Разбор задачи «Заворожили кота»

Автор и разработчик задачи: Анатолий Максудов

Заметим, что количество целых чисел от 1 до m, которые кратны k, равно $\left\lfloor \frac{m}{k} \right\rfloor$. Следовательно, количество целых чисел от l до r, которые кратны k, равно $\left\lfloor \frac{r}{k} \right\rfloor - \left\lfloor \frac{l-1}{k} \right\rfloor$.

Следовательно, количество подходящих по условию задачи положительных чисел равно $\left\lfloor \frac{10^n - 1}{2^x} \right\rfloor - \left\lfloor \frac{10^{n-1} - 1}{2^x} \right\rfloor$. Число 0 будет подходить при $n = 1$ и любом x. Асимптотика такого решения $\mathcal{O}(1)$.

Для решения подзадач 1, 2 и 3 можно было отдельно рассмотреть расчёт количества чисел, кратных 2, 4 и 8 соответственно. В подзадачах 4 и 6 можно было в цикле пройтись по подходящим числам с шагом 2^x. В подзадаче 5 первое подходящее число равно 10^{n-1}, причём $10^{n-1} > 0$.

Разбор задачи «СуперКопилка»

Автор и разработчик задачи: Всеволод Лепешов

Заметим, что если возможно применить x операций, то также возможно и применить $x - 1$ операцию. Поэтому мы можем решать задачу техникой бинарного поиска по ответу.

Осталось научиться определять: возможно ли сделать x операций?

Пускай это возможно, и способ это сделать, предполагает c_1 операций от партнёра 1, c_2 операций от партнёра 2, ..., c_n операций от партнёра n. Тогда, чтобы это было возможно, все ограничения на массив c таковы:

- $c_1 + c_2 + \ldots + c_n = x$
- $c_i \geqslant 0$
- $c_i \leqslant a_i$
- $x - c_i \leqslant b_i$

То есть, на каждое c_i есть ограничение: $max(0, x - b_i) \leqslant c_i \leqslant a_i$. Поэтому если для какого-то i окажется, что $max(0, x - b_i) > a_i$, сделать x операций невозможно, так как нет подходящего значения c_i. Иначе заметим, что $\sum c_i$ может принять любое значение от $\sum max(0, x - b_i)$ до $\sum a_i$. Поэтому если $\sum max(0, x - b_i) \leqslant x \leqslant \sum a_i$, возможно выбрать c_i такие, что $\sum c_i = x$, и значит возможно сделать x операций. А иначе $\sum c_i = x$ добиться невозможно, значит и сделать x операций невозможно. Эта проверка выполняется за $O(n)$, и асимптотика всего решения $O(n \log A)$, где $A = 2 \cdot 10^9$, максимально возможный ответ на задачу.

Разбор задачи «В школу по снегу»

Автор и разработчик задачи: Иван Белков

Подгруппа 1. $l = 1, dt = 0$

В этой подгруппе количество тепла при проходе по любому переходу не меняется. Тогда задача сводится к задаче о кратчайшем пути.

В этой подгруппе $l = 1$, следовательно bfs найдёт такой кратчайший путь за $O(n + m)$.

Подгруппа 2. $dt = 0$

Как и в подгруппе 1 задача сводится к задаче о кратчайшем пути. Но теперь переходы могут быть произвольной длины, а значит в этой подгруппе задачу о кратчайшем пути стоит решать алгоритмом Дейкстры. Получим решение за $O(m \cdot log(n))$.

Подгруппа 3. $dt > 0$

В этой подгруппе количество тепла может только увеличиваться. Давайте пересчитывать $dp[t][u]$ - длина кратчайшего пути в перекрёсток u с количеством тепла t. Тогда $dp[t][u]$ пересчитывается по переходам из значений с меньшим t.

Переберём возможные значения t от 0 до 30, пересчитаем длины кратчайших путей по переходам из уже рассмотренных состояний.

Получим решение за $O(61 \cdot (n + m))$.

Подгруппа 4. Граф ацикличен

В этой подгруппе в графе нет циклов.

Найдём топологическую сортировку граф — такой порядок вершин, что все рёбра ведут слева направо. В данном случае - такой порядок перекрёстков, что все переходы ведут из перекрёстка с меньшим номером в перекрёсток с большим номером.

Давайте пересчитывать $dp[u][t]$ - длина кратчайшего пути в вершину u с количеством тепла t. Тогда $dp[u][t]$ пересчитывается по переходам в порядке топологической сортировки перекрёстков (в таком порядке все переходы будут вести из перекрёстков с уже посчитанным значением dp).

Получим решение за $O(61 \cdot (n + m))$.

Подгруппа 5. $n, m \leqslant 10^5, -30 \leqslant dt \leqslant 30$

Решаем задачу о кратчайшем пути, причём хотим посчитать $dist[u][t]$ для каждого перекрёстка и каждого значения количества тепла.

Заметим, что множество состояний не слишком большое: всего различных значений t - 61.

Построим граф, где вершина u_t будет означать такое состояние: мы находимся в вершине u и наше количество тепла равно t.

Для перехода $u\ v\ l\ dt$ будем добавлять в наш граф ребро длины l, ведущее из u_t в v_{t+dt} для всех значений t.

В полученном графе остаётся найти кратчайший путь из вершины 1_0 в одну из вершин n_t для некоторого t. Такую задачу решаем запуском алгоритма Дейкстры из вершины 1_0.

Получим решение за $O(61 \cdot (n + m) \cdot log(n))$.

Разбор задачи «Новогодние эксперименты»

Автор: Руслан Капралов, разработчик задачи: Ильнур Валеев

Подзадача 1. $n, q \leqslant 10^3$**,** $t \leqslant 5$**, все типы операций**

Для решения данной подзадачи можно было реализовать описанный процесс. Обрабатывать запросы 1–3 можно за $O(n)$, для ответа на операцию 4 нужно обратиться к ячейке массива за $O(1)$. Чтобы ответить на запрос 5, необходимо создать копию массива и отсортировать её.

Подзадача 2. $n \cdot q \leqslant 10^8$**,** $t \leqslant 2$**, все типы операций**

Для решения данной подзадачи нужно было ускорить предыдущее решение, а именно научиться отвечать на запрос 5 за $O(n)$. Для этого используется алгоритм поиска k-й порядковой статистики за линейное время.

Подзадача 3. $q \leqslant 4 \cdot 10^4$, $t \leqslant 5$, $\max a_i < 2^{10}$

Различных чисел в массиве a не более 2^{10}. Воспользуемся этим для оптимизации предыдущего решения. Создадим массив d, где $d[x] = x$. Теперь операции типов 1–3 будем выполнять на массиве d за 2^{10}. Для ответа на запрос 4 используем значение в массиве $a[i]$ как индекс для массива d, то есть $d[a[i]]$. Отвечать на запрос 5 можно за 2^{10}.

Подзадача 4. Нет операции 5-го типа

Для данной подзадачи нужно построить булевую функцию, чтобы отвечать на запросы типов 1–4 за $O(1)$.

Подзадача 5. $n, q \leqslant 2 \cdot 10^5$**, нет операции 3-го типа**

Рассмотрим, как влияют битовые операции на числа. **AND** и **OR** одинаково влияют на числа. Для конкретного бита i:

- **OR** с единицей или **AND** с нулём «схлопывает» бит i.

Такое «схлопывание» происходит не более одного раза для каждого бита. Если после очередной операции какой-то бит «схлопывается», пересчитываем массив заново. Таким образом, массив перестроится не более $\log(A)$ раз. Для ответа на запрос 5 создаём копию и сортируем массив только в случае, если до этого были операции **AND** и **OR**, которые схлопнули хотя бы один бит.

Подзадача 6. $n, q \leqslant 2 \cdot 10^5$**, нет операций 1-го и 2-го типов**

Для решения этой задачи удобно использовать структуру данных — двоичный бор. Храним массив чисел a в боре, построенном от старших битов к младшим. В таком виде числа в дереве будут отсортированы. Если в каждой вершине бора хранить количество чисел в поддереве, можно находить k-й наибольший элемент в массиве за $\log(A)$, где A — максимальное число в массиве.

Теперь рассмотрим влияние битовых операций на бор.

- **XOR** с нулём не изменяет значения битов и структуру дерева.
- **XOR** с единицей меняет местами детей у каждой вершины бора на глубине i.

Для каждой глубины i достаточно поддерживать флажок, указывающий, в какую из дочерних вершин ведёт нулевой бит.

Решение на полный балл

AND и **OR** одинаково влияют на бор. Для конкретного бита i:

- **OR** с единицей или **AND** с нулём «схлопывает» бит i, то есть на глубине i все вершины будут иметь ровно одну дочернюю вершину.

Подобное «схлопывание» происходит не более одного раза для каждого бита. Если после очередной операции какой-то бит «схлопывается», просто перестраиваем бор. Таким образом, бор перестраивается не более $\log(A)$ раз.

Итоговая асимптотика — $O(n \log^2(A) + q \log(A))$.

Разбор задачи «Парад планет»

Автор и разработчик задачи: Всеволод Лепешов

Для начала решим задачу при $\alpha = -1$. Для этого надо заметить свойство операции mod: если $a_i < a_{i+1}$, то $a_i \bmod a_{i+1} = a_i$. Поэтому, если в перестановке найдётся индекс $a_{p_i} < a_{p_{i+1}}$, то вся сумма $a_{p_1} \bmod a_{p_2} + \ldots + a_{p_{n-1}} \bmod a_{p_n}$ будет $\geqslant a_{p_i}$, а значит и $\geqslant \min(a_1, a_2, \ldots, a_n)$. Таким образом,

единственные перестановки, для которых возможно выполнение условие при $\alpha = -1$ такие, в которых $a_{p_1} \geqslant a_{p_2} \geqslant \ldots \geqslant a_{p_n}$. Ясно, что уникальный массив $\{a_{p_1}, a_{p_2}, \ldots, a_{p_n}\}$ с таким свойством только один: отсортированный по неубыванию массив a. А количество перестановок дающих такой массив можно посчитать по формуле $c_1! \cdot c_2! \cdot \ldots \cdot c_k!$, где c_i — количество раз, сколько встречается i-е наименьшее уникальное число в массиве a. Далее будем считать количество подходящих уникальных массивов-перестановок массива a, и потом домножим ответ на этот коэффициент.

Итого для группы $\alpha = -1$ достаточно только проверить отсортированный по неубыванию массив a на выполнение условия.

Далее перейдем к группе $\alpha = 0$. Аналогично учтём отсортированный по неубыванию массив, если он подходит под условие. Теперь осталось рассмотреть все массивы такие, что есть хотя бы индекс $a_i < a_{i+1}$, и уже в этом индексе сумма модулей будет равна a_i, что точно не меньше чем $\min(a_1, a_2, \ldots, a_n)$. Значит, единственная опция, когда такой массив подойдёт под условие при $\alpha = 0$: если $a_i = \min(a_1, a_2, \ldots, a_n)$, и $a_1 \bmod a_2 = 0, \ldots, a_{i-1} \bmod a_i = 0, a_{i+1} \bmod a_{i+2} = 0, \ldots, a_{n-1} \bmod a_n = 0$.

Фактически, нужно посчитать число способов разбить элементы массива на две группы, чтобы внутри групп каждое число делилось на каждое, а также минимум массива содержался в первой группе.

Пусть массив содержит k различных элементов: $b_1 > b_2 > \ldots > b_k$ в количествах $c_1, c_2, \ldots, c_k$ (c_i копий числа b_i).

Тогда можем решать задачу методом динамического программирования. Будем поочерёдно добавлять $b_1, b_2, ..$ в группы, изначально предполагаем группы пустые. Состояния будут: $dp[i][j]$ — количество способов, если последний элемент первой группы b_i, а последний элемент второй группы b_j. Пересчёт делается просто, надо перебрать предыдущий элемент и проверить условие делимости. Ответом соответственно будет $dp[1][1] + \ldots + dp[1][k]$, так как минимум должен быть в первой группе. Такая динамика наивным образом работает за $O(k^3)$, если написать чуть более аккуратно, то за $O(k^2)$. Что набирает некоторые баллы, но при $n = 3 \cdot 10^5$ слишком медленно. Однако можно заметить интересный факт, что если есть хотя бы одно подходящее разбиение, то количество различных чисел во всём массиве не превосходит $2 \cdot \log_2(10^9)$, так как если в каждой группе все числа делятся на всех, то каждое новое различное число в группе хотя бы вдвое меньше старого, а значит всего их не больше чем $\log_2(10^9)$. Таким образом, если количество различных чисел > 60 мы можем сразу заключить, что ответ равен нулю, и не считать динамику, а при $k \leqslant 60$ даже решение с асимптотикой $O(k^3)$ будет более чем быстрым. Это решение для $\alpha = 0$.

Для $\alpha = 1$ добавляется три новых случая, при которых сумма может быть равна $\min(a_1, a_2, \ldots, a_n) + 1$.

- В двух группах также все делятся на всех, заключительный элемент первой группы равен $\min(a_1, a_2, \ldots, a_n) + 1$
- В двух группах также все делятся на всех, кроме одного места в одной из групп где $a_i \bmod a_{i+1} = 1$ в точности, заключительный элемент первой группы равен $\min(a_1, a_2, \ldots, a_n) + 1$
- Если $\min(a_1, a_2, \ldots, a_n) = 1$, то возможно разбиение на три группы, заключительный элемент первых двух равен 1, и все на всех делятся.

Несложно понять, что других случаев когда сумма равна $\min(a_1, a_2, \ldots, a_n) + 1$ нет. Каждый из трёх случаев нужно посчитать отдельно. Это делается при помощи аналогичного динамического программирования. Для первого случая нужно просто вычислить ответ по-другому, а динамика полностью аналогична случаю $\alpha = 0$. Для второго случая нужно добавить в состояния дп флаг — был ли использован $a_i \bmod a_{i+1} = 1$. И для третьего случая нужно добавить в дп состояние для последнего индекса третьей группы. Наивная реализация будет работать за $O(k^4)$, что достаточно быстро, ведь если $k > 3 \cdot \log_2(10^9)$ мы все ещё можем не считать динамику, и заключить, что ответ равен 0.