Задача A. Сумасшедшие транспортные налоги

Заметим, что таблица налоговых ставок упорядочена по строгому возрастанию мощностей, задающих диапазоны. Тогда для ответа на каждый запрос можно использовать двоичный поиск для поиска последней строки таблицы такой, что в ней мощность строго меньше, чем в запросе.

Далее нужно было умножить мощность из запроса на найденную ставку, и не забыть про 64-битный тип данных для результата. Итоговая сложность решения $O(m \log(n))$.

Задача B. Безумный танец

Пусть в массиве c_i мы будем считать, сколько раз Джокер произнёс i-ю цифру. Изначально этот массив заполнен нулями. Заметим, что каждую секунду каждый элемент этого массива не уменьшается, а хотя бы один элемент увеличивается. Поэтому сумма всех элементов этого массива каждую секунду строго увеличивается. Поэтому, Если танец Джокера конечен, то он не может длиться дольше секунд, чем сумма элементов массива b_i. А также заметим, что так как сумма элементов массива c_i строго увеличивается, мы можем с помощью бинарного поиска найти первый момент, когда она стала хотя бы такой же, как сумма элементов массива b_i. Тогда либо в этот момент эти массивы равны, и тогда ответ это этот момент, либо они не равны, и тогда Джокер будет танцевать вечно.

Осталось понять, как по моменту времени восстановить массив c_i. Пусть момент времени равен x. Для чисел с длиной меньше длины x мы знаем, что каждое из них будет произнесено. Для чисел с длиной равной длине x, нужно перебрать длину префикса числа, которая совпадает с префиксом x, перебрать следующую цифру, которая должна быть строго меньше соответствующей цифры в x, и тогда число может заканчиваться на любую последовательность цифр правильной длины.

Задача C. Ограбление банка

Эта задача может быть решена методом динамического программирования. Вычислим значение $dp[i][j]$ — количество последовательностей длины i, заканчивающихся на символ j, и соответствующих первым i числам из подсказки. Понятно, что $dp[1][a_1] = 1$, а все остальные $dp[1][i] = 0$. Для того, чтобы посчитать значение $dp[i][j]$, нужно сложить $dp[i-1][j-a_i]$ и $dp[i-1][j+a_i]$ (каждое из значений прибавляется только в том случае, если существует соответствующее состояние).

Задача D. Беспорядочное выступление

Будем решать задачу жадным алгоритмом. Очевидно, что порядок уменьшения порядочности не играет роли, а уменьшения порядочности разных людей не зависят друг от друга, поэтому суммарное внимание будет минимально, если мы будем уменьшать порядочность «самых наблюдаемых» зрителей, то есть тех, за которыми следит наибольшее число полицейских.

Для этого, для каждого зрителя найдем количество следящих за ним полицейских: заведем события $(+1, l)$ и $(-1, r)$ для каждого наблюдаемого отрезка зрителей $[l, r)$. Эти события можно отсортировать за $\mathcal{O}(n \log n)$ или за $\mathcal{O}(n)$ с помощью подсчета, так как ограничения на n позволяют это сделать. После этого, обработав события в порядке их следования и посчитав на них префиксные суммы, мы получаем для каждого зрителя количество полицейских, следящих за ним.

Осталось только в порядке уменьшения этих значений изменять порядочность зрителей на минимум из их текущей порядочности и оставшейся харизмы k. Алгоритм завершаеися, если все зрители имеют порядочность 0 или если $k = 0$.

Задача E. Сумасшедшее домино

Несложно привести подходящую расстановку шашек. Например, для чётных n:

```
#..#
....
#..#
....
```

Для нечётных n:

```
#....
....#
#....
....#
#....
```

Шашки в крайних столбцах делают замощение доминошками однозначным.

Задача F. Вырваться из окружения

Разделим задачу на 4 квадранта относительно клетки с Джокером. Найдем клетки, Манхэттенское расстояние до которых равно d, находящиеся на границе. Внутри квадранта искомые клетки лежат на одной диагонали, а значит, зная две крайние клетки, можно найти количество клеток, принадлежащих этой диагонали

Задача G. Убийственная математика

Заметим, что из каждой пары (a, b) мы можем получить четыре различных результата. Сделаем *bfs* по такому неявному графу и получим минимальное «расстояние» от исходной пары до пары с двумя одинаковыми элементами.

Данный алгоритм работает за время $\mathcal{O}(b^2)$. Количество операций до достижения ситуации $a = b$ меньше $\log_2 b$, так как есть последовательность изменений, в которой расстояние между числами каждый раз уменьшается хотя бы в два раза (всегда заменять b на меньшее из двух средних, например). А тогда максимальное число состояний *bfs* равно $4^{\log_2 b} \sim b^2$.

Так же за ту же асимптотику работает метод динамического программирования по подотрезкам, где $\mathtt{dp}_{i,j}$ равно минимальному числу операций до достижения пары равных чисел. Пересчет происходит за $\mathcal{O}(1)$, так как из каждого состояния есть не более четырех переходов.

Тем не менее, есть жадный алгоритм, так же решающий эту задачу, но за $\mathcal{O}(b)$, который каждый раз заменяет b на минимальное из двух средних, однако сложность его доказательства не позволяет рассматривать его как ожидаемое решение, и поэтому все правильные решения за $\mathcal{O}(b^2)$ получали вердикт «OK».