Задача A. Пора домой

Автор задачи: Демид Кучеренко, разработчик: Павел Ральников

Для начала найдём максимальный префикс массива a с суммой не больше W. Пусть его длина равна l. Тогда понятно, что если $l \leqslant p$, то $(n - l)$ — ответ на задачу (если прицепить к составу длиной l хотя бы ещё один вагон, то мост рухнет). Иначе найдём максимальный префикс массива a такой, что сумма в любом его окне размера p не больше W. Чтобы это сделать быстро, пойдём по массиву слева направо указателем i, поддерживая сумму в окне $[i - p + 1, i]$. Пусть первый раз, когда такая сумма стала больше W, произошёл, когда указатель был в позиции k. Тогда очевидно, что максимальная длина состава, который сможет проехать по мосту — это $k - 1$, а значит ответ на задачу — это $n - k + 1$

Задача B. K-mex

Автор и разработчик задачи: Евгений Карпович

Давайте рассмотрим простое решение и постараемся его улучшить.

В простом решении мы можем просто добавлять и удалять элементы из множества, а при ответе на запрос перебирать числа $0, k, 2k, 3k, \ldots$ и так, пока не найдём ответ. Это решение будет долго работать, если ответ будет равен $c \cdot k$, где c большое.

Будем улучшать решение. Сначала рассмотрим решение задачи без операции удаления. Если к нам первый раз придёт запрос для заданного k, то посчитаем для него ответ жадно и запомним его. В дальнейшем будем уже не с 0 проверять, есть ли число во множестве, а с предыдущего ответа.

Теперь рассмотрим вариант задачи с операцией удаления. Давайте для фиксированного k будем в `set` хранить все числа, которые мы удалили и они $\leqslant$ чем максимальный ответ, найденный для этого k. Тогда посмотрим, что происходит при операции поиска ответа. Если `set` для заданного k будет не пустой, то ответом будет минимальный элемент из множества, иначе будем пробовать улучшить текущий максимальный ответ для этого k (то есть, если он был равен $c \cdot k$, то будем проверять $c \cdot k, (c + 1) \cdot k, \ldots$).

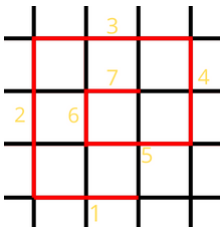
Осталось разобраться, как нам в случае операции добавления/удаления пересчитывать эти `set`. Давайте на самом деле просто для каждого значения запомним, в каких `set` оно участвует, и будем обновлять все такие.

Посчитаем время работы. Давайте поймём, в скольких множествах может участвовать заданное значение x. Во-первых, оно лежит во множествах, где x делится на k. Во-вторых, если x лежит во множестве для числа k, то уже добавлено хотя бы $\frac{x}{k}$ чисел. То есть, если x лежит в t множествах и среди этих k будут наибольшие делители x, то у нас уже должно быть добавлено примерно $\sum_{k_i=1}^{t} \frac{x}{k_i}$, где k_i — i-й наибольший делитель числа x. Так как суммарно у нас q запросов, то заданное значение x может лежать не в большом количестве множеств.

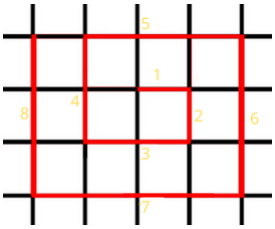
Задача C. Горнолыжный курорт

Автор задачи: Демид Кучеренко, разработчик: Александр Шефер

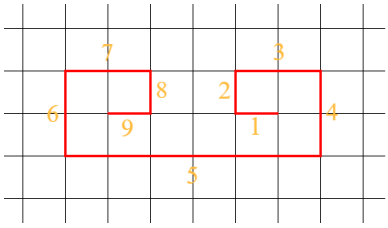
Заметим, что из ограждений можно составить 3 типа спиралей:



Первый тип задаётся условием $a[i] > a[i + 2]$ для $i \in \{2, 3, .., n - 2\}$ (при этом значение $a[1]$ не играет роли).



Второй тип задаётся условием $a[i] < a[i+2]$ для $i \in \{1, 2, 3, .., n-3\}$ (при этом значение $a[n]$ не играет роли).



Третий же тип подразумевает, что мы можем выделить такой индекс $i \in \{3, 4, .., n-2\}$, что из ограждений с номерами $1, 2, .., i$ можно составить спираль второго типа, из ограждений $i, i+1, .., n$ — спираль первого типа, и при этом $a[i] > a[i-2] + a[i+2]$ (в примере выше $i = 5$).

Итак, для ответа на задачу мы должны проверить наш массив на соответствие хотя бы одному из типов выше.

Задача D. Засада на переговорах

Автор задачи: фольклор, разработчик задачи: Владимир Аверенков

Построим общий пример конструктивно. Строки с номерами от 1 до N будем заполнять таким образом: у строки с номером i заполним первые $(2N - i + 1)$ клеток 2, а остальные 0. У строк с номерами от $(N + 1)$ до $2N$ первые $(2N - i)$ клеток заполним 2, клетку с номером $(2N - i + 1)$ 1, а все остальные 0.

В первых N строках будут суммы $4N, 4N - 2, \ldots, 4N - 2N + 2 = 2N + 2$, во вторых N строках будут суммы $2N - 1, 2N - 3, \ldots, 1$.

В первых N столбцах будут суммы $4N - 1, 4N - 3, \ldots, 4N - 2(N - 1) - 1 = 2N + 1$, во вторых N столбцах будут суммы $2N, 2N - 2, \ldots, 2$.

Пример для $N = 3$:

2	2	2	2	2	2
2	2	2	2	2	0
2	2	2	2	0	0
2	2	1	0	0	0
2	1	0	0	0	0
1	0	0	0	0	0

Задача E. Копейский рок

Автор задачи: Савелий Беляков, разработчик: Савелий Григорьев

Заметим следующее: если какое-то число в таблице встречается более m раз, то невозможно расставить числа так, чтобы в каждом столбце все числа были различные.

Докажем, что если все числа встречаются не более m раз, то искомая расстановка возможна.

Доказывать будем индукцией по количеству столбцов:

База: $m = 1 \Rightarrow$ очевидно.

Переход: $m \Rightarrow m+1$. Сейчас рассматриваем доску $n \times (m+1)$. Построим двудольный граф, где в левой доле будет n вершин — строки таблицы, а в правой доле будут числа. Ребро $r-k$ обозначает, что число k встречается в строчке r. По лемме Холла существует паросочетание, насыщающее левую долю: рассмотрим произвольные t вершин из левой доли. Тогда в этих t строках суммарно $t \cdot (m+1)$ клеток, а поскольку каждое число встречается не более $m+1$ раз, то различных чисел в этих строках хотя бы $\geqslant \frac{t(m+1)}{m+1} = t$, следовательно, любые t вершин левой доли связаны с хотя бы t вершинами правой доли, поэтому в графе есть паросочетание, насыщающее левую долю.

Разобьём все числа на два множества: пусть $A_1, A_2, \dots A_p$ — числа, которые встречаются в таблице ровно $m+1$ раз, а $B_1, B_2, \dots B_q$ — числа, которые встречаются меньше $m+1$ раза. Докажем, что всегда можно построить паросочетание, насыщающее левую долю, что в него входят все числа из A:

Найдём какое-нибудь паросочетание, насыщающее левую долю. Пусть в него не вошло число A_i. Построим ориентированный граф на рёбрах исходного графа, где рёбра из паросочетания направлены слева направо, а рёбра не из паросочетания — справа налево.

Посмотрим на все достижимые вершины правой доли из A_i в этом графе: если среди достижимых встречается число из B, то прочередуем рёбра на пути из A_i в эту вершину, тем самым, количество чисел из A, лежащих в паросочетании увеличится.

Если не встретилось ни одного числа из B, значит, все посещённые вершины в правой доле — из A. Заметим, что если мы посетили k вершин левой доли, то в правой доле мы посетили $k+1$ вершину. Тогда у нас получилось, что $k+1$ число правой доли умещается в k строк, но это невозможно, так как в k строках $k \cdot (m+1)$ клеток, а нам нужно $(k+1) \cdot (m+1)$ клеток. Противоречие.

Следовательно, мы всегда можем найти путь из A_i до числа из B, прочередовать его и увеличить количество чисел из A в паросочетании. Следовательно, можно добиться того, что все числа из A будут в паросочетании.

Выберем такое паросочетание, расставим в соответствии с ним числа в текущем столбце, и перейдём от $m+1$ к m. Заметим, что все числа, которые встречаются в таблице $m+1$ раз мы расставили в текущем столбце, следовательно, в оставшейся части матрицы все числа встречаются не более m раз, значит, можно воспользоваться предположением индукции.

Из доказательства следует алгоритм построения ответа: идём по столбцам, строим граф, в нём находим максимальное паросочетание, в которое входят все числа из A, и переходим к следующему столбцу. Чтобы в паросочетание гарантированно были включены все числа из множества A пройдёмся алгоритмом Куна сначала по вершинам из A, а затем по всем остальным. Поскольку алгоритм Куна жадный, то он добавит в паросочетание все числа из A.

Итоговая ассимптотика — $\mathcal{O}(m \cdot \texttt{алгоритм Куна}) = \mathcal{O}(m \cdot n \cdot mn) = \mathcal{O}(n^4)$.

Задача F. Лекция по товароведению

Автор задачи: Демид Кучеренко, разработчик: Алексей Калмыков

Разберём два случая: когда $x = y$ и когда $x \neq y$.

В первом случае важно только количество коробок в башне, различных высот не больше, чем суммарное количество коробок, и не больше чем $2 \cdot \min(a, b)+1$, возьмём из этого минимум и выведем.

Во втором случае у нас три разных вида башен: в которых количество красных коробок на 1 больше количества синих коробок, в которых красных и синих коробок поровну, и в которых синих коробок на 1 больше. Для первого случая таких башен $\min(a, b+1)$, для второго — $\min(a, b)$, для третьего — $\min(a+1, b)$. Сумма этих трёх чисел и будет ответом.

Задача G. Минимальное максимальное

Автор и разработчик задачи: Евгений Карпович

Заметим, что $f(l, r) \leqslant f(l, r+1)$. Чтобы доказать этот факт, давайте посмотрим, как поменяется сумма и `xor` при добавлении элемента x. Сумма увеличится на x, а xor не может увеличиться больше, чем на x.

Дальше можно было использовать два указателя или бинарный поиск для решения задачи. Если вы решаете задачу вторым способом, то вы перебираете правую границу ответа и для неё

ищете оптимальную левую границу бинарным поиском. Вам потребуется за $O(1)$ находить сумму на отрезке и `xor` на отрезке. Для этого можно использовать префиксные суммы и префиксный `xor`.

Задача H. Ящики

Автор задачи: Максим Сапрыкин, разработчик: Александр Шефер

Вместо того, чтобы пытаться все ящики перетащить в первую стопку, будем действовать наоборот. Пусть все $\sum d[i]$ ящиков стоят в первой стопке, давайте добьёмся того, чтобы для любого i в i-й стопке стояло $d[i]$ ящиков.

Заведём массив cur_d, отвечающий за текущее расположение ящиков. Вначале $cur\_d = (\sum d[i], 0, 0..., 0)$. Мы хотим из cur_d с помощью обратных операций получить d. Под обратной операцией подразумевается следующее:

i-й мальчик берет $\lfloor \frac{n}{i} \rfloor - 1$ ящиков из i-й стопки и по одному расставляет в ящики с номерами $2i, 3i..., \lfloor \frac{n}{i} \rfloor i$.

Посмотрим, как с помощью обратных операций добиться равенства массивов. Посмотрим на первую стопку. Сначала в ней стоит $\sum d[i]$ ящиков. Изменять количество ящиков в ней может только первый мальчик. Чтобы добиться равенства $cur\_d[1] = d[1]$ первый мальчик должен совершить ровно $\frac{\sum_{i>1} d[i]}{n-1}$ обратных операций (при этом $\sum_{i>1} d[i]$ должно делиться на $n-1$). Давайте проделаем эти операции. Теперь состояние массива $cur\_d = (d[1], \frac{\sum_{i>1} d[i]}{n-1}, .., \frac{\sum_{i>1} d[i]}{n-1})$.

Посмотрим на вторую стопку. Так как первый мальчик больше не может совершать операции, то вторую стопку может изменять только второй мальчик. Так что, если $cur\_d[2] < d[2]$, то можно заранее сказать, что ответ `NO` (так как второй мальчик из второй стопки может только забирать ящики). Если же $cur\_d[2] \geqslant d[2]$, то второй мальчик должен совершить ровно $\frac{cur\_d[2]-d[2]}{\lfloor \frac{n}{2} \rfloor - 1}$ операций (опять-таки $cur\_d[2] - d[2]$ должно делиться на $\lfloor \frac{n}{2} \rfloor - 1$). Совершим эти операции и пойдем дальше.

Давайте будем таким образом проходиться по массиву в порядке возрастания индексов стопок. Заметим, что когда мы приходим в i-ю стопку, только i-й мальчик может её изменить (и при этом он может только уменьшить количество ящиков в ней). Поэтому при проходе по массиву мы будем делать следующее:

1. Будем проверять условия $cur\_d[i] < d[i]$, $(\lfloor \frac{n}{i} \rfloor - 1) \nmid (cur\_d[i] - d[i])$. Если хотя бы одно выполнено, то ответ `NO`. (Если $\lfloor \frac{n}{i} \rfloor - 1 = 0$, то i-й мальчик в данном случае не может производить обратные операции, поэтому будем проверять $cur\_d[i] == d[i]$. Если это условие не выполнено, то добиться равенства нельзя).

2. Если условия не выполнены, то совершаем $\frac{cur\_d[i]-d[i]}{\lfloor \frac{n}{i} \rfloor - 1}$ обратных операций i-го мальчика (добиваясь таким образом нужного количества ящиков в i-й стопке) и переходим к следующей стопке.

3. Если мы обработали все стопки и не получили отрицательный ответ, значит, $cur\_d = d$ и ответ `YES`.

Задача I. Необычный конкурс

Автор и разработчик задачи: Никита Насонков

Массив можно гарантированно узнать за n запросов. Покажем, как это сделать.

Сделаем запрос `? 1 3 4`, ответ на него обозначим за x.

Сделаем запрос `? 2 3 4`, ответ на него обозначим за y.

Тогда $x \oplus y = a_1 \oplus a_3 \oplus a_4 \oplus a_2 \oplus a_3 \oplus a_4 = a_1 \oplus a_2$. Обозначим $a_1 \oplus a_2$ за z.

Зная z, для всех i $(3 \leqslant i \leqslant n)$ можем задавать запрос вида `? 1 2 i` (обозначим ответ на него за p_i), и a_i будет равно $p_i \oplus z$, т.к. $p_i \oplus z = a_1 \oplus a_2 \oplus a_i \oplus a_1 \oplus a_2 = a_i$. Таким образом мы сделали в начале 2 запроса, чтобы узнать x и y, и с помощью $n-2$ запросов узнали все элементы массива, кроме a_1 и a_2, итого n запросов.

Осталось узнать a_1 и a_2 за 0 запросов, но теперь мы знаем a_3 и a_4, а также помним x и y, поэтому это очень просто:

- $a_1 = x \oplus a_3 \oplus a_4$
- $a_2 = y \oplus a_3 \oplus a_4$

Задача J. Работа в Снежинске

Автор и разработчик задачи: Евгений Карпович

Задачи подобного плана, как правило, решаются с помощью структур данных, и эта не исключение. Так как ограничения в задаче достаточно небольшие, то логично подумать в сторону корневых оптимизаций.

Давайте разделим массив на блоки длины k, таких блоков у нас будет примерно $\frac{n}{k}$. Для каждого блока мы хотим поддерживать ответ, то есть минимальное значение $\frac{lcm(a_i,b_i)}{gcd(a_i,b_i)}$.

Давайте посмотрим, что происходит при операции первого типа. Если блок частично пересекается с отрезком запроса, то можно за $O(k)$ пройтись по этому блоку и пересчитать ответ. Таких блоков не больше чем два, поэтому суммарно мы на это потратим $O(k)$ (временем работы gcd пренебрегаем). Если же блок целиком лежит в отрезке запроса (а таких блоков может быть $\frac{n}{k}$), то нужно как-то более быстро пересчитывать ответ.

Для этого давайте для каждого блока предподсчитаем следующую величину: $answer_x$ — какой будет ответ в блоке, если мы всем числам присвоим значение x. Научимся для начала считать ответ для фиксированного x. Для этого давайте переберём все делители d числа x — на самом деле, перебирая этот делитель, мы будем пытаться фиксировать gcd. Тогда заметим, что так как мы хотим минимизировать значение, то нам нужно найти минимальное значение b_i, которое делится на d. И тогда сделаем $answer_x = \min(\frac{b_i}{d})$ по всем таким d. Давайте заметим, что на самом деле $\gcd(b_i, x)$ может быть не равен d, но мы точно знаем, что $\gcd(b_i, x) \geqslant d$, а так как мы хотим минимизировать значение, то хуже мы не делаем.

Уже сейчас мы можем внутри блока посчитать ответ за $A \log A$, где A — максимальное значение. Но можно и ещё лучше сделать!

Давайте заметим, что $answer_x = min(answer_{\frac{x}{p}} \cdot p)$, где p является простым делителем числа x, и ещё не забываем случай, когда $d = x$. Это следует из того, что все делители числа x содержатся среди делителей чисел вида $\frac{x}{p}$.

Давайте посчитаем время работы и найдём оптимальное k. Нам нужно для каждого числа найти все его делители, чтобы внутри блока быстро узнавать минимальное число, которое делится на заданное — это мы делаем за $O(n\sqrt{A})$. Внутри каждого блока у нас предподсчёт теперь работает за $O(A \log \log A)$, то есть суммарно по всем блокам $O(\frac{n}{k} \cdot A \log \log A)$. На запрос мы отвечаем за $O(k \cdot \gcd + \frac{n}{k})$. Отсюда получаем, что выгодно взять k примерно $\sqrt{n}$ (у нас все величины одного порядка, поэтому используем n).

Задача K. Бинарная сортировка

Автор и разработчик задачи: Евгений Карпович

Давайте мысленно представим следующий массив длины $n-1$: $a_i = 0$, если $s_i = s_{i+1}$, и 1 иначе. Заметим, что если мы применяем операцию для индекса i, то все значения массива a не изменяются, кроме a_{i-1}. Давайте рассмотрим это более подробно:

- Для $i \leqslant j$ заметим, что j-й и $(j+1)$-й элементы инвертируют своё значение, поэтому a_j не меняется.
- Для $j < i - 1$ заметим, что j-й и $(j+1)$-й элементы не меняют своё значение, поэтому a_j не меняется.
- Для $j = i - 1$ заметим, что j-й элемент не меняет своё значение, а $(j+1)$-й элемент меняет, поэтому a_j изменит своё значение.

Если мы рассмотрим массив a для отсортированной бинарной строки, то можем увидеть, что в этом массиве не больше одной единицы (у вас либо строка состоит только из нулей или только из единиц, либо выглядит следующим образом — $0000\ldots01\ldots11111$).

Пусть s — это количество единиц в исходном массиве a. Сейчас мы показали, что ответ $\geqslant \max(s-1,0)$. На самом деле, если строка начинается с 0, то ответ равен $\max(s-1,0)$, а иначе он равен s.

Докажем, что если строка начинается с 0, то можно получить ответ $\max(s-1,0)$ (случай с 1 будет похожим). Покажем конструктивное доказательство на маленьком примере $s = 0001110010$:

- Выбираем $i = 3$, тогда $s = 0000001101$,
- Выбираем $i = 7$, тогда $s = 0000000010$,
- Выбираем $i = 9$, тогда $s = 0000000001$.

Задача L. Игра на дереве

Автор задачи: Семён Чебыкин, разработчик: Алексей Калмыков

Давайте подвесим дерево за любую отмеченную вершину. Теперь пойдём по графу с помощью алгоритма `DFS`, начиная с корня, и в каждой отмеченной вершине при входе в неё и выходе из неё будем делать запрос с ней и предыдущей встреченной отмеченной вершиной.

Заметим, что каждое ребро в таком сценарии попадёт в наши запросы ровно два раза. Следовательно, ответ на задачу — это сумма всех ответов на запросы, делённая на 2.

Задача M. Чебаркульские спортсмены

Автор и разработчик задачи: Павел Ральников

Понятно, что задача сводится к тому, чтобы посчитать количество графов на n вершинах с единственной топологической сортировкой.

1. Так как для каждого графа, подходящего под условие, топологическая сортировка единственна, ответ можно посчитать как $count_1 \cdot n!$, где $count_1$ — количество графов, для которых топологическая сортировка — это последовательность вида $1, 2, 3, \ldots, n-1, n$.

2. Во всех таких графах точно есть рёбра вида $i \to i+1$, так как должен существовать путь из вершины i в $i+1$, чтобы сортировка была единственна (иначе можно поменять местами в топологической сортировке i и $i+1$, при этом последовательность останется корректной топологической сортировкой).

3. Научимся считать $count_1$. Для начала заметим, что в нужных нам графах не бывает рёбер вида $i \to j$, где $i \geqslant j$, поскольку перед нами топологическая сортировка ациклического графа.

4. Исходя из всего выше сказанного, становится ясно, что для искомых графов есть 2 требования: отсутствие рёбер «назад» и наличие рёбер вида $i \to i+1$. Тогда $count_1$ — это $2^{\frac{n(n-1)}{2}-(n-1)}$, так как есть $\frac{n(n-1)}{2}$ рёбер, ведущих из вершин с меньшим номером в вершины с большим, из них $n-1$ ребро вида $i \to i+1$. Итого остаётся $\frac{(n-2)(n-1)}{2}$ рёбер, которые мы можем добавить в граф, содержащий только рёбра $i \to i+1$, чтобы он всё ещё подходил под условия, описанные выше в этом пункте. Мы можем добавить в этот граф любое подмножество таких рёбер, а значит $count_1 = 2^{\frac{(n-2)(n-1)}{2}}$, а значит, ответ на задачу — это $2^{\frac{(n-2)(n-1)}{2}} \cdot n!$.

Задача N. Очередные операции над массивом

Автор и разработчик задачи: Евгений Карпович

Давайте сделаем важное наблюдение: $\gcd(n-1, n) = 1$ при любом значении n. Более того, выбор $i = n-1$ и $i = n$ — это самые дешёвые операции. Отсюда мы можем сделать вывод, что ответ $\leqslant 3$.

Пусть g — это gcd всех чисел в массиве. Тогда у нас возможны следующие случаи:

- Если $g = 1$, то операции можно не применять и ответ равен 0,
- Иначе давайте попробуем применить самую дешёвую операцию $i = n$. Если $\gcd(g, n) = 1$, то ответ 1.
- Иначе попробуем применить следующую самую дешёвую операцию, то есть $i = n - 1$. Если $\gcd(g, n - 1) = 1$, то ответ равен 2.
- Иначе ответ равен 3, так как $\gcd(g, n - 1, n) = 1$.

Задача O. Лети, лети, лепесток...

Автор и разработчик задачи: Валентин Голодов

В данной задаче основные сложности состоят в анализе строки и сохранении только нужных значений. Для рейса туда нас интересует код авиакомпании, метка времени прилёта и стоимость, для рейсов обратно код авиакомпании, метка времени вылета и стоимость.

Далее стоит учесть, что авиакомпании, предоставляющие скидку, и авиакомпании, присутствующие в поисковой выдаче, могут не совпадать как множества.

Само решение представляет полный перебор вариантов. Для корректного вывода потребуется умение работать с заполнителями, либо придётся писать много `if`-ов.

Немного упрощает жизнь тот факт, что даты не переваливают даже за месяц, поэтому метку времени достаточно составлять из номера дня в месяце, часов и минут.