Задача A. Подготовка к олимпиаде

Для начала определим, что составить план подготовки нельзя, если $n > m$, так как в этом случае задачи закончатся раньше, чем наступит день олимпиады, а решать 0 задач в день нельзя.

Иначе, так как должно быть выполнено $a_i \leqslant a_j$ при $i \leqslant j$, можно поступить следующим образом:

- В первые $n-1$ дней решать по 1 задаче.
- В последний день решить $m-(n-1)$ задач.

У данной задачи также существуют и другие решения. Так, например, в первые $n-1$ дней можно решать по $\frac{m}{n}$ задач, а в последний — $\frac{m}{n} + (m \bmod n)$.

Задача B. Рафаэль и Кейси Джонс

Заметим, что Рафаэль и Кейси Джонс потратят хотя бы $\min(a_1, b_1)+\min(a_2, b_2)+\ldots+\min(a_n, b_n)$ времени, так как каждого бандита необходимо сразить. Вычтем $\min(a_i, b_i)$ из a_i и b_i для всех i. Теперь у нас остались два массива, при этом для каждого i верно, что либо $a_i = 0$, либо $b_i = 0$. Теперь задачу можно решать жадно, то есть выдавать 0 тому бойцу, у друга которого максимальное на данный момент значение времени. Как только кто-то получит n бандитов, надо отдать оставшихся его другу.

Авторское решение реализует этот подход с помощью сортировки бандитов по $a_i - b_i$, после чего выдаёт первых n Рафаэлю, а оставшихся Кейси Джонсу. Асимптотика решения $O(n \log n)$.

Задача C. Зет — это Самость, а Самость — это Зет

Заметим, что всегда существует такой путь для обоих персонажей, при котором каждое ребро будет пройдено только одним из них, то есть оригинал и клон будут двигаться по непересекающимся поддеревьям. В оптимальном случае оригинал и клон начинают в одной из вершин диаметра, а завершают путь в противоположных концах диаметра. В этом случае суммарное расстояние, которое они пройдут, будет равно $2 \cdot (n-1) - \mathrm{diam}$, где diam— это длина диаметра дерева. Можно доказать, что меньшего ответа достичь невозможно.

Задача D. Собеседование в Сбер

Поймем, что если на каком-то круге Ваня не решил ни одной задачи, то он и дальше не сможет ничего решить. Значит, Ваня сделает не больше n кругов решения задач. Тогда можем сделать решение за $O(n^2)$. Будем n раз идти по списку задач и решать задачи, которые можем.

Сделаем более быстрое решение. Запомним для каждой сложности задачи все позиции, где она встречается. Будем поддерживать структуру данных, в которой для каждой сложности задачи, меньше или равной текущему уровню Вани, запомним индекс нерешенной задачи, которую Ваня решит раньше всего. Изначально в ней лежит индекс самой первой 1 в массиве. Затем, пока структура не пустая, нужно найти минимальный индекс задачи, который больше последней решенной задачи, и достать его из структуры. После решения задачи навык увеличивается на 1, значит, должны добавить в структуру минимальный индекс задачи, равный текущему навыку и больший, чем текущий индекс. Аналогично нужно добавить задачу, у которой сложность равна сложности последней решенной.

Для решения будем использовать структуру данных std::set, тогда время работы будет $O(n \log(n))$, так как проводим $O(n)$ операций с сетом, и каждая требует $O(\log(n))$ действий.

Задача E1. Мышеловы атакуют (простая версия)

На самом деле достаточно сделать 20 запросов. Стоит вспомнить, что любое число представимо в двоичной записи. Нам известно, что младший (нулевой) бит числа равен 1 (p простое и $p \geqslant 3$). Попробуем угадать первый бит. Спросим, правда ли остаток от деления на 4 равен 1. При положительном ответе первый бит равен 0, при отрицательном — 1. Аналогично будем угадывать остальные биты по возрастанию, в качестве d будем использовать уже построенное число. Поскольку $p \leqslant 10^6 < 2^{20}$, за девятнадцать запросов мы спросим биты, соответствующие $2^1, 2^2, \ldots, 2^{19}$, а значит полностью восстановим число. Двадцатым запросом выведем ответ.

Задача E2. Мышеловы атакуют (сложная версия)

Решение простой версии не помогает решать сложную. В описанном ниже решении нам придётся сделать почти 1000 запросов.

Выберем числа m_1 и m_2, такие что $\gcd(m_1, m_2) = 1$, а $m_1 \cdot m_2 \geqslant 10^6$. Как их выбрать узнаем чуть позже. Проверим, является ли p делителем m_1 или m_2. Для этого спросим все простые делители m_1 с модулем m_1 и все простые делители m_2 с модулем m_2. Если угадали, выводим ответ.

Пусть мы знаем, что $p \equiv d_1 \mod m_1$ и $p \equiv d_2 \mod m_2$. В таком случае можно применить КТО (Китайскую теорему об остатках) и найти p. Чтобы узнать d_1 надо перебрать все числа, меньшие m_1 и взаимно простые с ним. Этого достаточно, так как в противном случае p имело бы нетривиальный делитель. Аналогично можно найти d_2.

Остаётся вопрос как выбрать m_1 и m_2. Нужно чтобы $\varphi(m_1)+\varphi(m_2)+\varepsilon \leqslant 1000$, где φ это функция Эйлера, а ε это суммарное количество простых делителей m_1, m_2 и вывод ответа. Прекальком можно найти подходящую пару чисел, например, $m_1 = 1155$, $m_2 = 1118$.