

Задача 1. Мешок

У нас есть неизвестный диапазон целых чисел $[x; x + n - 1]$, в котором ровно n различных чисел.

Если тролль доставал карточки только с числами от $\min$ (минимального значения из всех что показал тролль) до $\max$ (максимального значения из всех что показал тролль), значит, все эти числа точно лежат внутри диапазона $[x; x + n - 1]$. Следовательно, диапазон тролля должен, как минимум, вмещать отрезок $[\min; \max]$. Отсюда следует несколько выводов:

1. Нижняя граница диапазона не может быть больше минимального числа: $x \leq \min$.
2. Верхняя граница диапазона не может быть меньше максимального числа: $\max \leq x + n - 1$ или же $\max - n + 1 \leq x$.

Таким образом, получаем: $\max - n + 1 \leq x \leq \min$.

Задача 2. Брачный договор. Один? ДВА!

Сначала нужно посчитать, сколько пар мы можем создать. Их количество равно: $\min(N, M)$. Для каждой пары мы можем заключать брак и расторгать его, что даст нам три документа за $K + L$ дней.

Обозначим за $[x]$ — целую часть числа x . За T дней мы можем заключить и расторгнуть брак для одной пары: $P = \left\lfloor \frac{T}{K + L} \right\rfloor$ раз. После этого остается еще $Rem = T - P \cdot (K + L)$ дней. Если $K \leq Rem$, то мы можем еще раз заключить брак, а для его расторжения уже дней не хватит.

Таким образом, итоговая формула выглядит так: $\min(N, M) \cdot \left(3 \cdot P + \left\lfloor \frac{Rem}{K} \right\rfloor \right)$

Задача 3. Урок физкультуры

Представим данные как граф, где вершины — это дети, каждое ориентированное ребро — требование, что один ребенок должен стоять раньше другого.

Задача свелась к ответу на вопрос, можно ли упорядочить вершины так, что все ориентированные ребра будут идти от вершины с меньшим порядком к вершине с большим порядком.

Для решения первой подгруппы можно было перебрать все возможные $N!$ порядков вершин, для каждого порядка вершин проверить каждое из M ребер. Если мы нашли порядок вершин, при котором все ребра идут от вершины с меньшим порядком к вершине с большим порядком, то ответ YES. Если ни для какого порядка вершин это условие не соблюдается, то ответ NO. В итоге решение работает за $O(T \cdot M \cdot N!)$

Для решения второй подгруппы можно было заметить, что мы можем действовать жадным алгоритмом: на каждом шаге выбирать произвольную вершину, в которую не входит ни одно ребро, и добавлять такую вершину следующей по порядку.

Это можно было реализовать следующим образом. Будем для каждой вершины, еще не выбранной ранее, считать количество ребер, которое входит в эту вершину. Учитываются только те ребра, оба конца которых не были выбраны ранее.

Если на каждом из N шагов была вершина, в которую не входит ни одно из рассматриваемых ребер, то ответ YES.

Если же на каком-то шаге еще остались вершины, и не можем выбрать ни одну из них для добавления в порядок, то подходящего порядка вершин не существует и ответ NO.

Выбор каждой вершины работает за $O(N + M)$, значит, один тестовый набор работает за $O(N \cdot (N + M))$, и все решение работает за $O(T \cdot N \cdot (N + M))$.

Для нахождения полного решения, заметим, что можно оптимизировать предыдущее решение.

Для этого вначале для каждой вершины подсчитаем, сколько ребер в нее входят.

Сохраним отдельно все вершины, в которые не входит ни одно ребро, например, в очереди.

Если на каком-то шаге очередь пуста, подходящего порядка вершин не существует и ответ будет NO.

Иначе берем вершину из очереди, и заметим, что при добавлении ее в порядок перестанут учитываться только те ребра, которые исходят из этой вершины. Значит, достаточно обработать только

такие ребра, и если для рассматриваемого ребра в вершину на конце не входит только одно ребро, то добавляем такую вершину в очередь.

Каждый тестовый случай будет работать за $O(N + M)$, и все решение будет работать за $O(T \cdot (N + M))$.

Такое решение **не пройдет по времени**, так как в условии **нет дополнительного ограничения на сумму N** .

Поэтому заметим, что на ответ никак не влияют вершины, в которые не входит и из которых не выходит ни одно ребро. Для этого достаточно перебирать не все вершины, а только те вершины, которые являются началом или концом какого-либо ребра. Таких вершин не более $2 \cdot M$, поэтому решение будет работать за сумму количества ребер по всем подтестам, которое уже ограничено числом $5 \cdot 10^5$.

Альтернативное полное решение

Можно было заметить, что такой порядок вершин существует тогда и только тогда, когда нет циклов в ориентированном графе.

Проверять наличие цикла в графе можем с помощью поиска в глубину.

Вспомним, что ребра перехода в поиске в глубину образуют дерево. Для каждой вершины будем хранить одно из трех значений: 0, если вершина еще не была посещена, 1, если вершина была посещена, но еще не обошли все ребра из нее, и 2 — если для вершины обошли все ребра и вернулись в ее предка при обходе.

Если в какой-то момент в процессе обхода рассматриваемое ребро ведет в вершину со значением 1, то это ребро идет от вершины к какому-то его дальнему предку, тем самым образовав цикл, и ответ NO.

Если такого ни разу не произошло, то цикла нет, и ответ YES.

Задача 4. ШИМ калибровка

Для решения первой подзадачи можно воспользоваться простым перебором. Вариантов написания перебора при таких ограничениях очень много, а потому их реализацию оставим в качестве упражнения.

Для решения второй подзадачи можно написать перебор по времени, проверив все возможные значения T от 1 до 1000. Первое подходящее (делящее импульсы на нужное число пакетов) и будет решением.

Для решения третьей и четвертой подзадач заметим, что количество пакетов зависит только от «разрывов» (разниц $t_{i+1} - t_i$) между импульсами.

Не сложно догадаться, что перебирать все возможные значения T — бессмысленно, ведь гораздо логичнее и правильнее перебрать только значения разрывов между импульсами. Почему? Если мы введём функцию числа пакетов от времени $p(T)$, то очевидно, что она будет менять своё значение только в тех точках, где она будет пересекать одну из разностей $t_{i+1} - t_i$, а значит, рассматривать остальные точки смысла не имеет. Более того, не сложно понять, что данная функция будет являться монотонно убывающей (чем больше значение T — тем меньше значение функции $p(T)$).

Отсюда следует два решения, которые мы можем написать:

РЕШЕНИЕ 1 (Бинарный поиск)

Так как число пакетов можно представить как монотонно-убывающую функцию $p(T)$, то найти время T можно с помощью бинарного поиска для T , двигаясь влево или вправо по числу пакетов.

Сложность решения $O(n \cdot \log(t_n))$.

РЕШЕНИЕ 2 (Сортировка)

Зная, что число пакетов уменьшается с увеличением T , мы можем вычислить разности между импульсами, записать их в массив и затем отсортировать.

Нам нужно получить k пакетов, то есть нам нужны $k - 1$ самых больших разрывов (разниц $t_{i+1} - t_i$). Остальные должны быть $\leq T$. То есть $T = d[k - 1]$, где d — отсортированный массив разностей между импульсами.

Однако это верно тогда и только тогда, когда $d[k - 2] \geq d[k - 1]$. То есть последний разрыв, который должен быть «разделителем пакета», должен быть строго больше порогового T .

Если $d[k - 2] = d[k - 1]$, то при этом T они оба станут либо $\geq T$, либо $\leq T$. То есть невозможно получить ровно k пакетов.

Итоговая сложность решения: $O(n \cdot \log(n))$.

Подзадача четыре отличается от подзадачи три большими требованиями к качеству реализации решения.

Задача 5. Диджей Иван

Для решения первой подзадачи было достаточно перебрать и проверить все возможные расстановки «медляков» за $O(2^N \cdot N)$.

Для решения второй подзадачи обозначим за $dp[i]$ ответ на задачу для префикса из треков от 1 до i включительно. Тогда $dp[N]$ будет являться ответом на задачу.

$dp[0]$ обозначим за 1, все остальные — за 0.

Теперь для каждого i от 1 до N включительно пройдемся по j от $i - 1$ до 0 включительно. Если сумма длин треков на отрезке $[j + 1; i]$ лежит в промежутке $[x; y]$, это значит, что нам подходят ответы, где стоят «медляки» после трека j и после трека i , и между ними нет других «медляков». (Для простоты можно считать, что плейлист начинается и заканчивается «медляком», это не влияет на ответ).

Для каждого подходящего j будем делать $dp[i] = (dp[i] + dp[j]) \% MOD$.

Сумму на отрезке $[j + 1; i]$ можно поддерживать и обновлять при обходе j .

Такое решение работает за $O(N^2)$.

Для полного решения оптимизируем решение второй подзадачи.

Для этого осчитаем префиксные суммы длин треков. Для каждого i при помощи бинарного поиска найдем l , начиная с которого сумма на отрезке $[l + 1; i] \leq y$, и r , начиная с которого сумма на отрезке $[r + 1; i] < x$. (Суммы на отрезке будем считать при помощи префиксных сумм). Тогда для текущего i подходят все j от l до $r - 1$ включительно.

Таким образом, нам нужно к $dp[i]$ прибавить все значения $dp[l] \dots dp[r - 1]$. Чтобы делать это быстро, будем поддерживать еще один массив префиксных сумм — для массива dp .

Такое решение работает за $O(N \log N)$.

Задача 6. Начинающий астроном

Для решения подгруппы с $N = 1$ нужно найти все возможные значения a для прямоугольника. Первое, что хочется сделать — посчитать значение a для вершин многоугольника. Однако к данному решению легко подобрать контрпример. Например, взять $b = 4$ и квадрат $(1, 4)$, $(3, 4)$, $(3, 6)$, $(1, 6)$. Тогда экстремальное значение $a = 1$, в этом случае парабола касается прямоугольника в точке $(2, 4)$, а остальные значения дают пересечения с прямоугольником. Поэтому будем искать максимальное или минимальное значение a для всей стороны.

Для начала получим, что парабола проходит через точку (x, y) при $a = \frac{bx - y}{x^2}$ или же $a = \frac{b}{x} - \frac{y}{x^2}$.

Рассмотрим случай вертикальной стороны (когда $x_1 = x_2$), тогда заметим, что в $a = \frac{b}{x} - \frac{y}{x^2}$ значение $\frac{b}{x}$ не изменяется, а во второй значение $-\frac{y}{x^2}$ линейно зависит от y , а значит, минимума и максимума a достигает в вершинах стороны.

Теперь рассмотрим случай, когда сторона не вертикальная. Тогда можно представить прямую, на которой лежит сторона, в виде $\alpha x + \beta$, где $\alpha = \frac{y_2 - y_1}{x_2 - x_1}$, а $\beta = y_0 - \beta \cdot x_1$. Подставим эту формулу в формулу для a : $a = \frac{bx - \alpha x - \beta}{x^2} = \frac{(b - \alpha)}{x} - \frac{\beta}{x^2}$. Найдя производную по x , получаем: $a'(x) = -\frac{(b - \alpha)}{x^2} + 2\frac{\beta}{x^3}$. Приравняв к 0, получаем: $-\frac{(b - \alpha)}{x^2} + 2\frac{\beta}{x^3} = 0$, преобразуя: $2\beta - (b - \alpha)x = 0$,

отсюда $x_c = \frac{2\beta}{(b-\alpha)}$. Теперь, если x_c лежит между x_1 и x_2 , то значение a лежит между $a(x_c)$ и $a(x_1)$ и $a(x_2)$, иначе a лежит между $a(x_1)$ и $a(x_2)$.

Альтернативно, a_c можно найти с помощью тернарных поисков.

Теперь решим задачу для больших N . Мы знаем, что для каждого прямоугольника у нас существует отрезок a , при котором парабола пересекает прямоугольник. Отметим эти отрезки (l_i, r_i) . Для каждого отрезка добавим 2 пары $(l_i, 1)$ и $(r_i, -1)$ и отсортируем их по возрастанию первой координаты, а в случае равенства — по убыванию второй. Тогда пройдемся по этому массиву и будем поддерживать сумму вторых координат — количество пересекаемых прямоугольников, а разница первых координат у соседей — количество пересечений. Таким образом, задача решается за $O(n \log n)$.