Разбор задачи «Закономерности»

Пройдём по ячейкам — числам от 1 до n^2 — и для каждого числа i найдём количество делителей: переберём потенциальные делители от 1 до i и проверим каждый из них. (Также можно проверять делители от 1 до $\sqrt{i}$ и вместе с каждым делителем j учитывать парный ему делитель i/j, аккуратно учитывая возможный делитель $\sqrt{i}$.)

Разбор задачи «Интересная экскурсия»

Сформулируем задачу в терминах теории графов. Нам задан ориентированный граф, рёбра которого покрашены в m цветов. Необходимо в нем найти рёберно-простой цикл, в котором все соседние ребра имеют различные цвета.

Сначала опишем решение за $O(m^2)$. Построим новый граф, в котором вершинами будут рёбра изначального графа. Из одного ребра можно перейти в другое, если конец первого ребра совпадает с началом второго, а их цвета различны. Любой цикл в таком графе будет соответствовать искомому циклу в изначальном графе. Однако количество переходов в таком графе может быть порядка m^2, для получения решения с асимптотикой $O(m)$ переходы следует строить более эффективно.

Рассмотрим построение переходов для одной вершины v. Упорядочим все исходящие из v рёбра по цвету. Обозначим их количество за d_v. Теперь все рёбра, имеющие определенный цвет c, образуют подряд идущий отрезок в этом отсортированном массиве, а все рёбра всех остальных цветов — объединение какого-то префикса и какого-то суффикса этого массива.

Для каждой вершины v создадим $2d_v$ вершин $\texttt{prefix}_i$ и $\texttt{suffix}_i$. Проведем ребра из $\texttt{prefix}_i$ в $\texttt{prefix}_{i-1}$, а также в i-е ребро отсортированного списка. Аналогично, из $\texttt{suffix}_i$ проведем рёбра в $\texttt{suffix}_{i+1}$ и в i-е ребро. При таком построении из вершины $\texttt{prefix}_i$ будут достижимы все рёбра с номером не больше i, а из вершины $\texttt{suffix}_i$ — все рёбра с номером не меньше i.

Рассмотрим ребро $u \to v$ цвета c. Пусть ребра, исходящие из v и имеющие цвет c , находятся на отрезке $[l, r]$. Добавим переходы из этого ребра в $\texttt{prefix}_{l-1}$ и $\texttt{suffix}_{r+1}$. Таким образом, из данного ребра можно будет перейти во все ребра, цвет которых либо меньше c, либо больше c. В таком графе следует найти цикл, номера вершин, которые соответствуют ребрам изначального графа будут образовывать искомый чередующийся цикл. Вспомогательные вершины $\texttt{prefix}_i$ и $\texttt{suffix}_i$ следует из найденного цикла выбросить.

Оценим количество вершин и рёбер в новом графе. Для каждой вершины v мы добавили $2d_v$ новых вершин, из каждой из которых исходит не более двух рёбер. Сумма всех d_v равна количеству рёбер m. Таким образом, в получившемся графе $3m$ вершин и не более $6m$ ребер. Нахождение цикла в таком графе можно реализовать, например, с помощью обхода в глубину, что занимает $O(m)$ времени.

Разбор задачи «Прыжки с поворотом»

Заметим, что если длина одной из сторон равна 1, а другой — больше либо равна 3, то решения не существует, потому что вне зависимости от порядка будет три подряд идущие точки, лежащие на одной прямой. Случаи 1×1 и 1×2 имеют тривиальное решение.

Научимся конструктивно строить решение для всех остальных случаев. Изначально встанем в угол прямоугольника и научимся прыгать таким образом, чтобы побывать во всех клетках первого и последнего столбца, а потом прыгнуть в угол оставшегося прямоугольника. При этом, чтобы весь прямоугольник лежал в левой полуплоскости относительно последнего прыжка. То есть, чтобы следующим ходом можно было прыгнуть в любую клетку. Не умаляя общности будем считать, что изначально мы стоим в нижнем левом углу. Будем прыгать в таком порядке: самая нижняя непосещенная клетка в первом столбце, самая нижняя непосещенная клетка в последнем столбце, самая верхняя непосещенная клетка в последнем столбце, самая верхняя непосещенная клетка в первом столбце. Несложно показать, что такая последовательность прыжков удовлетворяет условию. Если высота нечетна, то последняя посещенная клетка в этой последовательности будет в последнем столбце, и тогда следующим ходом надо прыгнуть в верхнюю клетку предпоследнего столбца. А если высота четна, то мы остановимся в первом столбце, и следующим ходом прыгнем в нижнюю клетку второго столбца.

Аналогичным образом можно посетить верхнюю и нижнюю строки.

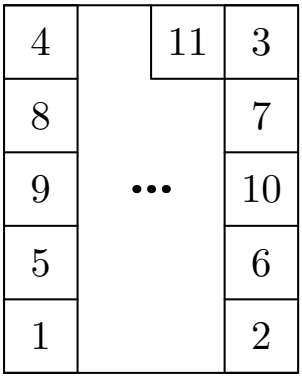

Рис. 1: Нечетная высота

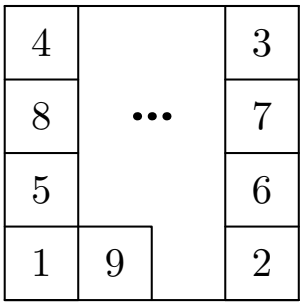

Рис. 2: Четная высота

Мы научились переходить к случаю, в котором одна из сторон короче на 2. Теперь, если одна из сторон четная, то будем уменьшать ее длину на 2, пока она не станет равна 0. Если же обе стороны нечетные, то уменьшим их до тех пор, пока они не станут равны 3, а потом решим конструктивно для прямоугольника 3×3.

4	8	3
5	9	7
1	6	2

Разбор задачи «Подсчеты в строю»

Будем решать задачу только для людей, которые смотрят влево (для остальных ответ находится аналогично).

Будем рассматривать солдат по очереди слева направо. Будем поддерживать список солдат $1 \leqslant j_1 < j_2 < \ldots < j_k \leqslant i$, которых видит рассматриваемый солдат с номером i. Посмотрим, как этот список изменится при переходе к солдату $i+1$: солдат $i+1$ видит солдата i, а некоторых солдат из списка $j_1, j_2, \ldots, j_k$ теперь не видит $i+1$, так как ему мешает солдат i. Несложно заметить, что $h_{j_1} \geqslant h_{j_2} \geqslant \ldots \geqslant h_{j_k}$, поэтому солдаты, которых перестал видеть солдат $i+1$ — подряд идущий отрезок в конце списка.

Следовательно, можно поддерживать номера видимых солдат в структуре данных «стек».

Разбор задачи «Разные цифры»

Выберем старший разряд, в котором итоговое число будет отличаться от изначального (возможно, этот разряд — левее старшей цифры изначального числа, как в примере $98 \to 101$). Это самый правый из разрядов, для которых верны два условия:

а) Слева от этого разряда в изначальном числе нет двух одинаковых цифр подряд (иначе эта пара останется и в итоговом числе).

б) Цифру в этом разряде можно увеличить на 1 — а если она при этом совпадёт с цифрой слева от неё, то на 2, — и при этом она не должна, конечно, превысить 9.

В этом найденном разряде цифру изначального числа увеличим на 1 или на 2 согласно пункту «б)». После чего все цифры справа от этого разряда заполним нулями и единицами по очереди, начиная с нуля.

Разбор задачи «Рисование»

Одна из возможных конструкций выглядит следующим образом.

В первой большой строке закрасим полностью 4-ю строку. Во всех остальных больших клетках закрасим первый столбец. Таким образом в каждой большой клетке будет закрашено ровно 4 маленькие, а вся закрашенная фигура будет связной.

Теперь закрасим в каждой большой клетке оставшееся нужное количество маленьких клеток. Для больших клеток из первого ряда будем вначале закрашивать клетки в третьем столбце, а потом во втором. Для остальных больших клеток, будем вначале закрашивать клетки второго столбца, а потом третьего.

Заметим, что закрашенная фигура осталась связной, так как каждая новая закрашенная клетка находится рядом с уже закрашенной. Можно доказать, что незакрашенная фигура также будет связной.

Разбор задачи «Последняя битва»

Фактически в задаче дана перестановка $a_1, a_2, \ldots, a_n$, и требуется найти на какое минимальное число элементов k надо циклически сдвинуть нашу перестановку влево так, чтобы в ней не оказалось позиции i, число на которой равно i.

Это условие можно записать так: ни для какого i не выполняется, что $i - a_i \equiv k \pmod{n}$. Заметим также, что после n сдвигов перестановка перейдет в себя, а значит ответ, если он есть, лежит в пределах от 0 до $n-1$. Каждое число запрещает i нам ровно один циклический сдвиг от 0 до $n-1$.

Найдем все запрещенные сдвиги, переберем возможный сдвиг от 0 до $n-1$, если его нет в булевом массиве запрещенных, то он является ответом. Если же все сдвиги запрещены, то ответа нет, и выведем -1.

Разбор задачи «Расписание»

Для каждого дня нам интересно только максимальное время, когда Антон может проснуться, а это время зависит только от того, когда в этот день у него первый урок. Пусть в какой-то день у Антона первый урок — это урок с номером j. Тогда в этот день он может проснуться максимум в $\min(10, 5 + j)$ часов. Если у него нет уроков в этот день или это суббота или воскресенье, то в этот день он может проснуться максимум в 10 часов. Количество часов, которое Антон может спать в конкретный день, совпадает с максимальным временем, в которое он может проснуться, ведь засыпает он в 12 часов ночи.

Будем поддерживать, сколько часов недосыпа есть у Антона. Изначально это 0, а затем каждый день происходит следующее: к этому числу добавляется 8, и вычитается минимум из количества часов, которое он может спать в этот день, и количества часов, сколько у него есть недосыпа. То есть если у него было t часов недосыпа, а в этот день он может спать a часов, то после этого дня у него будет $t + 8 - \min(t + 8, a)$ часов недосыпа. Посчитаем, сколько у Антона будет недосыпа после одной недели учёбы и после двух. Пусть это будет x и y часов соответственно. Если во время того, как мы считаем эти числа, где-то недосып стал хотя бы k часов, то надо вывести этот день. Если $x \geqslant y$, то и после остальных недель у него будет недосып не больше, чем после первой, поэтому он всё время не будет пропускать уроки. Заметим, что это не тоже самое, что проверить, что $x = 0$, ведь в понедельник второй недели он мог спать больше, чем в понедельник первой недели, чтобы уменьшить недосып за первую неделю. Если же $x < y$, то за каждую неделю будет накапливаться дополнительный недосып $y - x$, а значит можно запустить цикл по всем дням, и посчитать, когда же Антон проспит свой первый урок.

Разбор задачи «Пицца»

Найдём ответ с помощью формулы включений-исключений. (Прочитайте о ней, чтобы понять дальнейший текст.)

Перебираем, какое подмножество друзей не обрадуется. Из необрадованности друга точно узнаем, присутствуют ли ингредиенты из его списка пожеланий. Если не нашли противоречий, то добавляем к ответу 2^k с нужным знаком (зависящим от чётности количества друзей в подмножестве), где k — количество ингредиентов, незадействованных в списках пожеланий друзей из подмножества (каждый из этих ингредиентов можно как включить в рецепт, так и не включать).

Если перебирать все возможные подмножества в цикле, то решение будет работать за время $O(2^m \cdot \sum a_i)$, что недостаточно эффективно.

Заметим, что более эффективен рекурсивный перебор масок: список пожеланий друга i будет просмотрен в таком случае всего лишь 2^i раз вместо 2^m. Получаем решение за $O(2^m \cdot \max(a_i))$, это укладывается в ограничения.

Альтернативное решение использует битовое сжатие (зато не использует ограничение на a_i). В процессе рекурсивного перебора будем поддерживать два битовых вектора: для хороших (T) и плохих (F) ингредиентов. Кроме того, заранее вычислим аналогичные векторы для входных списков пожеланий. Для заданного подмножества друзей следует сделать две вещи: проверить наличие противоречий и найти число незадействованных ингредиентов. Первое эквивалентно проверке того, что битовое «И» T и F пусто. Второе легко делается подсчётом числа единичных битов в T и F. Сложность такого решения: $O(2^m \cdot n/32)$, где 32 — количество битов, обрабатываемых за одну операцию.

Разбор задачи «Ретвинтим twinter»

Переберём длину числа m (количества твинтов в ответе), например, начиная с 1.

При фиксированной длине числа m будем жадно наполнять твинты новой цепочки словами, при этом оставляя справа место для концовки «(i/m)», причём i мы в этот момент знаем, а для m просто отводим известное нам число символов.

Если количество твинтов в набранной таким образом цепочке имеет ровно ту длину, которую мы предположили в этой итерации, это ответ на задачу. Иначе этой длины числа m не хватает, а значит, надо переходить к следующей возможной длине.

Разбор задачи «Дробление»

Переупорядочим числа по возрастанию. Будем, считать, что $a \leqslant b \leqslant c \leqslant d$. Заметим, что в ответе, числитель обеих дробей меньше, чем знаменатель, так как иначе, можно поменять их местами и значение суммы уменьшится. Таким образом, есть три варианта ответа $\frac{a}{b}+\frac{c}{d}$, $\frac{a}{c}+\frac{b}{d}$, $\frac{a}{d}+\frac{b}{c}$. Убедимся, что второй из них минимальный.

$$\frac{\frac{a}{b}+\frac{c}{d}}{\frac{a}{c}+\frac{b}{d}} = \frac{c(\frac{a}{bc}+\frac{1}{d})}{b(\frac{a}{bc}+\frac{1}{d})} = \frac{c}{b} \geqslant 1$$

,

$$\frac{a}{d}+\frac{b}{c}-\left(\frac{a}{c}+\frac{b}{d}\right) = \frac{ac+bd-ad-bc}{cd} = \frac{a(c-d)-b(c-d)}{cd} = \frac{(a-b)(c-d)}{cd} \geqslant 0$$

.

Альтернативное решение — перебрать все перестановки из четырёх элементов, посчитать соотвествующее значение и выбрать минимум.