Задача A. Математичное сложение

У нас есть уравнение: $\frac{x}{u} + \frac{y}{v} = \frac{x+y}{u+v}$.

Давайте домножим левую и правую часть на $u * v * (u + v)$.

Получили: $x * v * (u + v) + y * u * (u + v) = (x + y) * u * v$.

После раскрытия скобок и упрощения имеем: $x * v^2 + y * u^2 = 0$.

Одним из решением этого уравнения является $x = -u^2$, $y = v^2$

Задача B. Покраска прямоугольников

Прямоугольники после разрезания будут покрашены в шахматную раскраску. Значит, если площадь четна, то количество клеток разных цветов одинакова, а если нечетна, то отличается ровно на 1.

Найдем, какую часть занимают закрашенные клеточки по отношению ко всем. Для четной площади это отношение всегда $\frac{1}{2}$. Для нечетной $\frac{S-1}{2 \cdot S}$. Чем меньше нечетная площадь, тем меньше отношение. Площаль равную 1 получить нельзя, поэтому лучшее отношение равно $\frac{1}{3}$ и достигается при площади равной 3.

Тогда получим нижнюю оценку на ответ: $answer \geqslant n \cdot m \cdot \frac{1}{3}$. Отлично!

Мы знаем, что ответ целый, поэтому если у нас получится построить такое разрезание, что небходимо покрасить ровно такое cnt клеточек, что $\frac{n \cdot m}{3} \leqslant cnt < \frac{n \cdot m}{3} + 1$, то cnt будет ответом. Ведь cnt - минимальное целое значение удовлетворяющее оценке.

Если одна из сторон делится на 3, то очевидно как резать на прямоугольники 1×3 и получить идеальный ответ.

Если остатки сторон 1 и 1 или 2 и 2, то можно разрезать на прямоугольники 1×3 и один прямоугольник с площадь 4, в котором надо закрасить 2 клеточки. Тогда ответ тоже подходит под оценку.

Если остатки сторон 1 и 2, то после разрезания на прямоугольники 1×3 останется прямоугольник 1×2, в котором надо покрасить одну клеточку. Ответ тоже подходит под оценку.

Для всех пар остатков есть способ построить ответ удовлетворяющий неравенству. Поэтому ответ равен $\lceil \frac{n \cdot m}{3} \rceil$

Задача C. Два массива

Давайте сначала отсортируем оба массива.

Теперь посмотрим на минимальные элементы в обоих массивах. Очевидно, что, если $a_1 + 1 < b_1$ или $a_1 > b_1$, то ответ «No». Если $a_1 = b_1 = x$, то в массиве a после всех преобразований должен быть элемент x. Можем оставить a_1 без изменений. Если же $a_1 + 1 = b_1$, то можем увеличить a_1 на 1. В обоих случаях задача сводится к задаче с массивами меньшей длины.

Следуя этой логики для остальных элементов, получаем, что достаточно проверить, что $a_i = b_i$ или $a_i + 1 = b_i$ для всех $1 \leqslant i \leqslant n$.

Итоговая сложность решения $O(nlog(n))$.

Задача D. Угадай перестановку

Заметим, что кол-во инверсий на убывающем отрезке длины l равно C_l^2.

Так как мы развернули два непересекающихся подотрезка, то кол-во инверсий для любого отрезка последовательности кол-во инверсий будет равно кол-во инверсий на кусках развернутых отрезков, которые в него входят, а они убывают.

Давайте первым запросом спросим кол-во инверсий $A := C_{k-j+1}^2 + C_{j-i}^2$ во всей последовательности, на это мы потратим 1 запрос.

Теперь посмотрим, что будет если мы спросим кол-во инверсий на отрезке $[x, n]$. Если оно меньше, чем A, значит не оба подотрезка вошли целиком, то есть $i < x$, иначе $i \geqslant x$.

Зная это, мы можем применить бинарный поиск, чтобы найти i. На это мы потратим $log_2(n)$ запросов.

Давайте, теперь спросим кол-во инверсий на отрезке $[i+1, n]$, назовем это число — B. На это мы потратили еще 1 запрос. Из конструкции понятно, что $A - B = |\{x|x > i, a_x < a_i\}| = ||[i + 1, j - 1]|| = j - i - 1$.

Теперь мы можем вычислить $j - i$, j и $\frac{2}{j-i}$, зная это и A получаем C^2_{k-j+1}.

Теперь остается решить квадратное уравнение относительно $k-j+1$, найдя эту длину, получаем k.

Итого мы потратили $log_2(n) + 2 \leqslant 32$ запроса, но мы дали вам чуть больше, на случай если в какой-то из частей алгоритма ваше решение тратит на несколько запросов больше.

Задача E. Игра с камнями

У этой игры есть жадная стратегия: берем первую кучу, так как у нее всего один сосед, то Боб обязан все её камни удалять в парах с камнями из следующей. Если следующей кучи нет, или в ней меньше камней, то Боб проиграл. Иначе, Боб делает первую кучу пустой и убирает соответствующее число камней из второй кучи. Теперь первую простую кучу можно просто выкинуть, без изменения сути игры. Боб побеждает, если на момент, когда осталась одна куча, она уже пуста.

Давайте, будем итеративно строить массив c, где c_i — кол-во камней оставшихся в i-й куче, после того, как Боб жадно опустошил кучи $1, \ldots, i-1$. Пусть $c_0 = 0$, тогда $c_i = a_i - c_{i-1}$.

Пусть массив c целиком состоит из неотрицательных чисел, тогда на любом ходу Боб смог опустошить соответствующую кучу, иначе рассмотрим первый момент t, когда $c_t < 0$, это значит, что до этого момента Боб справлялся опустошать кучи согласно жадной стратегии, а тут произошло $c_{t-1} > a_t$, значит он проиграл. Чтобы проверить, что в конце осталось пустая куча, достаточно проверить $c_n = 0$.

Таким образом мы получили критерий выигрышности последовательности — $c_i \geqslant 0$ для всех i, и $c_n = 0$.

Если мы раскроем рекурсивную запись c_i, то получим $c_i = a_i - a_{i-1} + a_{i-2} - \ldots + (-1)^{i-1} \cdot a_1$.

Мы будем решать задачу для каждой левой границы l по отдельности. Определим последовательность $a^l := a_l, a_{l+1} \ldots a_n$, $a^l_i = a_{l+i-1}$. Для нее посмотрим соответствующий массив c^l. Найдем в нем позицию первого отрицательного числа t ($c^l_t < 0$), если такая существует, иначе положим $t = n - l + 2$. И теперь найдем найти кол-во нулей, на отрезке $[1, t-1]$ массива c^l, что даст нам кол-во выигрышных подотрезков вида $[l, r]$. Просуммирую такие кол-ва по всем l, получим ответ на задачу.

Заметим, что $c^l_i = a^l_i - a^l_{i-1} + \ldots + (-1)^{i-1} \cdot a^l_1 = a_{l+i-1} - a_{l+i-2} + \ldots + (-1)^{i-1} \cdot a_l = c_{l+i-1} + (-1)^{i-1} \cdot c_{l-1}$.

Заметим, что $c^l_i < 0$, тогда и только тогда, когда $_{l+i-1} < (-1)^{i-1} \cdot c_{l-1}$. Теперь, разобьем задачу отдельно на четные и нечетные индексы. Теперь вместо отрицательности будет «меньше x». Найти первую позицию можно множеством способов, например, спуском по дереву отрезков.

Заметим, что $c^l_i = 0$, тогда и только тогда, когда $_{l+i-1} = (-1)^{i-1} \cdot c_{l-1}$. Задача также разбивается на четные и нечетные индексы. Вместо кол-во нулей теперь надо искать «кол-во равных x». Мы предлагаем для каждого значения c_i запомнить позиции в которых оно встречается (для разных четностей индексов отдельно), и делать бинарные поиски по этим наборам вхождений.

Итоговая асимптотика решения: $\mathcal{O}(n \cdot log(n))$

Задача F. Странная НОП

Введем граф на вершинах (c, msk), где chr это какой-то символ, а $mask$ это n-битная маска вхождений (i-й бит единичный, тогда и только тогда, когда мы рассматриваем второе вхождение символа c в i-ю строку). Заметим, что не все маски возможны для каждого c так как буква может входит меньше, чем 2 раза. Заметим, что вершина определяет выбор символа и его позиций из каждой строки.

Заметим, что в графе $\mathcal{O}(|\Sigma| \cdot 2^n)$ вершин.

Введем строго сравнение вершин ($<$): $(chr1, msk1) < (char2, msk2)$ тогда и только тогда, когда позиции выбранные первой вершиной строго левее выбранных второй (во всех строках). Введем нестрогое сравнение ($\leqslant$) также, но разрешим некоторым позициям совпадать. Заметим, что оба строгое сравнение анти-рефлексивно, оба сравнения транзитивны и выполняется: $(a \leqslant b < c \Rightarrow a < c)$.

Граф будет содержать все ребра из строго меньших вершин в большие и только их. Заметим, что в силу транзитивности и антисимметричности, граф ацикличен.

Заметим, что существует биекция между путями в графе и общими подпоследовательностям. Вершинная длина пути будет совпадать с длиной общей подпоследовательности.

Мы хотим посчитать динамику $dp(c,\ msk)$ – длина наибольшего пути, исходящего из этой вершины. Эта dp легко считается на DAG, но ребер слишком много. Мы хотим убрать некоторые ребра так, чтобы dp не изменилось. Заметим, что если при удаление ребра из какой-то вершины, dp в ней не поменялось, то оно не поменялось и в других вершинах.

Посмотрим на произвольную вершину (c, msk), из которой исходит хотя бы одно ребро и наидлиннейший исходящий из нее путь (в нем есть хотя бы одно ребро): $[(c, msk) \to (c2, msk2) \to \ldots]$. Пусть существует такая маска $MidMsk \neq msk2$, что: (c, mask) $<$ $(c2,\ MidMsk) \leqslant (c2,\ msk2)$. По свойствам сравнений, мы можем поменять вторую вершину наидлиннейшего пути на $(c2,\ MidMsk)$.

Для фиксированного c найдем такую маску, что соответствующие позиции выбраны как можно левее в каждой из строк, так чтобы оставаться правее позиций, выбранных $(c,\ msk)$. Такая маска ищется за $\mathcal{O}(n)$. Из показанного ранее, мы можем считать, что наидлиннейший путь ($c2$ зафиксирован) проходит через нее. Поэтому мы можем удалить все остальные ребра, идущие из этой вершины в вершины с символом $c2$, но другой маской. Можем сделать так для всех символов.

Теперь в графе каждая вершина имеет $\mathcal{O}(|\Sigma|)$ исходящих ребер, что позволяет посчитать динамику быстро. Восстановить строку по динамике уже не сложно.

Заметим, что граф здесь это абстракция для понимания, в решении он не нужен.

Заметим, что можно даже не считать все значения динамики, достаточно лишь найти $dp(c, 0)$ для всех c, это можно показать похожими рассуждениями.

Итоговая асимптотика решения: $\mathcal{O}(n \cdot |\Sigma|^2 \cdot 2^n)$

Задача G. Допустимые отрезки

Расстояние от точки P до отрезка $[AB]$ равно максимуму из расстояния от точки P до луча $[AB)$ и расстояния от точки P до луча $[BA)$.

Зафиксируем точку P_i. Теперь нужно найти все точки P_j, что расстояние от всех точек $P_k (1 \leqslant k \leqslant n)$ до луча $[P_i, P_j)$ не больше R.

Для каждой точки P_k построим касательные из точки P_i к окружности с центром в точке P_k и радиусом R. Эти касательные образуют угол A_k. Расстояние от точки $P_k (1 \leqslant k \leqslant n)$ до луча $[P_i, P_j)$ не больше R тогда и только тогда, когда луч $[P_i, P_j)$ лежит внутри угла A_k. Построим все углы A_k и пересечем их. Теперь для всех $1 \leqslant j \leqslant n$ нужно проверить, что луч $[P_i, P_j)$ лежит внутри пересечения углов.

Итоговая сложность решения $O(n^2)$.