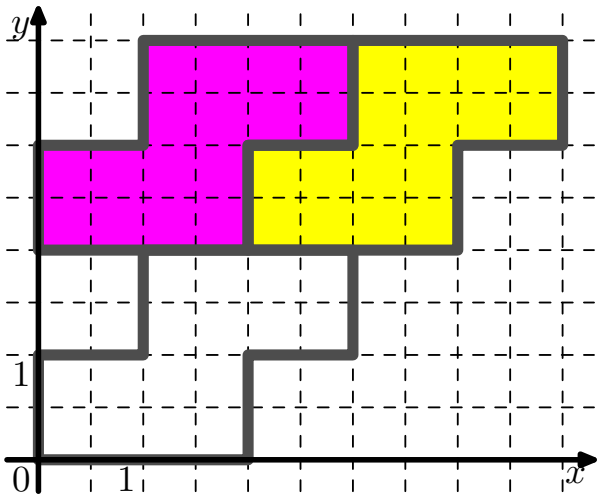

Рис. 4: Двоичная ломаная, $n = 2$

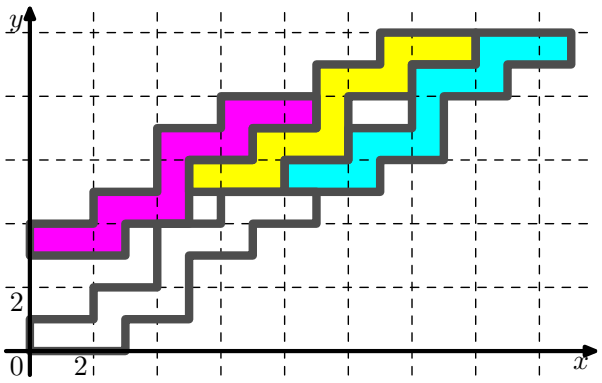

Рис. 5: Двоичная ломаная, $n = 3$

с другой стороны от парка, мы бесплатно удвоим наше n: пример работает при $n \leqslant 20$ и набирает 85 баллов. Кроме того, в этой конструкции есть ещё пара мест, куда можно втиснуть фигурку, так что итоговый результат — $n \leqslant 22$, 95 баллов.

На сто баллов нужно было придумать конструкцию, в которой количество вершин полиномиально от n. И такая конструкция есть! Её придумали несколько лет назад Глеб Ненашев и Вячеслав Соколов. В ней $4n$ вершин. Идея такова: рассмотрим ломаную линию $A_0A_1 \dots A_n$, в которой каждое следующее ребро хотя бы вдвое короче, чем предыдущее, а каждый следующий угол примерно вдвое ближе к $180°$ — таким образом, каждое ребро больше суммы всех следующих, каждый внешний угол больше суммы всех следующих. В решении жюри $A_0 = (0, 0)$, $\overrightarrow{A_iA_{i+1}} = \overrightarrow{\left(2^{n-1}3^{n-1-i}, 6^{n-1-i}\right)}$. У этой ломаной есть интересное свойство: если её сдвинуть на вектор $2\overrightarrow{A_iA_j}$, $i < j$, то полученная ломаная $A'_0A'_1 \dots A'_n$ не пересекается с исходной: действительно, единственные шансы пересечься есть у ребра $A'_{i-1}A'_i$ с ребром A_JA_{J+1}, $J \geqslant j$. Но рёбра A_JA_{J+1}, $J \geqslant j$, не дотянутся до серединного перпендикуляра к $A_jA'_i$, поскольку их сумма слишком мала, а с той стороны от серединного перпендикуляра, где находится точка A_j, ребро $A'_{i-1}A'_i$ находится дальше от прямой $A_jA'_i$, чем рёбра A_JA_{J+1}, так как последние слишком слабо отклоняются от этой прямой.

Теперь рассмотрим центрально симметричные точки A_{-i} к точкам A_i относительно A_0. Получится ломаная $A_{-n}A_{-(n-1)} \dots A_{-1}A_0A_1 \dots A_n$. Рассмотрим её n параллельных сдвигов на векторы $\overrightarrow{A_{-i}A_i}$: легко понять, что каждый такой сдвиг касается ломаной в точке $A_i = A'_{-i}$. Между тем эти параллельные сдвиги между собой не пересекаются, так как если i-й и j-й сдвиги пересеклись, то тогда ломаная пересеклась со своим $2\overrightarrow{A_iA_j}$-сдвигом, что невозможно по интересному свойству.

Наконец, придадим объёма нашей ломаной, чтобы она стала многоугольником: обозначим $R_{-n}R_{-(n-1)} \dots R_{-1}R_0R_1 \dots R_n$ и $L_{-n}L_{-(n-1)} \dots L_{-1}L_0L_1 \dots L_n$ ломаные, полученные из $A_{-n}A_{-(n-1)} \dots A_{-1}A_0A_1 \dots A_n$ сдвигом на вектор $\overrightarrow{(1, 0)}$ и $\overrightarrow{(-1, 0)}$ соответственно. Тогда наш многоугольник — $L_{-n}L_{-(n-1)} \dots L_{-1}A_1A_2 \dots A_{n-1}A_nR_nR_{n-1} \dots R_1A_{-1}A_{-2} \dots A_{-n}$. Тончайшая добавка к ломаной не изменит её свойств: всё ещё $\overrightarrow{A_{-i}A_i}$-сдвиги касаются исходного многоугольника, но не пересекаются друг с другом.

В иллюстрациях мы назвали эту конструкцию «Инь-ян», поскольку она станет похожа на этот символ, если взять её выпуклую оболочку (и если перестать замечать на минутку, что эта фигура столь вытянута).

Заметим, что, как и в прошлом решении, тут можно соптимизировать в два раза конструкцию, сделав касания не только над, но и под многоугольником. Но в этом случае такая оптимизация не требовалась: даже ровно то, что мы привели, при всех $n \leqslant 23$ выдаёт кор-

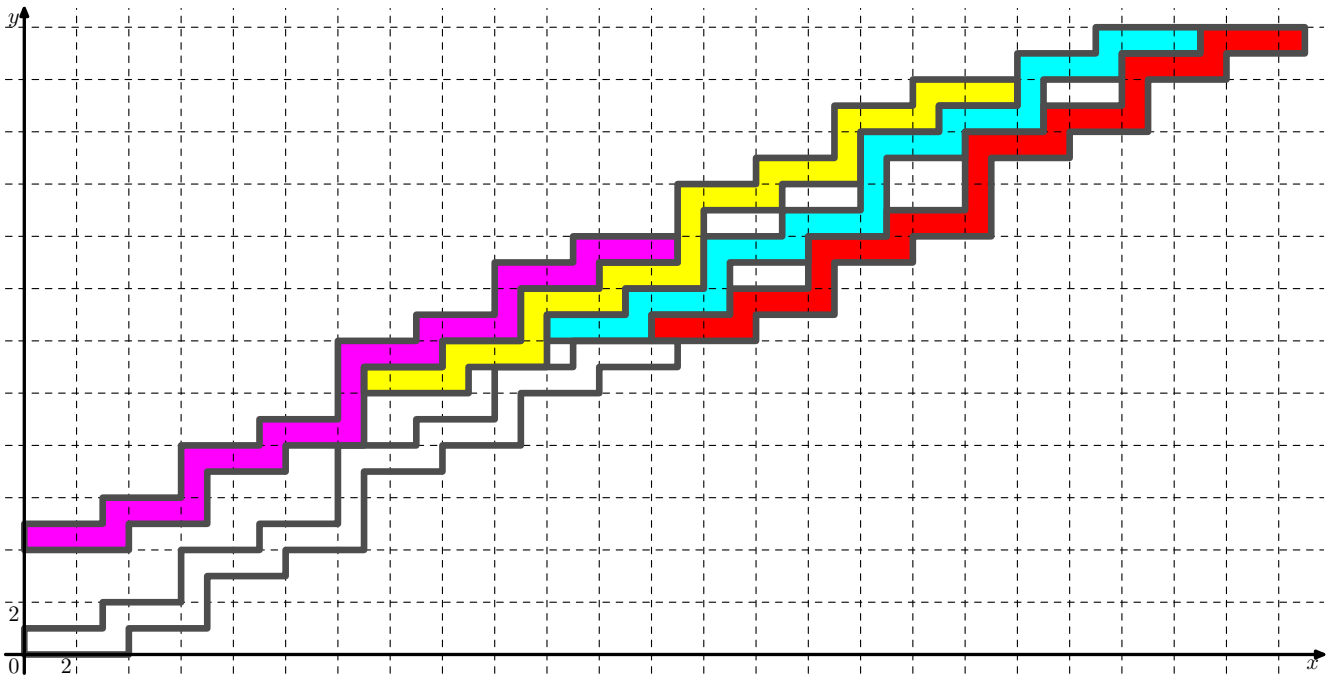

Рис. 6: Двоичная ломаная, $n = 4$

ректный ответ. Впрочем, в отличие от прошлого решения, тут у вершин многоугольника координаты растут стремительно, и решение быстро упирается в ограничение на максимальный модуль координаты. Поэтому, конечно, эта оптимизация вкупе с аккуратностью не повредят.

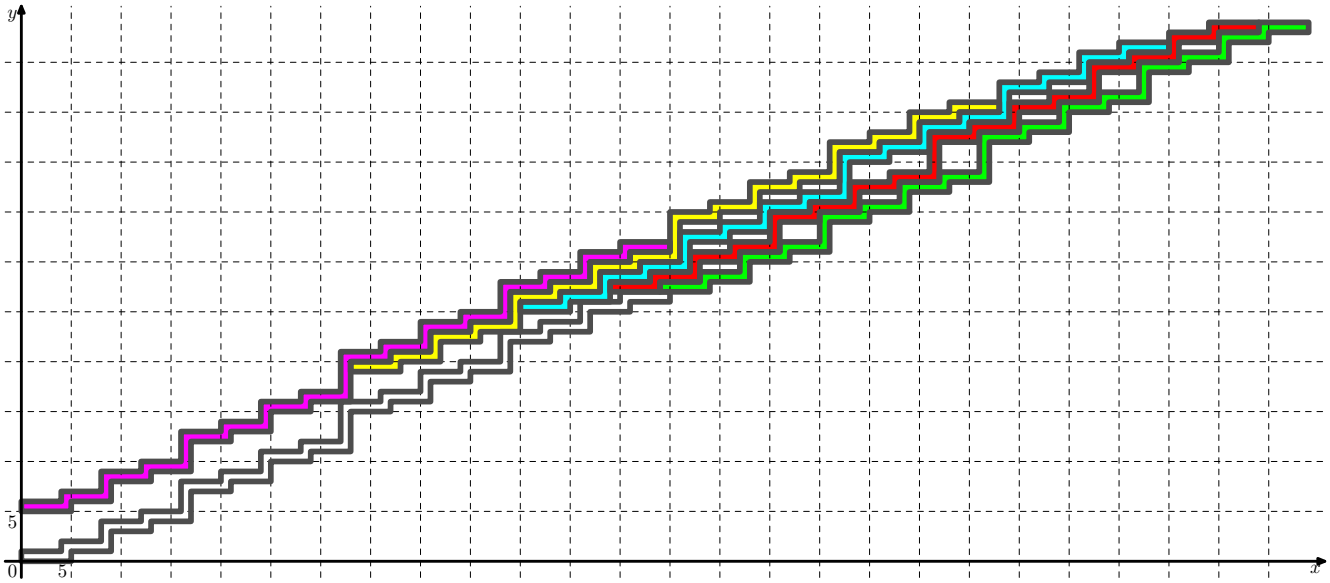

Рис. 7: Двоичная ломаная, $n = 5$

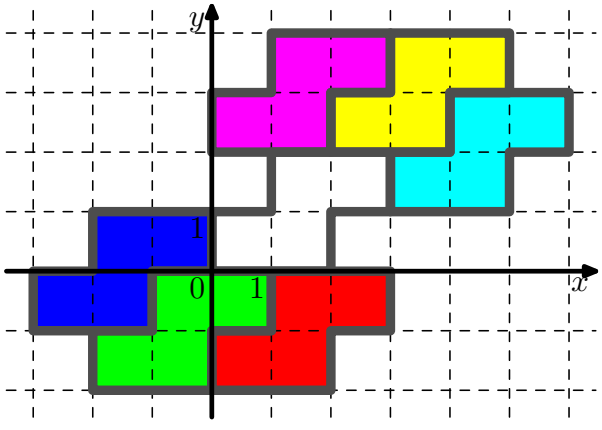

Рис. 8: Симметричная двоичная ломаная, $n = 6$

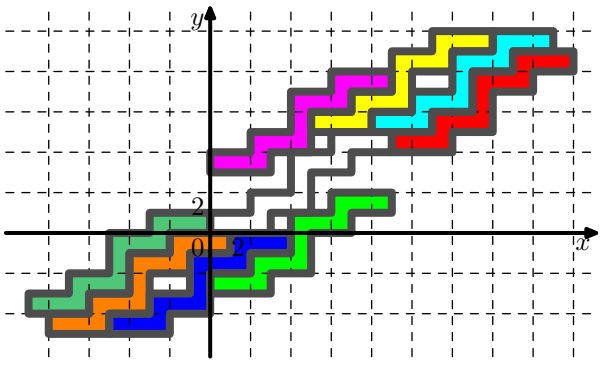

Рис. 9: Симметричная двоичная ломаная, $n = 8$

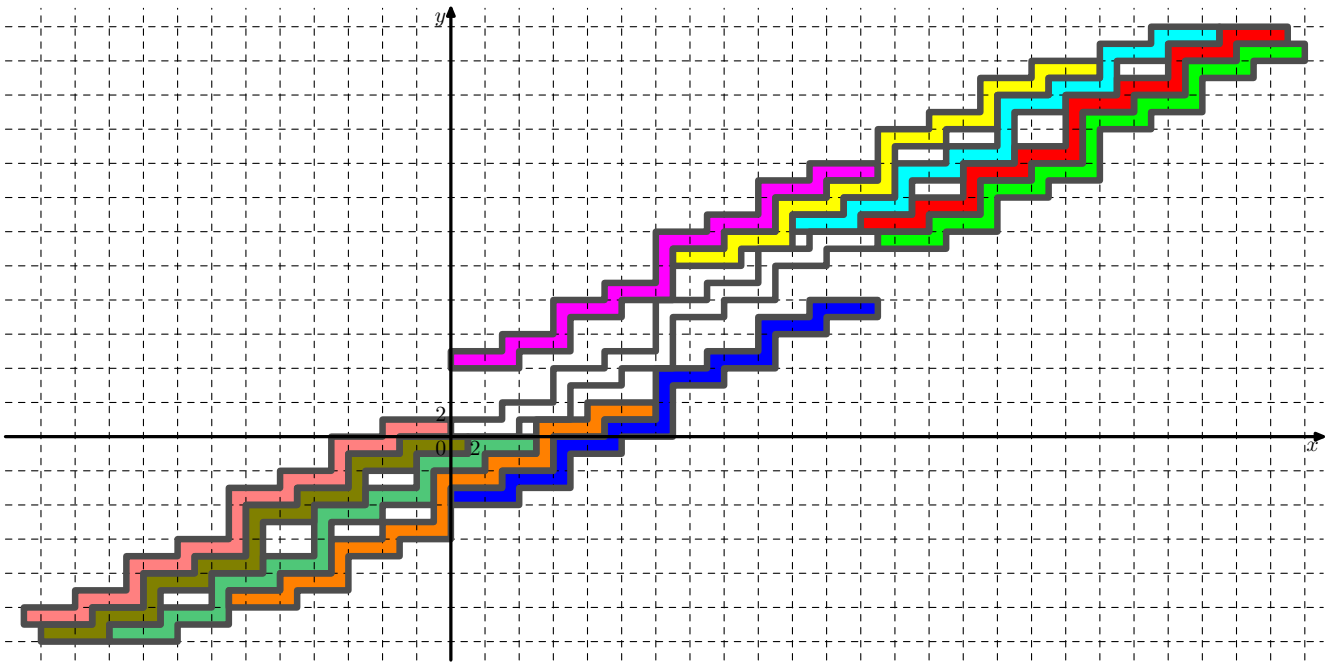

Рис. 10: Симметричная двоичная ломаная, $n = 10$

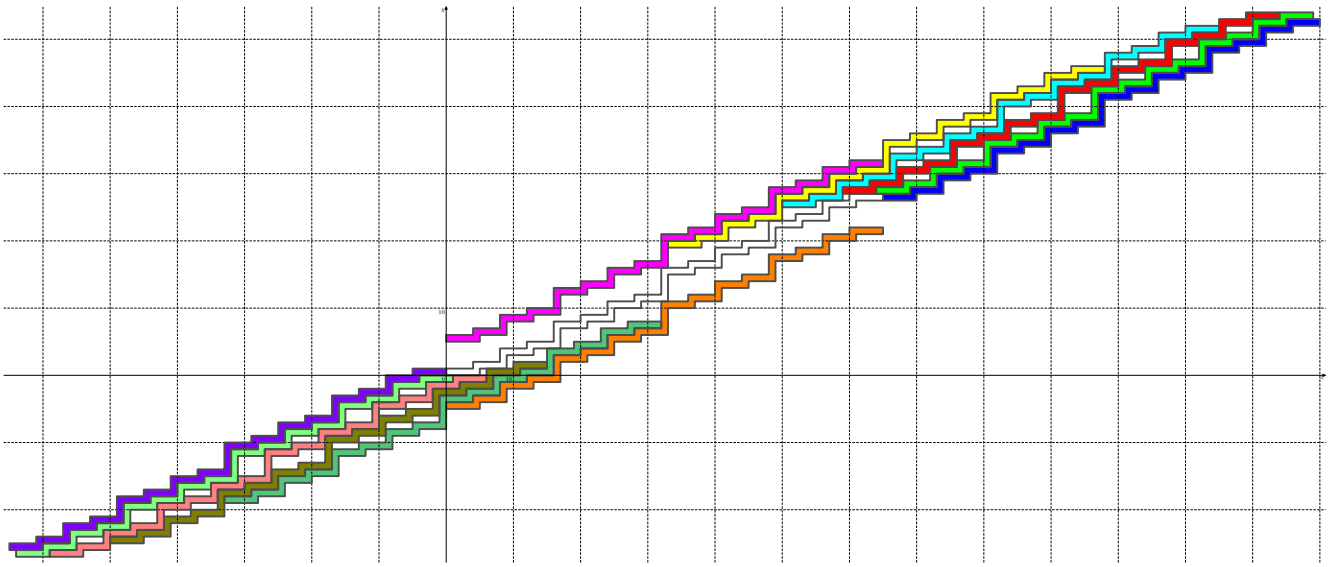

Рис. 11: Симметричная двоичная ломаная, $n = 12$

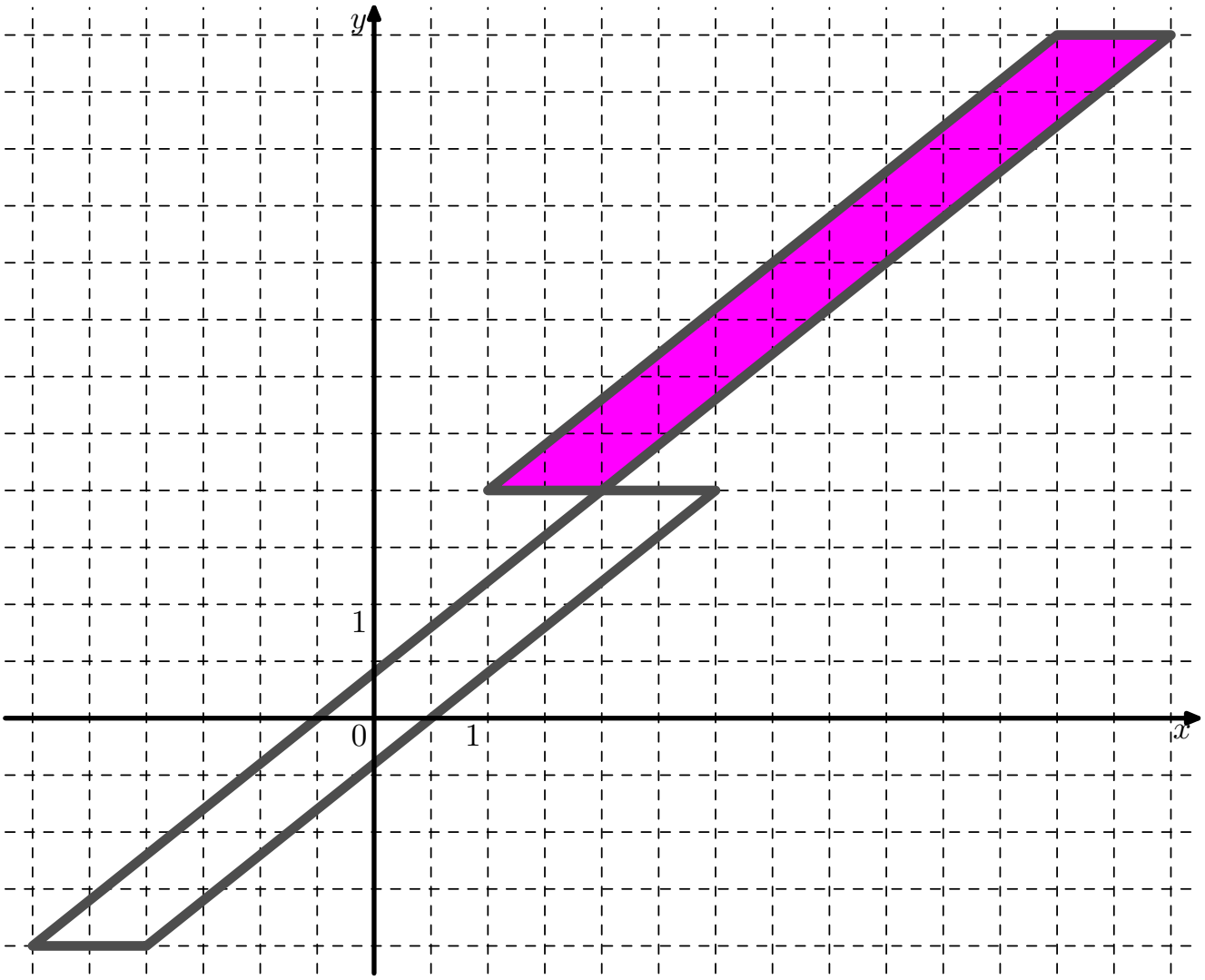

Рис. 12: Инь-ян, $n = 1$

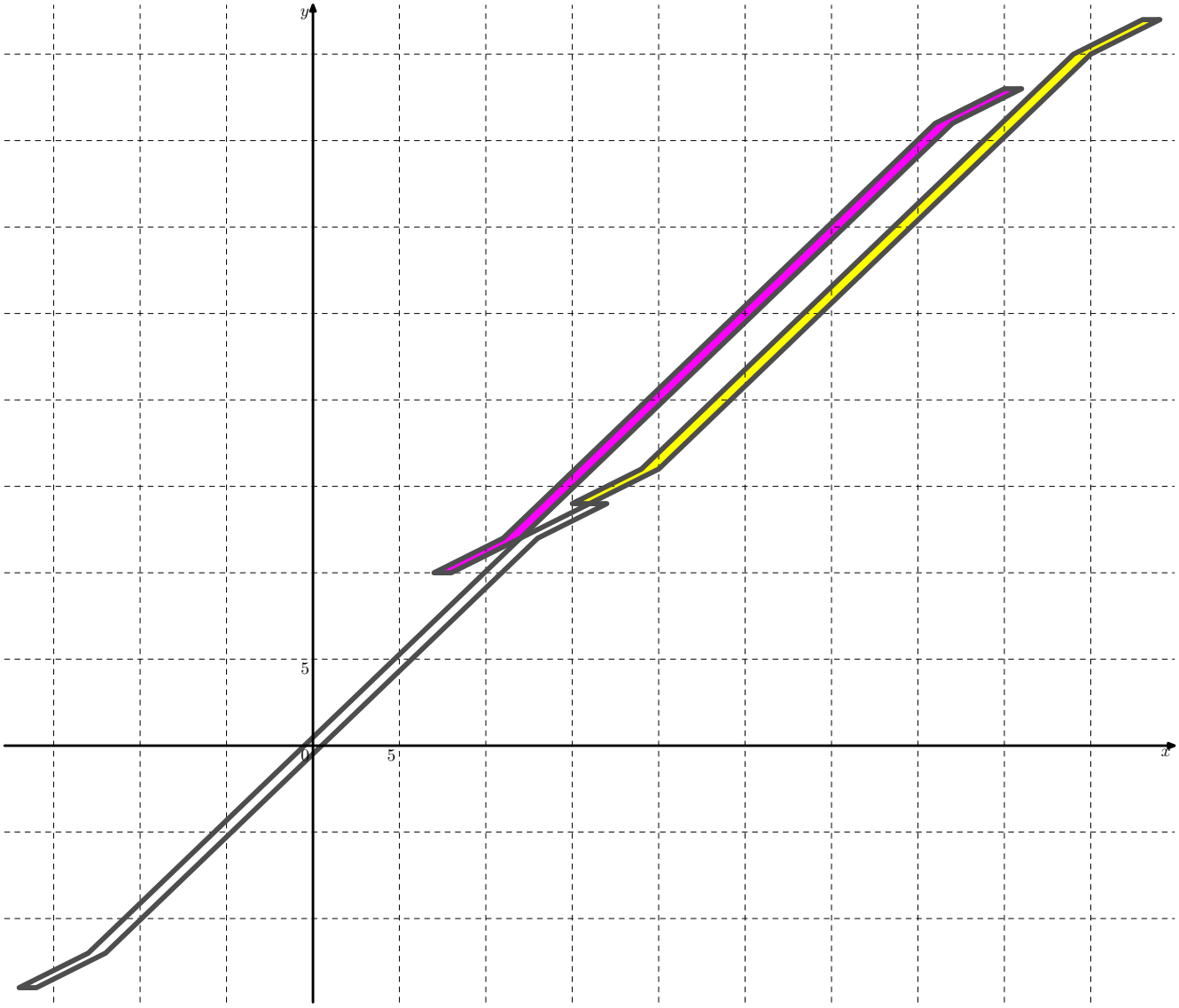

Рис. 13: Инь-ян, $n = 2$

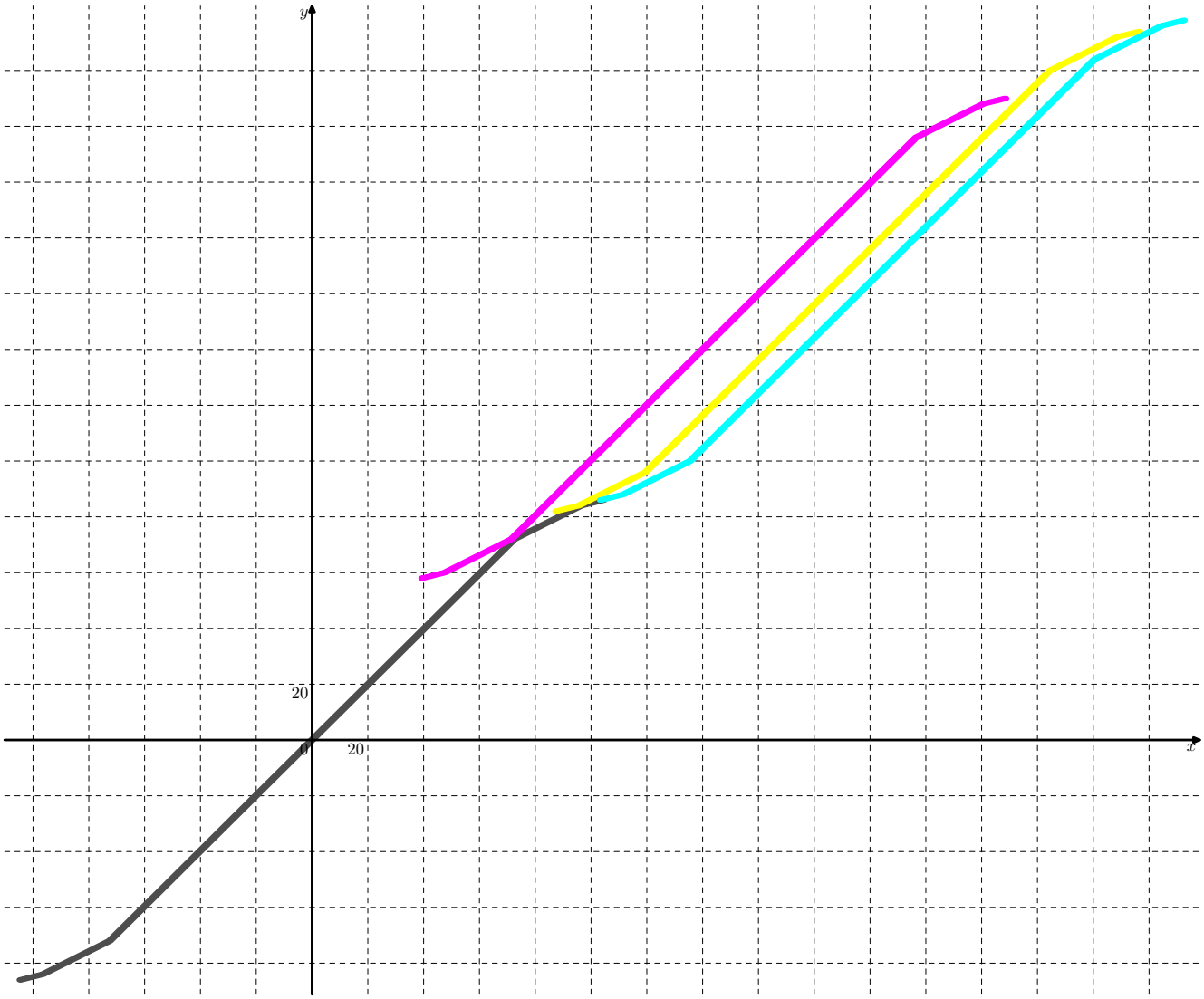

Рис. 14: Инь-ян, $n = 3$

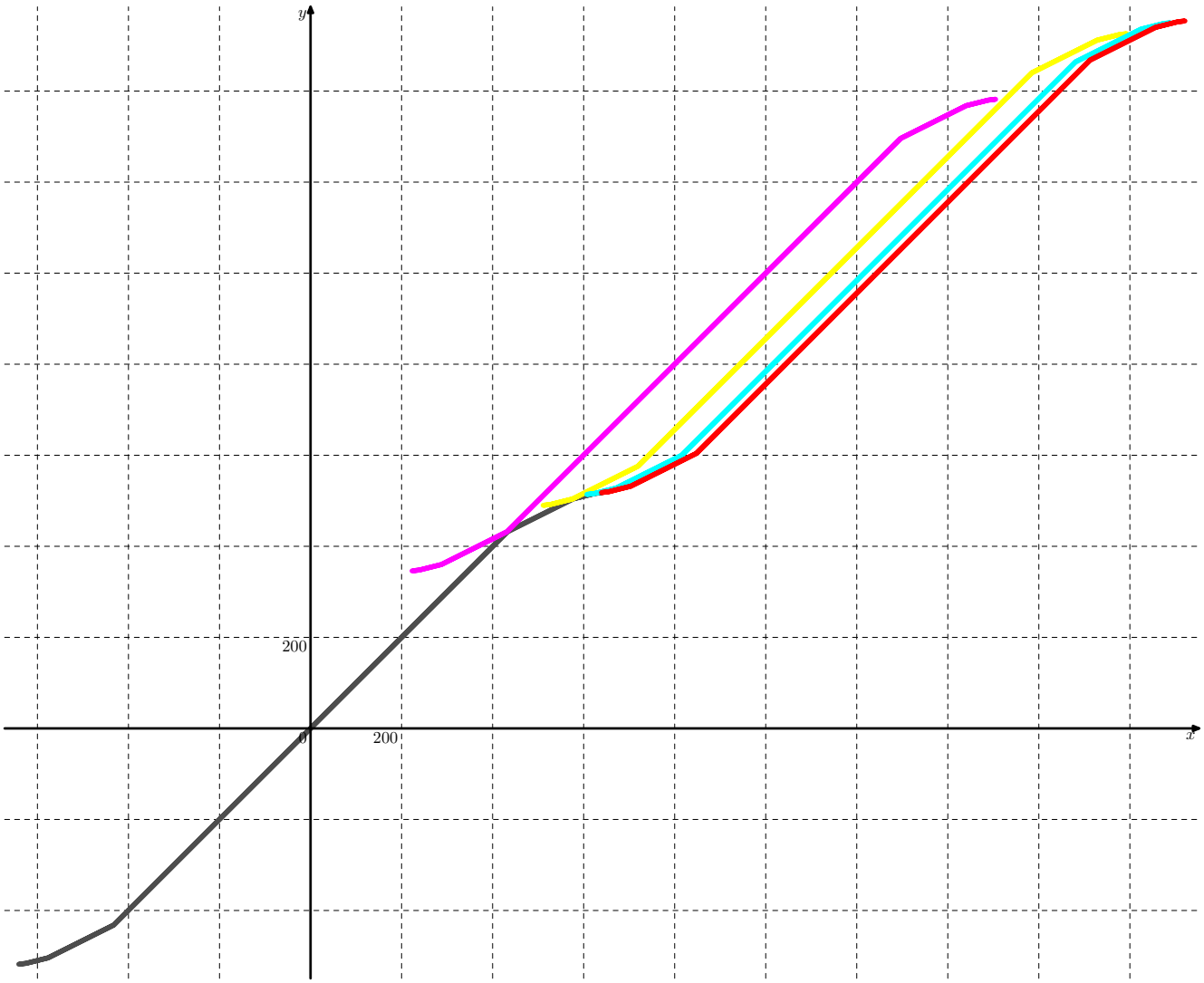

Рис. 15: Инь-ян, $n = 4$