Задача A. В погоне за Пингвином

Автор задачи: Мария Жогова, разработчики: Владимир Рябчун и Даниил Орешников

Для решения первых двух подзадач можно было написать полный перебор возможных действий. Во второй подзадаче следовало помечать точки как достижимые, и при попадании в какую-то точку по второму разу не перебирать дальнейшие перемещения из нее.

Для решения остальных подзадач заметим следующий факт. Пусть всего было сделано x шагов вперед и y шагов вправо, тогда суммарно за такие перемещения было потрачено $ax + by$ топлива. Таким образом, все достижимые точки можно описать уравнением $ax + by \leqslant s$. Задача сводится к тому, чтобы найти количество целочисленных решений такого уравнения.

В четвертой подзадаче проходит следующее решение: переберем количество сделанных шагов вправо f, тогда на движение прямо останется $S - af$ топлива, за которое можно попасть в точки от $(f, 0)$ до $(\left\lfloor \frac{s-af}{b} \right\rfloor, 0)$. Посчитать сумму таких величин по всем f можно за время $\mathcal{O}\left(\left\lfloor \frac{s}{a} \right\rfloor\right)$.

В третьей подзадаче можно было заметить, что при целых $\frac{s}{b}$ и $\frac{s}{a}$ количество точек с целыми координатами в треугольнике между точками $(0, 0)$, $(0, \left\lfloor \frac{s}{b} \right\rfloor)$ и $(\left\lfloor \frac{s}{a} \right\rfloor, 0)$ равно половине точек с целыми координатами в прямоугольнике с вершиной в точке $(\frac{s}{a}, \frac{s}{b})$, не считая точек на диагонали $ax + by = S$.

Точки в таком прямоугольнике посчитать просто: их ровно $\left(\frac{s}{a} + 1\right) \cdot \left(\frac{s}{b} + 1\right)$. А точки на диагонали следут между крайними с шагом $\left(+\frac{a}{\gcd(a,b)}, -\frac{b}{\gcd(a,b)}\right)$, то есть их количество равно $\frac{s \cdot \gcd(a,b)}{ab} + 1$. Осталось сложить и поделить пополам, время работы решения — $\mathcal{O}(1)$.

Для решения оставшихся подгрупп внимательнее рассмотрим формулу для ответа, которую получили ранее:

$$\left\lfloor \frac{s}{b} \right\rfloor + \left\lfloor \frac{s-a}{b} \right\rfloor + \ldots + \left\lfloor \frac{s - \left\lfloor \frac{s}{a} \right\rfloor a}{b} \right\rfloor.$$

Представим каждую дробь $\left\lfloor \frac{s-ta}{b} \right\rfloor$ как $\frac{s-ta}{b} - \frac{(s-ta) \bmod b}{b}$. Слагаемые первого вида образуют арифметическую прогрессию (за d_a обозначим $\left\lfloor \frac{s}{a} \right\rfloor$):

$$\frac{s}{b} + \frac{s-a}{b} + \ldots + \frac{s-d_a a}{b} = \frac{1}{b} \cdot \left(s(d_a + 1) - a\frac{d_a(d_a+1)}{2} \right).$$

А слагаемые второго вида образуют последовательность остатков по модулю b, начинающуюся в $s \bmod b$ и идущую с шагом $-a$. Можно заметить, что каждые $\frac{b}{\gcd(a,b)}$ идущих подряд элементов этой последовательности образуют период из остатков вида $(a \bmod b) + k\gcd(a, b)$ для целых k (просто записанных в другом порядке). Сумму такого периода тоже можно посчитать по формуле геометрической прогрессии, а количество периодов, которые целиком войдут в ответ, равно $\left\lfloor \frac{d_a+1}{\frac{b}{\gcd(a,b)}} \right\rfloor$.

После останется только хвост из остатков, которых меньше, чем длина периода. Но в таком случае их сумму можно просто посчитать циклом за $\mathcal{O}(b)$.

Для полного решения достаточно выбрать оптимальное из двух решений: за $\mathcal{O}\left(\frac{s}{b}\right)$ и за $\mathcal{O}(b)$ для каждого набора входных данных в зависимости от того, какая из двух величин меньше.

Задача B. Большой потоп

Автор задачи: Константин Бац, разработчик: Мария Жогова

В задаче требовалось выбрать порядок подрыва башен такой, что сумма объемов воды, находящихся в башнях в момент взрыва, была максимальна.

Заметим, что моменты для разрушения башен из системы i желательно выбирать до t_i. Иначе, вода находящаяся в не взорванных башнях будет спущена, и это не увеличит искомую сумму.

Для прохождения тестов первой подзадачи можно было явно перебрать моменты для подрыва башен. Поскольку $k = 1$ и $t_i \leqslant 5$, в каждый момент можно было взорвать не более одной башни, поэтому для каждой секунды от 1 до 5 нужно выбрать, какую башню стоит взорвать в эту секунду. Перебором среди всех таких комбинаций можно найти такую, которая даёт максимальный результат.

Во второй подзадаче гарантировалось, что $b_i = 1$ и всё t_i разные. На самом деле, это означало, что единственную башню каждой подсистемы нужно взорвать за секунду до спуска воды, то есть на $t_i - 1$ секунде. Очевидно, что таком случае ответ будет максимальным.

Третья подзадача выделялась тем, что водосброс происходил во всех системах одновременно, пусть это будет секунда T. Тогда очевидно, что всего мы можем взорвать не больше $k \cdot T$ башен. Предположим, суммарно в системах больше $k \cdot T$ башен. Тогда выгоднее взрывать башни из систем с большим начальным уровнем воды. Действительно, давайте заменим одну башню из системы i на башню из системы j и $a_i > a_j$, тогда вне зависимости от момента взрыва из башни выльется строго меньше воды. Предположим у нас меньше $k \cdot T$ башен тогда выгодно начать взрывать башни как можно позже. При этом, если все башни в итоге будут взорваны, порядок разрушения не важен.

В четвёртой подзадаче в каждой системе была всего одна башня. Давайте объединим все ранее приведённые идеи. Рассмотрим системы с самым поздним сбросом воды, то есть максимальным $t_i = T$. Выберем из них k с максимальным a_i, очевидно, их выгодно разрушить в секунду $t_i - 1$. Перейдем к секунде $T - 1$. Добавим в рассмотрение системы с $t_i = T - 1$. Среди оставшихся с предыдущего шага и добавленных в рассмотрение снова выберем k систем с максимальными a_i. Будем продолжать, пока не дойдем секунды 0 или в рассмотрении не останется систем. Если в рассмотрении закончились системы, опять найдём среди оставшихся $\max t_i$ и повторим всё выше описанное. Хотя $t_i \leqslant 10^9$, будет всего $\mathcal{O}(n)$ секунд, когда мы будем выбирать системы. Для быстрого выбора системы с максимальным a_i можно использовать приоритетную очередь с добавлением, удалением и поиском максимума за $\mathcal{O}(\log n)$. Время работы решения — $\mathcal{O}(n \log n)$.

В подзадаче пять не было ограничения на количество башен в системе, но t_i–е не превышали 10^5. Для решения можно было воспользоваться идей из четвёртой подзадачи. Выберем систему с максимальным начальным уровнем, взорвем в текущую секунду как можно больше башен из этой системы. Понятно, что, если у нас есть несколько систем с одинаковыми a_i, не важно, в какой системе разрушать башни. Поскольку башен может быть очень много, возможно, придётся каждую секунду выбирать системы, в которых нужно будет разрушить башни. Всего нужно будет не больше n раз выбрать систему с максимальным a_i и при этом рассмотреть не больше $\max t_i$ секунд. Значит временная сложность такого решения — $\mathcal{O}((n + \max t_i) \log n)$.

Для прохождения тестов последней подзадачи нужно было немного оптимизировать предыдущее решение. Пусть в течении p секунд с T по $T+p-1$-ю не происходит спусков воды, и в $T+p-1$-ю секунду максимальный начальный уровень воды имела система j. Тогда в течении p секунд можно подрывать башни только из системы j, и это не испортит ответ. Из этого следует, что нам не обязательно рассматривать каждую секунду. Достаточно рассматривать те секунды, когда мы добавляем в рассмотрение новую систему или в какой-то системе заканчиваются башни. Между этими моментами каждую секунду мы будем либо разрушать башни в какой-то определённой системе, либо ничего не разрушать. Таких моментов будет не больше $2n$, значит время работы такого решения будет $\mathcal{O}(n \log n)$.

Задача C. Спасительная загадка

Автор задачи и разработчик: Даниил Орешников

В задаче был дан массив, полученный из некоторого другого массива a вычитанием из него некоторого его циклического сдвига. Требовалось восстановить все возможные величины сдвига, для которых это возможно.

В первой подгруппе можно было перебрать все возможные массивы и величины сдвига. Действительно, если подходящее a существует, не теряя общности можно взять его первый элемент равным 0 и, используя b, восстановить его элементы по порядку. Если какой-то элемент остался не восстановленным, можно принять его также равным нулю, и восстанавливать дальше.

Можно заметить, что такое решение работает за $\mathcal{O}(n^2)$, если просто перебирать элементы по очереди, чтобы найти первый еще не восстановленный. Это же решение проходит и пятую подгруппу. Для первой подгруппы достаточно было написать любой менее оптимальный перебор, например, за время $\mathcal{O}(n^3)$ или дольше.

Дальше можно было заметить, что если выбрать первый элемент массива a равным нулю, и начать восстанавливать его элементы как $a_{i+x} = a_i - b_i$, если известна величина сдвига. Таким образом, можно восстановить элементы a_{i+x}, a_{i+2x}, и так далее. Рано или поздно эта последовательность вернется в исходный элемент a_i по кругу. Необходимым и достаточным условием на то, что все эти элементы можно восстановить — что сумма использованных элементов b равна 0, если мы начали с a_i и вернулись к нему же.

На самом деле, если n и x взаимно просты, то такая последовательность восстановлений обойдет все элементы a и вернется в исходный индекс. Если же нет, то весь массив a разобьется на $\gcd(n, x)$ «циклов», которые можно восстанавливать независимо друг от друга.

Поэтому в случае, когда $n \in \mathbb{P}$, все возможные сдвиги x взаимно просты с n, и достаточно проверить, что сумма элементов b равна нулю, тогда все сдвиги возможны.

А в подгруппе, в которой $n = 2^k$, наибольший общий делитель с произвольным x может быть толкьо равен 2^t для $t \leqslant k$. Чтобы проверить, какие сдвиги x возможны, надо проверить, какие сдвиги 2^t возможны, все сдвиги с таким же gcd с n будут тоже возможны. А сдвиг 2^t возможен, когда для каждого остатка $r < 2^t$ верно, что $b_r + b_{2^t+r} + b_{2\cdot 2^t+r} + \ldots = 0$.

Такое условие можно посчитать для каждого t несложной динамикой. Пусть $\texttt{sum}[t][r]$ — указанная сумма. Тогда $\texttt{sum}[t][r] = \texttt{sum}[t+1][r] + \texttt{sum}[t+1][2^t + r]$. Подсчет такой динамики займет $\sum\limits_{t=0}^{k} 2^k \approx 2n$ времени.

Для полного решения требовалось для каждого возможного делителя n определить, может ли он быть корректным сдвигом. Тогда любой x, gcd которого с n равен этому делителю, будет тоже корректным.

В четвертой подгруппе достаточно малое количество b не равно нулю, поэтому можно было для каждого делителя n определить, какие из них входят в соответствующие «циклы». Для определенного делителя d смотрим, какие остатки дают ненулевые b_i по модулю d, группируем по остаткам, и проверяем, что сумма в каждой группе равна нулю.

Для последних двух подгрупп можно было воспользоваться динамикой, аналогичной подгруппе с $n = 2^k$. Если для каждого d — делителя n, найти некоторое простое число p в его разложении на простые ($d = p \cdot c$), можно пересчитывать динамику $\texttt{sum}[c][r]$ как

$$\texttt{sum}[c][r] = \sum_{i=0}^{p-1} \texttt{sum}[d][c \cdot i + r].$$

Время работы такого пересчета равно $\sum\limits_{d|n} \min\_\text{prime}(d)$, что близко к линейному от n времени работы. Осталось только быстро находить минимальное простое в разложении каждого делителя n. Если делать это за $\sqrt{d}$, можно было решить предпоследнюю группу, а если воспользоваться линейным решетом Эратосфена, получалось полное решение.

Задача D. Необычная ловушка

Автор задачи и разработчик: Константин Бац.

Данная в задаче система комнат представляет из себя дерево, то есть связный ациклический неориентированный граф.

Из того, что ловушка позволяет высаживать людей из лифта, следует, что человек по пути из комнаты x в комнату y может посетить комнаты на пути из x в $\text{lca}(x, y)$ и пути из $\text{lca}(x, y)$ в y. Здесь и далее lca — ближайший к вершинам x и y общий предок.

Для решения первой и второй подзадач можно было явно просимулировать перемещение людей по вершинам и рёбрам графа. Давайте для всех вершин посчитаем минимальное количество лифтов, чтобы перевезти заложников в каждую из сторон. Ответ на задачу — $\sum\limits_{u \to v} w_{u \to v} \cdot \text{cnt}_{u \to v}$, где cnt — минимальное количество лифтов, необходимое, чтобы перевезти всех людей по ребру $u \to v$.

Ограничения третьей подзадачи указывали на то, что вместимость лифта позволяет перевезти всех людей за одну поездку. Таким образом, для каждого рёбра в одну и другую сторону нужно было понять, поедет ли кто-то по нему, и если это так, добавить в ответ $w_{u \to v}$ этого ребра.

В подзадаче четыре каждая комната соединялась с одной или двумя, поэтому граф на самом деле представляется в виде цепочки вершин. Это отличает группу от других тем, что весь граф можно представить в виде прямой, а путь из x в y как отрезок этой прямой. Пользуясь деревом отрезков или деревом Фенвика можно было, прибавляя c_i на отрезке, для каждого ребра посчитать сколько раз его пересекут. Дальше для каждого ребра почитать минимальное количество поездок лифта, чтобы перевезти всех людей и с учётом всех w_i посчитать ответ.

На самом деле, решение задачи в общем случае отличается лишь тем, что c_i нужно прибавлять не на отрезке, а в дереве.

Для пятой подзадачи можно было на каждую из m групп найти lca, затем явно выделить путь из x в y и за $\mathcal{O}(n^2)$ к каждому ребру для нужного направления прибавить c_i. Время работы такого решения равно $\mathcal{O}(m \cdot n^2)$.

Решение задачи на полный балл предполагало подсчёт суммы в поддереве. Нам нужно прибавлять c_i на пути от u до p, где p — какой-то предок u. Давайте в каждой вершине хранить два числа: e и f. Рассмотрим группу i, выделим путь из u в v, и пусть $\mathrm{lca}(u, v) = p$. Прибавим к e_v и f_v величину c_i и вычтем из e_p и f_p величину c_i. После рассмотрения всех групп людей, прибавим к e_v и f_v сумму e и f в поддереве соответственно.