Задача 1. Утраченный символ

Для полного решения задачи нужно было заметить, что если длина левой подстроки строго меньше правой, и правая подстрока не начинается с нуля, то данное разбиение корректно. Если длина левой подстроки строго больше, разбиение нам сразу не подходит. В случае, когда длина N нечетна, этого замечания полностью достаточно для решения. Если длина N четна, нужно дополнительно рассмотреть случай, когда длины подстрок равны. Разбиение подойдет нам, если левая подстрока лексикографически меньше или равна правой. Асимптотика решения — $O(\log_{10} N)$

Задача 2. Реверс

Группа 1: ограничения очень небольшие, поэтому можно запросы моделировать явно и после каждого запроса считать, сколько клеток каждого цвета на поле проходом по всем клеткам. Асимптотика $O(N * M * Q)$

Группа 2: можно заметить, что после каждого запроса изменяются не более чем $O(N + M)$ клеток, соответственно можно поддерживать количество клеток каждого цвета и обновлять эти счетчики за $O(1)$ при обновлении каждой клетки. Асимптотика $O((N + M) * Q)$

Задача 3. Популярная песня

В данной задаче необходимо было посчитать количество вхождений фрагментов мелодии как подстроки в изначальную запись песни.

Для решения первой группы тестов необходимо посчитать вхождение каждого из 7 символов в запись песни. После чего можно быстро суммировать произведение количества вхождений каждого фрагмента на его популярность.

Во второй группе тестов не было ограничения на длину фрагментов, но их не могло быть больше 10. Для решения этой группы тестов достаточно написать решение, которое явно сравнивает все подстроки.

Для решения третьей группы тестов нужно ускорить сравнение строк с помощью хешей.

Есть несколько способов полного решения задачи. Один из них — построить структуру данных Бор на всех подстроках длины 20 исходной записи песни. После чего можно быстро отвечать на вопрос — сколько раз каждый фрагмент мелодии входит как подстрока в исходный текст.

Задача 4. Странный вычислитель

Вычислитель можно представить в виде ацикличного ориентированного графа. Значение выражения в клетке можно представить в виде пары (a, b), представляющую выражение $a \cdot x + b$. Клетка с переменной имеет вид $(1, 0)$, с константой $(0, a)$. Топологически отсортируем граф. Тогда можно посчитать значение в клетке, просто выполним операции в клетке от предшественников, для которых эти значения уже посчитаны. Ответ будет находиться в последней клетке в списке отсортированных. Это работает за $O(N + M)$. После этого можно за $O(1)$ вычислять значение выражения. Итоговое время работы $O(N + M + Q)$.

Частичное решение 1: граф является деревом, соответственно значения в клетках можно не запоминать явно, а считать рекурсивно. Ограничения позволяют не использовать многочлен, а считать явно. $O((N + M) * Q)$

Частичное решение 2: Использовать многочлен в решении 1. $O(N + M + Q)$.

Частичное решение 3: Переменных нет, соответственно ответы на все запросы будут одинаковы и можно нужно выполнить только 1 запрос. $O(N + M + Q)$

Задача 5. Правильные многоугольники

Покажем, что единственная возможная фигура — квадрат.

Если k-угольник нельзя составить из отрезков, то kx-угольник для $x > 1$ также нельзя составить, иначе в нем можно будет выбрать k вершин, образующих правильный k-угольник.

Покажем теперь, что k-угольников не существует, при нечетном k.

Предположим, что он существует, длина стороны l, причем она наименьшая. Пусть многоугольник задается последовательностью векторов (a_i, b_i), a_i, b_i — целые. Для них выполняются следующие равенства $a_i^2 + b_i^2 = l^2$, $\sum_{i=1}^{k} a_i = 0$, $\sum_{i=1}^{k} b_i = 0$.

Возведем в квадрат последние два равенства и раскроем скобки:
$(\sum_{i=1}^{k} a_i)^2 = \sum_{i=1}^{k} a_i^2 + 2\sum_{i=1}^{k}\sum_{j=i+1}^{k} a_i \cdot a_j = 0$, $(\sum_{i=1}^{k} b_i)^2 = \sum_{i=1}^{k} b_i^2 + 2\sum_{i=1}^{k}\sum_{j=i+1}^{k} b_i \cdot b_j = 0$.
Просуммируем полученные равенства: $\sum_{i=1}^{k} a_i^2 + 2\sum_{i=1}^{k}\sum_{j=i+1}^{k} a_i \cdot a_j + \sum_{i=1}^{k} b_i^2 + 2\sum_{i=1}^{k}\sum_{j=i+1}^{k} b_i \cdot b_j = 0$.
Преобразуем его: $\sum_{i=1}^{k}(a_i^2 + b_i^2) + 2\sum_{i=1}^{k}\sum_{j=i+1}^{k}(a_i \cdot a_j + b_i \cdot b_j) = 0$.

Заметим, что первый член можно выразить как kl^2, а второй как $2x$. Отсюда, $kl^2 = -2x$. Так как k – нечетный, то l^2 – четный. Если l^2 кратен 4, то любой a_i и b_i кратен 2, а значит l — не наименьший. Получаем, что все a_i и b_i — нечетные, значит x четный (для каждой пары i, j: $a_i a_j + b_i b_j$ четно, так как каждое слагаемое нечетное). Значит, l^2 должен делиться на 4. Следовательно, многоугольников с нечетным количеством сторон не существует.

Покажем, что восьмиугольника не существует. Пусть есть два последовательных вектора длины l: (a, b) и (c, d). Их скалярное произведение равно: $ac + bd = l^2 \cos 45°$. Заметим, что левая часть выражения — целое число, а правая часть — иррациональное. Пришли к противоречию.

Получаем, что единственный допустимый многоугольник — квадрат.

Для того, чтобы посчитать количество квадратов, преобразуем отрезки таким образом, чтобы конец находился в верхней полуплоскости. Разделим все отрезки по длине и направлению. Например, их можно хранить в словаре. После этого, для каждой группы отрезков, найдем группы, с которыми можно образовать квадрат. Так как все отрезки пронумерованы, то для группы получаем $N_1(N_1 - 1)N_2(N_2 - 1)$ различных квадратов. Итоговая асимптотика $O(n \log n)$

Задача 6. Грандиозный праздник

Для решения первой подгруппы достаточно написать простой перебор за $O(3^{N \cdot M})$

Для решения второй группы можно использовать метод динамического программирования по профилю, где профиль — заполненность ряда.

Введем функцию $f(mask1, mask2)$ — количество переходов, чтобы из профиля $mask1$ получить профиль $mask2$. Посчитать значение можно побитово, для пары единичных бит из $mask2$ можно поставить двух участников размера 1×1 или одного участника размера 2×2, если соответственные биты в $mask1$ нулевые. Итоговое время работы $O(N \cdot 2^{2N} \cdot M)$.

Для решения третьей группы можно заметить, что ответ для пустого блока длины M можно посчитать с помощью бинарного возведения в степень матрицы f из решения для предыдущей группы. Итоговое время работы $O(2^{3N} \cdot \log M)$

Заметим, что каждый столбец со статуями «отсекает» от площади блок, возможно, нулевой длины. Таких блоков не более чем $K + 1$. Так же заметим, что для матриц верно следующее равенство: $v \times (A \times B) = (v \times A) \times B$, здесь v — вектор количества способов, A и B — некоторые матрицы. Умножение вектора на матрицу можно производить за $O(2^{2N})$. Таким образом, итоговое время работы: $O(2^{2N} \cdot \log M \cdot K)$.