Задача A. Важное научное число

Автор задачи и разработчик: Даниил Орешников

Поскольку числа во вводе до 10^9, решение, перебирающее x от 1 до потенциального ответа и проверяющее, подходит ли данный x, не проходит по времени.

Знакомые с Китайской теоремой об остатках могли воспользоваться ею для решения этой задачи, однако это заметное усложнение оригинального решения. На самом деле было достаточно посмотреть на число $a+b+x$ и заметить, что оно обязано делиться как на a, так и на b, так как

- $a+x \vdots b$, $b \vdots b$, значит $a+x+b \vdots b$
- $b+x \vdots a$, $a \vdots a$, значит $b+x+a \vdots a$

Делимость на a и на b равносильна делимости на $\mathtt{lcm}(a,b)$, где `lcm` обозначает *наименьшее общее кратное*, которое можно найти как $\frac{ab}{\mathtt{gcd}(a,b)}$, а `gcd` или *наибольший общий делитель* можно найти с помощью алгоритма Евклида.

Итак, обозначим наименьшее общее кратное чисел a и b за y, тогда задача сводится к поиску наименьшего неотрицательного x, что $x+a+b$ делится на y. Очевидно, что тогда любой подходящий нам x имеет вид $ky-a-b$ для некоторого целого k.

Осталось выбрать среди таких минимальный. Несложно показать, что $a, b \leqslant y$, а значит $x_2 = 2y-a-b \geqslant 0$, соответственно, если ответ меньше, чем x_2, то это может быть только $x_1 = y-a-b$, так как дальше идут только отрицательные числа ($-a-b < 0$). Получаем следующий ответ — если $y \geqslant a+b$, то это $y-a-b$, иначе $2y-a-b$.

Задача B. Погоня за бабочкой

Автор задачи: Владислав Власов, разработчик: Михаил Анопренко

Для решения задачи нужно применить динамическое программирование по поддеревьям.

Запустим поиск в глубину из корня дерева, данного нам в условии. Для каждой вершины посчитаем следующие величины: d_i — расстояние от вершины i до корня и h_i — минимальное расстояние от вершины i до какого-то из листов, лежащего в поддереве этой вершины. Каждую из этих величин легко посчитать в процессе обхода в глубину: $d_v = d_u + 1$, где u — родитель вершины v; $h_u = 1 + \min\limits_{v \in V} h_v$, где V — множество сыновей вершины u.

Также будем в обходе в глубину считать ответ для каждой вершины. Пусть ans_i — это ответ для поддерева вершины i, то есть минимальное количество друзей, которое нужно поставить в некоторые из листьев этого поддерева, чтобы гарантированно поймать бабочку при условии, что она обязательно полетит в это поддерево. Будем вычислять ans_i для вершины, пользуясь ответами для всех ее детей.

Заметим, что если для вершины u выполнено условие $h_u \leqslant d_u$, то $ans_u = 1$. Действительно, достаточно поставить одного друга в лист с минимальной глубиной, и до вершины u он успеет добраться не позже, чем бабочка из корня. В противном случае, $ans_u = \sum\limits_{v \in V} ans_v$, где V — множество сыновей вершины u, так как мы точно не успеем поймать бабочку в вершине u, а значит, нам нужно быть готовыми ловить ее в любом из поддеревьев сыновей вершины u.

Ответ на задачу — это значение ans_1. Решение требует одного обхода дерева в глубину, асимптотика времени его работы составляет $O(n)$.

Задача C. В поход!

Автор задачи: Алексей Шик, разработчик: Дмитрий Гнатюк

Эту задачу следовало решать бинпоиском по ответу. Для этого достаточно заметить, что если все Смешарики могут собраться за время t, то они могут собраться и за любое время, большее t, просто сделав несколько одинаковых шагов после, а поэтому бинпоиск можно применять.

Давайте теперь проверим, смогут ли Смешарики собраться за время t. Каждый Смешарик может оказаться в любой точке, для которой *манхэтеннское расстояние* меньше или равно t (манхэттенским расстоянием называется сумма расстояний по двум координатам).

Несложно заметить, что множество точек, удаленных от данной по манхэттенскому расстоянию не более, чем на заданную константу — это правильный ромб. Таким образом, необходимо пересечь ромбы, задающие достижимые для Смешариков за время t точки, и проверить, содержит ли оно хотя бы одну точку с целочисленными координатами. Воспользуемся матрицей поворота, чтобы повернуть систему и, для простоты, искать пересечения квадратов (https://ru.wikipedia.org/wiki/Матрица_поворота).

Время работы такого решения — $\mathcal{O}(n \log C)$, где C — максимальная начальная координата Смешарика.

Задача D. Починка цепочки

Автор задачи: Даниил Орешников, разработчик: Николай Будин

Для начала, переформулируем задачу в терминах теории графов. Дан граф из n вершин и m ребер. Изначально все вершины покрашены в белый цвет. За один ход можно:

- Взять белую вершину, удалить из графа все смежные с ней ребра и покрасить её в чёрный цвет.
- Взять черную вершину, соединить её ребрами с произвольным подмножеством белых вершин, и покрасить её в белый цвет.

Требуется сделать минимальное число операций, чтобы граф стал выглядеть как простой путь $1, 2, \ldots, n$. И все вершины были белыми.

Несложно заметить, что к каждой вершине нужно максимум один раз применять первую операцию. И если к вершине применили первую операцию, то к ней нужно применить и вторую. Следовательно, нужно выбрать минимальное по размеру множество вершин, что если ко всем ним применить первые операции, все оставшиеся в графе ребра будут соединять соседние по номерам вершины. Эта задача может быть решена за время $O(2^n \cdot n)$ с помощью перебора всех вариантов выбора множества вершин, к которым будут применены операции. Это решение может быть соптимизировано до времени $O(2^{\frac{n}{2}} \cdot n)$ с помощью метода meet in the middle.

Задача E. Змейка

Автор задачи и разработчик: Николай Будин

Если n или m четно, то существует Гамильтонов цикл, проходящий по всем клеткам поля. Можно водить змейку по этому циклу, пока игра не будет выиграна. Между двумя соседними съеденными яблоками змейка сделает не более $n \cdot m$ шагов. Следовательно, суммарно будет сделано не более 10 000 шагов. Змейка никогда не врежется сама в себя, потому что она катается по циклу длины $n \cdot m$.

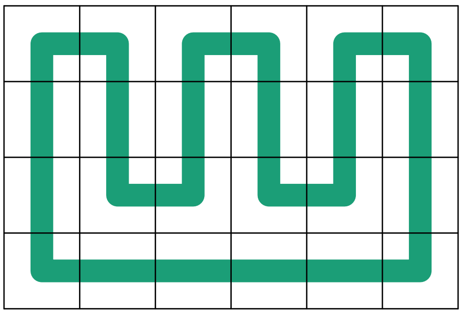

Рис. 1: Пример цикла при четном количестве столбцов.

Если и n, и m нечетны, то можно вырезать из поля одну угловую клетку и тогда на нем тоже можно будет построить Гамильтонов цикл (см. иллюстрацию). Как и в предыдущем случае, будем водить змейку по этому циклу. Единственное отличие — если яблоко находится в удаленной угловой клетке. В таком случае, нужно заехать в нее, когда голова змейки будет проезжать мимо.

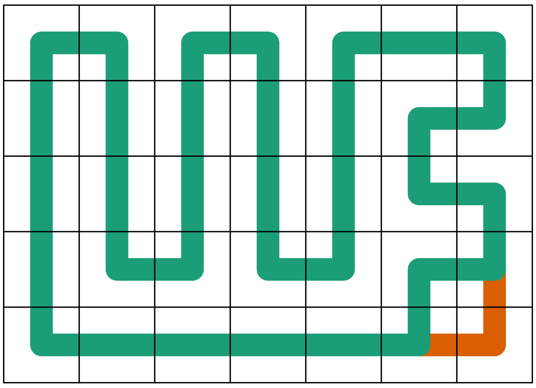

Рис. 2: Пример цикла при нечетных n и m.

Задача F. Конфликт интересов

Автор задачи: Аслан Тамаев, разработчик: Михаил Иванов

Сначала найдём число способов выбрать высоты горизонтальных сторон одного прямоугольника. Спроецируем для этого прямоугольник на вертикальную ось, он на ней высечет промаркированный отрезок длины H. Нам надо выбрать на нём подотрезок длины не более h. Если длина равна единице, то есть H положений, где его разместить, если двойке, то $H-1$, и так далее — если h, то есть $H-h+1$ положений. По формуле для суммы арифметической прогрессии всего $\frac{(H+(H-h+1))h}{2} = \frac{(2H-h+1)h}{2}$ положений. Аналогично, вертикали для вертикальных сторон можно выбрать $\frac{(2W-w+1)w}{2}$ способами.

Итого есть $\frac{(2H-h+1)h}{2} \cdot \frac{(2W-w+1)w}{2}$ способов выбрать один прямоугольник. Если мы временно забудем про условие, что прямоугольники не пересекаются, то второй прямоугольник можно выбрать столькими же способами, и всего получается $\left(\frac{(2H-h+1)h}{2} \cdot \frac{(2W-w+1)w}{2}\right)^2$ способов выбрать пару прямоугольников.

Теперь надо вычесть количество пересекающихся пар прямоугольников. Заметим, что прямоугольники пересекаются тогда и только тогда, когда пересекаются их проекции на вертикальную ось и пересекаются их проекции на горизонтальную ось. Пусть $P_{H,h}$ — число способов выбрать на промаркированном отрезке длины H два пересекающихся отрезка с целыми концами длины не более h. Тогда количество пересекающихся пар прямоугольников равно $\left(\frac{(2H-h+1)h}{2} \cdot \frac{(2W-w+1)w}{2}\right)^2 - P_{H,h} \cdot P_{W,w}$, и, таким образом, задача решится, как только мы найдём $P_{H,h}$.

Давайте предположим, что на отрезке длины H надо разместить пересекающиеся отрезки длин p и q. Сколько есть способов сделать это? Если $p + q \geqslant H + 1$, то отрезки пересекутся при любом раскладе, то есть способов будет ровно $(H + 1 - p)(H + 1 - q)$. Нам будет удобно переписать это число в более симметрическом виде: $(H + 1)((H + 1) - (p + q)) + pq$.

Что же, если $p+q \leqslant H$? И в этом случае формула есть, правда, немного другая: а именно, из всех $(H+1)((H+1)-(p+q))+pq$ способов надо вычесть те $((H+2)-(p+q))((H+1)-(p+q))$, когда отрезки всё-таки не пересекаются (зафиксируем положение первого отрезка и для него посмотрим, сколько не пересекающихся с ним положений второго отрезка: увидим, что это сначала арифметическая прогрессия, убывающая от $(H+1)-(p+q)$ к нулю, потом постоянный ноль, а потом арифметическая прогрессия, возрастающая назад к $(H + 1) - (p + q)$; удвоенная сумма арифметической прогрессии как раз даст нужный результат). Получится $((p + q) - 1)((H + 1) - (p + q)) + pq$.

Итак, надо перебрать все p от 1 до h, все q от 1 до h, для каждого из них взять $(H + 1)((H + 1) - (p + q)) + pq$, если $p + q \geqslant H + 1$, иначе взять $((p + q) - 1)((H + 1) - (p + q)) + pq$ и всё это сложить. Для начала заметим, что pq можно сложить отдельно, получится $\left(\frac{h(h+1)}{2}\right)^2$, и потом мы это добавим к первому слагаемому. А первое слагаемое зависит только от $p + q$, которое обозначим за s.

Так что надо перебрать все s от 2 до $2h$, для каждого из них взять $(H + 1)((H + 1) - s)$, если $s \geqslant H + 1$, и $(s - 1)((H + 1) - s)$, если $s \leqslant H$, и всё это сложить. А по сколько раз брать слагаемое?

Столько раз, сколько способов есть представить s в виде суммы двух слагаемых p и q, каждое из которых от 2 до h. Это число можно выразить как $\min\{s-1, 2h+1-s\}$.

Заметим, что $\min\{H+1, s-1\}((H+1)-s)$ как раз равно $(H+1)((H+1)-s)$, если $s \geqslant H+1$, и $(s-1)((H+1)-s)$, если $s \leqslant H$ (если $s = H+1$, то $(H+1)((H+1)-s) = (s-1)((H+1)-s)$, поскольку вторая скобка равна нулю). Поэтому итоговая формула такова:

$$P_{H,h} = \left(\frac{h(h+1)}{2}\right)^2 + \sum_{s=2}^{2h} \min\{s-1, 2h+1-s\} \min\{H+1, s-1\}(H+1-s).$$

Если по этой формуле найти $P_{H,h}$ и $P_{W,w}$ и подставить в формулу выше, то ответ будет найден за $\mathcal{O}(\max\{h, w\})$, и эта асимптотика достаточно хороша, поскольку по условию $h, w \leqslant 3 \cdot 10^5$. Заинтересованный читатель спросит: а можно ли решить задачу с временной сложностью $\mathcal{O}(1)$? Что ж, да, можно. Указанная выше сумма может быть ещё лучше свёрнута, хоть этого подвига и не требовалось совершить на олимпиаде. Если $2h \leqslant H$, то

$$P_{H,h} = h^2 \left(Hh - \frac{11h^2 - 6h - 5}{12}\right),$$

а если $2h > H$, то

$$P_{H,h} = \frac{5h^4 + (H-1)H(H+1)(H+2) - 10(2H+1)h^3 - 4\left(H^2 + H - 1\right)(2H+1)h + \big(24H(H+1)+1\big)h^2}{12}.$$

В качестве упражнения можете доказать, что при любых целых H и h эти выражения принимают целые значения, а в качестве значительно более сложного упражнения — что не просто целые, а те, что нужно.

Задача G. Урок математики

Автор задачи и разработчик: Даниил Орешников

Решение этой задачи состоит в том, чтобы внимательно посмотреть на формулу для среднего геометрического и выразить исходные числа через x, y и z, а не пытаться подобрать их так, чтобы совпали ли средние геометрические. Особенно, учитывая, что числа вещественные, и перебор займет очень большое время.

Если посмотреть на $x = \sqrt{bc}$, $y = \sqrt{ac}$, $z = \sqrt{ab}$, то можно заметить, что если $a, b, c > 0$, то

$$\frac{yz}{x} = \frac{\sqrt{a^2bc}}{bc} = a$$

Похожим образом выражаются b и c, достаточно взять произведение двух средних геометрических и разделить его на третье. Получаем, что в ответ надо вывести числа $\frac{yz}{x}$, $\frac{xz}{y}$, $\frac{xy}{z}$.

Также стоит отметить, что в `C++` стоило использовать `printf` или `std::cout.precision` с `std::fixed`, чтобы выводить вещественные числа с достаточной точностью.

Задача H. Школьные переписки

Автор задачи: Григорий Шовкопляс, разработчик: Даниил Орешников

Для начала опишем решение задачи без каких-либо дополнительных оптимизаций, которое работает за $\mathcal{O}(nq)$ в худшем случае. Чтобы получить такое решение, требовалось дословно реализовать все, что описано в условии. Для этого будем хранить для каждого пользователя его множество непрочитанных сообщений $\texttt{unread}_t$.

Когда пользователь a_i отправляет сообщение i пользователю b_i:

1. добавляем i в $\texttt{unread}_{b_i}$
2. если при этом a_i — учитель, а b_i — ученик, добавляем i в $\texttt{unread}_{\texttt{director}}$
3. если, наоборот, a_i — ученик, а b_i — учитель, аналогично добавляем сообщение в непрочитанные директора, а затем **проходим циклом** по всем учителям и добавляем каждому это сообщение

Когда пользователь a_i читает сообщение x_i:

1. удаляем это сообщение из $\texttt{unread}_{a_i}$
2. если a_i — учитель, **проходим циклом** по всем учителям, и удаляем x_i из непрочитанных каждого из них

При таком подходе для ответа на вопрос Лосяша достаточно вывести размер соответствующего $\texttt{unread}_{a_i}$. Теперь обратим внимание на выделенные части решения. Заменим проходы цикла на операцию, занимающую $\mathcal{O}(1)$ времени на исполнение.

Для этого заметим, что каждое сообщение, полученное учителями от ученика, ровно один раз в один момент добавляется в «непрочитанные» каждого учителя, и так же одновременно навсегда удаляется из всех их «непрочитанных». Заведем отдельный `unreadT`, хранящий сообщения, полученные учителями от учеников и внесем следующие изменения в базовое решение:

- при получении учителем сообщения от ученика, положим его в `unreadT`
- если учитель a_i хочет прочитать сообщение, оно лежит либо в $\texttt{unread}_{a_i}$, либо в `unreadT`, надо проверить его наличие в каждом из этих множеств и удалить оттуда, где оно нашлось
- на запрос Лосяша в случае учителя ответ будет суммой размеров соответствующего $\texttt{unread}_{a_i}$ и `unreadT`

Таким образом, можно каждый запрос обрабатывать за $\mathcal{O}(1)$, и этого достаточно для решения задачи.

Задача I. Индикатор

Автор задачи и разработчик: Михаил Иванов

Если $a \leqslant b$, то надо просто $b - a$ раз нажать на кнопку и получить нужное число.

Если $a > b$, то ситуация значительно усложняется: надо сначала $m - a$ раз нажать на кнопку и получить число m, затем нажать один раз и получить число 1, а потом нажать $b - 1$ раз и получить b. Итого будет произведено $(m - a) + 1 + (b - 1) = m + b - a$ нажатий.

Задача J. Марафонец

Автор задачи: Николай Будин, разработчик: Ильдар Загретдинов

Задача решается парсингом входных данных и суммированием получившихся чисел.

Как вариант, можно при считывании времени переводить его в секунды и просуммировать, и в самом конце перевести в указанный формат. Также можно завести структуру время, содержащая полями часы, минуты и секунды и складывать сразу в таком формате.