Задача A. XOR-последовательность

Попробуем продолжить последовательность.

- $P_3 = A \oplus B$
- $P_4 = B \oplus (A \oplus B) = A$ (так как побитовое исключающее двух одинаковых чисел равно нулю)
- $P_5 = (A \oplus B) \oplus A = B$
- $P_6 = A \oplus B$

Получается, что элементы с 4 по 6 идентичны элементам с 1 по 3. Очевидно, что и дальше в последовательности эти 3 элемента будут чередоваться. Значит, мы можем взять остаток от деления N на 3 и вывести нужный элемент за $O(1)$.

Задача B. ПСП наносит ответный удар

Решение 1

Давайте заметим, что наша ПСП формируется схоже с деревом отрезков, тогда давайте будем считать кол-во скобок следующим образом:

1. Если l и r лежат в одной половине, то смотрим, если $l = 1$, то добавляем к ответу единицу и запускаемся для S_{N-1}.
2. Если l и r лежат в разных половинах, то смотрим, если $l = 1$, то добавляем к ответу единицу и запускаемся в две половины.
3. Если мы получили S_1 ПСП, то смотрим, если $l = 1$, то добавляем к ответу единциу и завершаем функцию.
4. Если l и r в точности равны границам S_N, то добавляем к ответу $\frac{|S_N|}{2}$.

Затем вызываем нашу функцию по описанию выше и выводим ответ на каждый запрос за $O(\log N)$, и тогда конечная асимптотика будет равна $O(Q \cdot \log N)$.

Решение 2

Давайте подсчитаем кол-во открывающихся скобок от 0 до $l - 1$ и от 0 до r, тогда заметим, что это похоже на префикс-сумму, ответом будет разность суммы с 0 по r и суммы с 0 по $l - 1$. Будем считать данную сумму следующим образом:

1. Если наша граница $x \geqslant \frac{|S_N|}{2}$, то к ответу добавляем $\frac{|S_N|}{2}$ и из x вычитаем $\frac{|S_N|}{2} + 1$, после этого рассматриваем ПСП S_{N-1}.
2. Если наша граница $x < \frac{|S_N|}{2}$, то к ответу добавляем единицу и вычитаем из x единицу, после этого рассматриваем ПСП S_{N-1}.
3. Если наша граница в точности равна $|S_N|$, то добавляем к ответу $\frac{|S_N|}{2}$ и возвращаем ответ.
4. Если наша ПСП равна S_1, то добавляем к ответу единицу и возвращаем ответ.
5. Если наша граница x равна нулю, то возвращаем ответ.

Тогда конечная асимптотика будет равна $O(Q \cdot \log N)$.

Задача C. Дуализм чисел

Давайте сначала решим задачу для чисел $c_i \geqslant 2$. Построим граф, где вершинами будут числа из массива и проведем ребро между вершинами, если сумма их чисел простая. Заметим, что такой граф будет двудольным, где вершины одной доли — четные, а другой — нечетные, так как между вершинами одной четности ребра быть не может, ведь иначе их сумма будет делиться на 2 (за исключением случая с двумя единицами, но сейчас мы решаем задачу для $c_i \geqslant 2$). В таком случае

минимальным требованием для того, чтобы разбить числа на пары, является равное количество вершин в четных и нечетных долях. Теперь найдем максимальное паросочетание в этом графе с помощью, например, алгоритма Куна. А потом проверим, что размер найденного паросочетания равен $\frac{N}{2}$.

Теперь обобщим задачу для $c_i \geqslant 1$. Изначально построим граф на всех числах, больших 1. Если после этого нечетных чисел в графе меньше, чем четных, то добавим в него некоторое количество единиц с соответствующими ребрами, не добавляя при этом ребра между единицами, чтобы количество четных и нечетных чисел уравнялось. Если заведомо четных чисел больше, чем нечетных, то ответа нет. Иначе также ищем максимальное паросочетание, а оставшиеся единицы ставим в пары друг с другом.

Задача D. Лестница

Будем хранить индексы всех элементов лестницы в порядке возрастания. После обновления p-го элемента найдем наибольший индекс idx такой, что $idx < p$. Теперь добавим p в текущее множество, если $p = 1$ или $a[p] > a[idx]$. При этом, если мы добавили текущий индекс в лестницу или он там уже находился, необходимо поддерживать возрастание самих значений лестницы. Для этого удалим из множества все индексы i такие, что $p < i$ и $a[p] \geqslant a[i]$. Все перечисленные операции мы можем реализовать с помощью структуры данных set или дерева отрезков. Так как за один запрос обновления мы добавляем не более одного элемента, количество операций удаления не превосходит $Q + N$. Следовательно, итоговая сложность решения $O(N + Q \cdot \log(N + Q))$.

Задача E. Прекрасные числа

Для начала найдем преподсчетом все красивые числа до 10^9. Троичная запись красивого числа не содержит цифр 2, а также имеет длину до 19 знаков. Тогда можно перебрать все такие троичные записи и проверить для каждой из них, является ли соответствующее число красивым. В результате окажется, что красивыми являются всего 40 чисел.

Теперь нужно решить классическую задачу о поиске подмножества с заданной суммой X. Обычно эта задача решается либо за $O(N \cdot X)$ с помощью динамического программирования, либо за $O(2^N)$ методом полного перебора, где N — количество красивых чисел. В текущей задаче оба способа будут работать слишком долго.

Ускорим алгоритм полного перебора с помощью техники meet-in-the-middle. Разобьем множество красивых чисел на две половины, для каждой из них посчитаем суммы всевозможных подмножеств, после чего будем искать разложение числа X на сумму двух чисел - по одному из каждой половины. Асимптотика такого решения будет равна $O(2^{\frac{N}{2}})$ на запрос, что все еще будет слишком долго для решения данной задачи.

Чтобы решить задачу полностью, нужно разбить множество красивых чисел на неравные части размерами 30 и 10 элементов соответственно. Чтобы найти всевозможные суммы подмножеств первой части, можно воспользоваться методом динамического программирования, так как сумма всех ее элементов равна $s = 4 \cdot 10^7$, а значит, дп будет работать за $O(s \cdot N)$. Чтобы еще ускорить подсчет, можно воспользоваться битмасками. Суммы второй части находим как и раньше с помощью полного перебора. Теперь чтобы найти ответ для числа X, переберем суммы S второй части и проверим, возможно ли собрать число $X - S$ элементами из первой части.

Кроме этого стоит отметить, что у задачи существует решение, основанное на рекурсивном полном переборе. Если рассмотреть множество красивых чисел, то окажется, что все числа там очень быстро растут. Из-за этого получается такая ситуация, что большинство чисел до 10^9 являются прекрасными. Если добавить несколько отсечений в рекурсию или кеширование ответов в рекурсии, то такой перебор тоже будет работать корректно.

Задача F. Вадим и судоку: пары Кропки

Скажем, что массив Кропки, если все пары соседних цифр — пары Кропки.

Будем решать методом динамического программирования. Пусть $dp[i][j]$ — наименьшее количество цифр, которое надо заменить, чтобы префикс длины i был массивом Кропки, а последняя его цифра равна j. $dp[1][j] = 1$, если $j \neq S[1]$, и $dp[1][j] = 0$, если $j = S[1]$.

$dp[i][j] = \min(dp[i-1][j \cdot 2], dp[i-1][j/2], dp[i-1][j-1], dp[i-1][j+1]) + 1$, если $j \neq S[i]$, и $dp[i][j] = \min(dp[i-1][j \cdot 2], dp[i-1][j/2], dp[i-1][j-1], dp[i-1][j+1])$, если $j = S[i]$. Очевидно, что если $j \cdot 2$, $j/2$, $j-1$ или $j+1$ не являются цифрой в промежутке от 1 до 9, то их не надо рассматривать в операции минимума.

Оптимальным количеством замен будет $\min(dp[|S|][j])$, где $j \in [1, 9]$. Осталось лишь построить массив целиком, это можно сделать проходя по переходам динамики в обратную сторону.

Задача G. Вадим и судоку: XV-пары

Давайте разберём несколько случаев:

1. Если $X = V$.
2. Если $X = V + 1$.
3. Если $X = V + 2 = 2 \cdot a$, при X — четное.
4. Если $X = V + 2 = 2 \cdot a + 1$, при X — нечетное.
5. Если $X = V + a$, при $a > 2$.

Для 1-го случая очевидно, что ответ — 2.

Заметим, что мы можем составить последовательность для 2-го случая следующим образом:

$$[X-1, 1, X-2, 2, \ldots, X-n, n]$$

Нетрудно доказать, что сумма соседних будет составлять X или V. Заметим, что на элементе $V+1$ мы зациклимся и попадём в элемент, который уже использовался.

Заметим, что мы можем составить последовательность для 3-го и 4-го случаев и следующим образом:

$$[X-1, 1, X-3, 3, \ldots, X-n, n]$$

Нетрудно доказать, что сумма соседних будет составлять X или V.

Рассмотрим 3-й случай и заметим, что на элементе $\frac{X}{2} + 1$ мы зациклимся, что не удолетворяет условию задачи, а значит, ответ на 3-й случай: $\frac{X}{2}$.

Рассмотрим 4-й случай и заметим, что на элементе $X - 1$ мы будем попадать в 0, что также не удолетворяет условию задачи, а значит, ответ на 4-й случай: $X - 1$.

Рассмотрим 5-й случай и заметим, что последоватеьность всегда будет выглядеть следующим образом:

$$[X-1, X-(X-1), \ldots, V-1, X-(V-1)]$$

Отсюда не трудно заметить, что на элементе $\lfloor \frac{X-3}{a} \rfloor \cdot 2 + 3$ мы попросту не можем поставить такое число, что удолетворяет условию задачи, так как оно будет отрицательным, а значит, ответ равен $\lfloor \frac{X-3}{a} \rfloor \cdot 2 + 2$.

Докажем 5-й случай:

Обозначим $z = a$. Скажем, что наша последовательность будет выглядеть следующим образом:

$$[y - 0 \cdot z, X - y + 0 \cdot x, y - 1 \cdot z, X - y + 1 \cdot z, \ldots, y - k \cdot z, X - y + k \cdot z]$$

Если $y = X - 1 \Rightarrow$ ответ будет $\lfloor \frac{X-2}{X-V} \rfloor \cdot 2 + 2$. Но если $X = f \cdot z + 2$, то нужно рассмотреть ещё случай:

Если $y = X - 1$, то можно составить следующую последовательность

$$[X-1, 1, X-1-z, 1+z, X-1-2 \cdot z, 1+2 \cdot z, \ldots, X-1-f \cdot z, 1+f \cdot z]$$

Так как последний элемент $(1 + f \cdot z) = X - 1$, y не может быть равен $X - 1$, таким образом, $y = X - 2$ и ответ в этом случае $\lfloor \frac{X-3}{X-V} \rfloor \cdot 2 + 2$.

Давайте докажем для не повторяющихся:

Если $X = f \cdot z + 2$, то $y = X - 2$, и тогда если $2 + a \cdot z = X - 2 - b \cdot z$, то $(a + b) \cdot z = X - 4$, а тогда $(X - 4)$ делится на z и $(f \cdot z + 2)$ делится на z, 2 делится на z, а $z > 2$.

Если $X = f \cdot z + q$ и $q \neq 2$, то $y = X - 1$, и тогда если $1 + a \cdot z = X - 1 - b \cdot z$, то $(a + b) \cdot z = X - 2$, и $X - 2$ делится на z, и $f \cdot z + q - 2$ делится на z, и $q - 2$ делится на z, но $q < x$ и $q - 2 \neq 0$. Поскольку $(X - 2) \not\vdots (X - V)$, то $\frac{X-2}{X-V} \cdot 2 + 2 = \lfloor \frac{X-3}{a} \rfloor \cdot 2 + 2$. Следовательно, формула одинаковая для всего 5-го случая.

Таким образом, ответ на 5-й случай будет $\lfloor \frac{X-3}{a} \rfloor \cdot 2 + 2$.

Задача N. Вадим и судоку: ренбан

Решение 1

Заметим, что отрезок $a[l, r]$ — ренбан, если $\max(a[l, r]) - \min(a[l, r]) = r - l$ и все числа на нём различны. Будем фиксировать r и находить количество подходящих l. Для начала предпосчитаем для каждой позиции r такое $lx[r]$, что в $a[lx[r], r]$ нет повторений и либо $lx[r] = 0$, либо в $a[lx[r]-1, r]$ есть повторения (это можно сделать двумя указателями и словарём (unordered_set) за $O(N)$).

Заведём структуру, которая будет хранить $\max(a[i, r]) - \min(a[i, r]) + i$ для отрезка $a[i, r]$ (обозначим это значение как $f(i, r)$). Тогда количество подходящих l для фиксированного r — количество значений r на отрезке $[lx[r], r]$ (это будет первым запросом для структуры).

После этого нужно перейти к $r + 1$ и изменить заведённую структуру. $f(r + 1, r + 1) = r + 1$. Если $a[r] = a[r + 1]$, то $f(i, r) = f(i, r + 1)$ для $i \in [0, r]$. Дальше есть два симметричных случая: пусть $a[r] < a[r + 1]$, тогда $\min(i, r) = \min(i, r + 1)$, значит, меняются только значения максимумов, рассмотрим их подробнее.

Заведём массив $MX_r[i] = \max(a[i, r])$ для $i \in [0, r]$ невозрастающий, который отличается от нужного $MX_{r+1}[i] = \max(a[i, r + 1])$ для $i \in [0, r + 1]$ только тем, что к нему применили «max =» $a[r + 1]$ и дописали $a[r + 1]$. Поскольку MX_r невозрастающий, то «max =» изменит только какой-то суффикс. Будем хранить массив MX_r в виде отрезков с равными значениями (например, вместо $[7, 6, 6, 6, 4, 4, 2]$ будем хранить $([7, 0, 0], [6, 1, 3], [4, 4, 5], [2, 6, 6])$, где каждый отрезок описывается тремя числами a_i, l_i, r_i — значение на отрезке, левая и правая граница отрезка), тогда, при добавлении элемента $a[r + 1]$ (который больше $a[r]$ по предположению), некоторые отрезки объединятся в один (например, если $a[r+1] = 5$, тогда получится $([7, 0, 0], [6, 1, 3], [5, 4, 7])$), тогда для каждого удалённого отрезка можно заметить, что сумма в структуре изменится на одинаковое значение, значит, можно добавить разницу на отрезке (это будет вторым запросом для структуры). Если же $a[r] > a[r + 1]$, то нужно просто добавить новый отрезок $[a[r + 1], r + 1, r + 1]$ и ничего обновлять в структуре не надо.

Заметим, что обновляем мы только при удалении/объединении отрезков длины не меньшей одного, а добавляется/объединяется в новый всегда только один, значит, всего удалённых отрезков будет $O(N)$. Понятно, что нужен аналогичный массив MN_r для обновления минимумов. Хранить эти массивы (отрезки с равными значениями) можно, например, на стеке.

Осталось определиться с структурой. Если она умеет выполнять оба запроса за $O(f(N))$, то всё решение работает за $O(N \cdot f(N))$. Это возможно сделать с помощью sqrt-декомпозиции со словарём (unordered_set) в каждом блоке за $O(N^{1.5})$.

Решение 2

Так же будем фиксировать r и находить количество подходящих l. Так же посчитаем для каждой позиции r соответствующее $lx[r]$. Будем рассматривать только отрезки $a[lx[r], r]$. По построению очевидно, что числа на нём не повторяются.

Рассмотрим произвольный элемент с значением x в этом отрезке. Построим *стрелку* из него, если на отрезке есть значение $x + 1$, и оно находится левее (тогда стрелка будет вести из индекса со значением x в индекс со значением $x + 1$), либо на отрезке нет значения $x + 1$ (тогда стрелка будет вести из индекса со значением x в -1). Если на отрезке есть значение $x + 1$, и оно находится правее, то стрелки не будет.

Построим все возможные стрелки для текущего отрезка. Заметим, что отрезок $a[l, r]$ является ренбаном тогда и только тогда, когда из него следует ровно одна стрелка в индекс $k < l$, причём начинаться она будет в наибольшем числе на этом отрезке.

Заведём дерево отрезков с операциями прибавления на отрезке и нахождения количества минимумов на отрезке. Построим его на массиве размера N и состоящего из нулей. Для каждой стрелки из ind_r в ind_l прибавим единицу на отрезке $[ind_l + 1, ind_r]$. Количеством подходящих l для зафиксированного r будет количество минимумов на отрезке $[lx[r], r]$.

Теперь научимся быстро переходить от r к $r + 1$. Заметим, что новым числом на отрезке $a[lx[r+1], r+1]$ по сравнению с $a[lx[r], r]$ будет только $a[r+1]$. Если на отрезке $a[lx[r+1], r+1]$ есть $a[r+1]-1$, тогда удалим стрелку из него, если была (пусть она была из ind_r в ind_l, тогда в дереве отрезков нужно вычесть единицу на отрезке $[ind_l + 1, ind_r]$). Если на отрезке $a[lx[r+1], r+1]$ есть $a[r+1]+1$, то проведём из $a[r+1]$ в него стрелку (сделаем прибавление на отрезке, как описано в предыдущем абзаце), в противном случае стрелка из $a[r+1]$ пойдёт в -1 (так же сделаем прибавление на отрезке). Таким образом, мы перестроили стрелки для отрезка $a[lx[r+1], r+1]$ из стрелок отрезка $a[lx[r], r]$.

Поиск $a[r+1]-1$ и $a[r+1]+1$ из предыдущего абзаца делается легко, если переходить двумя указателями от $a[lx[r], r]$ к $a[lx[r+1], r+1]$.

Таким образом, суммарная сложность решения будет равна $O(N \cdot \log N)$.

Задача I. Вадим и судоку: суммы регионов

Давайте заметим, что сумма в каждом регионе одинакова, тогда мы можем представить общую сумму в виде $n \cdot m$, где n — количество регионов и m — сумма в каждом регионе. Мы будем считать регионы успешно поделёнными тогда и только тогда, когда мы сможем найти все суммы $1 \cdot m$, $2 \cdot m$, …, $n \cdot m$. Если мы смогли найти такие числа в массиве префиксных сумм, значит, мы можем разделить на регионы суммы m. Ответом будет разделение по минимальному такому числу.

Задача J. Нужно больше трасс

Заметим, что если все города лежат на одной прямой, то ответ $N - 1$.

Заметим, что результатом будет планарный граф, причём все дороги, проходящие по выпуклой оболочке, должны быть построены. Пусть на выпуклой оболочке находится K городов (заметьте, что тут важно не количество вершин многоугольника выпуклой оболочки, а именно количество точек на нём).

Поскольку граф планарный, для него выполняется формула Эйлера $V - E + F = 2$, где V — количество вершин (городов), E — количество рёбер (построенных дорог), F — количество граней. Заметим, что существует грань, у которой K вершин (простыми словами, внешняя грань). Нетрудно показать, что если провести наибольшее количество дорог, все остальные грани будут треугольниками (пусть это не так, то существует грань, которая является многоугольником без самопересечений, у которого больше трёх вершин, в таком всегда можно провести диагональ, лежащую целиком внутри этого многоугольника и не пересекающую никакую из других сторон; таким образом многоугольник распался на два, если продолжать такой процесс, то все грани будут треугольниками).

Также заметим, что каждое ребро является частью ровно двух граней, это значит, что

$$2 \cdot E = K + 3 \cdot (F - 1) \Rightarrow F = \frac{2 \cdot E - K + 3}{3}$$

Также $V = N$, подставим это в формулу Эйлера:

$$N - E + \frac{2 \cdot E - K + 3}{3} = 2 \Rightarrow E = 3 \cdot N - 3 - K$$

Задача K. Нужно больше задач

В задаче требовалось научиться выполнять на корневом дереве два типа запросов: прибавлять число ко всем вершинам, лежащим в поддереве заданной вершины и отличающихся от неё по высоте не больше, чем на k, и отвечать, какое число в настоящий момент записано в вершине.

Выпишем все вершины в порядке обхода в глубину, построим на этом обходе дерево отрезков, в каждом узле которого заведем ещё одну структуру данных, изначально пустую. Поддерево каждой вершины представляет собой отрезок обхода, заданный временем входа и выхода данной вершины,

который в свою очередь разбивается на $\mathcal{O}(\log N)$ узлов дерева отрезков. При прибавлении в структуру данных внутри каждого из этих узлов добавим пару (*высота вершины запроса* $+ k$, x). Тогда при ответе на запрос о числе, записанном в вершине, нам необходимо пройтись по всем предкам узла, соответствующего этой вершине в дереве отрезков, и просуммировать вторые числа во всех парах, в которых первое число не меньше, чем высота нашей вершины.

Примеры внутренних структур данных, способных быстро выполнять такие операции добавления и суммы: любая реализация дерева поиска (например, декартово дерево) или дерево отрезков/Фенвика с локальным сжатием координат внутри каждого узла. Итоговая асимптотика — $\mathcal{O}(Q \log^2 N)$

Альтернативным подходом является корневая декомпозиция по запросам. Разобьем все запросы на блоки по корень, при переходе между каждыми двумя блоками за $\mathcal{O}(N + Q)$ построим числа, которые были записаны в вершинах до начала нашего блока. А по всем изменениям, случившимся внутри блока, будем пробегать при ответе на каждый запрос. За $\mathcal{O}(1)$, используя времена входа/выхода и высоты вершин, мы можем определить затронуло ли это изменение нашу вершину или нет. Асимптотика этого подхода — $\mathcal{O}(Q\sqrt{N})$

Задача L. Бить без разбора

Очевидно, что, чтобы ладья била максимальное количество клеток, нужно чтобы она стояла на вертикали x и горизонтали y таких, что ver_x и hor_y максимальны, где ver_x — количество клеток на вертикали x, а hor_y — количество клеток на горизонтали y. Тогда, если мы выбрали такие вертикаль x и горизонталь y, ответом будет $ver_x + hor_y$, если клетка (x, y) не принадлежит исходному множеству, и $ver_x + hor_y - 1$ если принадлежит. Давайте найдем все такие x и y и простым перебором найдем любую пару, которая не принадлежит исходному множеству, а если не нашли, то выберем любую пару с ответом $ver_x + hor_y - 1$. Таким образом, в худшем случае (если все рассматриваемые клетки принадлежат множеству) мы рассмотрим N позиций.

Задача M. Студент, или Туда и обратно

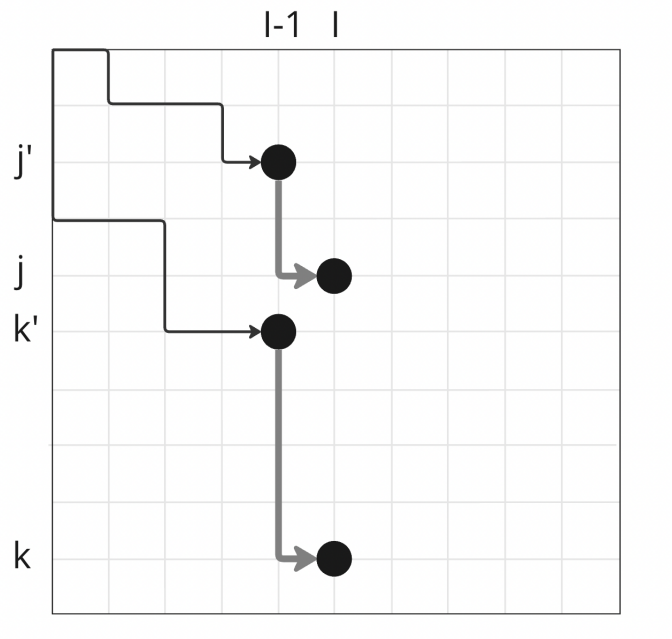
Решим задачу, когда студент движется только в одну сторону, но сразу двумя путями, которые не пересекаются между собой. Пусть один из них идёт строго выше другого. Посчитав число таких пар путей, мы бы нашли ответ, но делённый на 2.

Будем считать динамику $dp_{i,j,k}$. $dp_{i,j,k}$ — число пар таких путей, что верхний путь впервые пересёк столбец с индексом i в точке (j, i), а нижний впервые пересек столбец i в точке (k, i). Если научимся считать такие значения, то ответом на задачу будет: $2 \cdot \sum_{i=1}^{i<n} dp_{m,i,n}$.

Изначально зададим $dp_{1,1,k} = 1$ для всех $1 < k \leqslant n$ и $dp_{1,j,k} = 0$ для всех $1 < j, k \leqslant n$. Далее будем перебирать i от 2 до n и пересчитывать $dp_{i,j,k}$. Если $j \leqslant k$, то $dp_{i,j,k} = 0$, потому что в таком случае верхний путь должен был бы где-то пересечь нижний. Для $j < k$ переберем, в какой строке могли бы впервые пересечь $i-1$ столбец верхний и нижний путь и прибавим это значение для этого состоянии динамики к $dp_{i,j,k}$. Пусть это будут индексы j' и k' для j и k соотвественно. Понятно, что $k' > j$, иначе пути бы пересеклись в точке $(j, i-1)$. Кроме этого, понятно, что все дороги из $(j', i-1)$ в $(j, i-1)$ и из $(j, i-1)$ в (j, i) должны существовать, иначе пути пересекли бы столбец i где-то ранее, чем в строке j. Аналогично для нижнего пути и переменных k' и k.

Из этого всего мы получили алгоритм, который работает за $O(n \cdot m^5)$: перебираем i, j, k, j', k' и еще проверяем, что на пути из $(j', i-1)$ в (j, i) и из $(k', i-1)$ в (k, i), есть дороги. Долго. Но его можно ускорить, если научиться быстро находить сумму динамик, которые подпадают под все ограничения, которые описали ранее.

Можно заметить, что $dp_{i,j,k} = \sum_{j'=x,k'=y}^{j' \leqslant j, k' \leqslant k} dp_{i,j',k'}$, для некоторых x и y, которые равны минимальной строке, в которой могли пересечь $i-1$ столбец в первый раз нижний и верхний маршрут соотвественно. Чтобы их найти, достаточно найти первую дорогу, которая вертикальная и отсутствует в предыдущем столбце и при этом выше соответствующего перекрестка в i-м столбце. Это делается простым предподсчётом. А сумма есть ни что иное, как сумма в некотором прямоугольнике с одной вершиной в точке (j, k) и противоположной вершиной в точке (x, y), где в вершинах — $dp_{i-1,j',k'}$. Сумму в прямоугольнике можно найти предподсчётом префикс-сумм в прямоугольниках $((1, 1), (j, k))$ и методом включений и исключений.



Задача N. Парень и колдунья

Для удобства перевернём таблицу горизонтально (то есть так, что $M \leqslant N$). Рассмотрим все возможные значения M

- Если $M = 1$, то парень сможет только 1 раз сходить направо, то есть ответ равен 2. Разумеется, это выполняется, только если $N > 1$. При $N = 1$ парень не сможет никуда пойти и ответ будет равен 1.
- Если $M = 2$, то парень сможет пройти всю таблицу «змейкой». Ответ — $2 \cdot N$.
- Теперь рассмотрим случай, когда $M = 3$. Обозначим столб из трёх клеток как тройку. Заметим, что парень будет чередовать ходы по вертикали и горизонтали. Очевидно, пока парень не будет ходить налево, в каждой тройке клеток он посетит две из трёх, причём в каждой тройке точно будет посещена средняя клетка. При этом ход налево станет последним (он не сможет снова пойти налево или направо, так как только что ходил по горизонтали, а остальные клетки в этой тройке уже были посещены, следовательно, сходить вниз или вверх парень также не сможет). Таким ходом мы сможем увеличить ответ всего на 1, поэтому нам выгоднее сперва дойти до правого края таблицы. Нетрудно заметить, что для максимизации ответа первый ход должен быть вверх. Так как парень будет чередовать ходы по горизонтали и вертикали, каждые 2 хода он будет менять координаты по вертикали и горизонтали, то есть после чётных ходов на чётных тройках он будет стоять на средней клетке, а на нечётных — на одной из крайних. Чтобы сходить налево, ему перед этим придётся сходить по горизонтали. После такого хода на нечётных тройках он окажется на средней клетке. Но средняя клетка левой тройки уже посещена, так что парень сходить налево не сможет. И получается, что парень может пойти налево только когда N нечётно. Получается, что при чётном N ответ равен $2 \cdot N + 1$, а при нечётном — $2 \cdot N$.