Задача A. Аналогичная сортировка

Так как все числа попарно различны, то элементы обоих массивов можно пронумеровать целыми числами от 1 до N строго по возрастанию. Тогда задача сводится к тому, чтобы расположить во втором массиве числа в том же порядке номеров, как и в первом.

Будем сортировать пары чисел — первое число в паре содержит значение элемента, второе является вспомогательным значением. Прочитаем первый массив: первое число в i-й паре содержит i-й элемент, второе равно i. Прочитаем второй массив (что содержится во втором элементе каждой пары, неважно). Отсортируем оба массива по возрастанию первого числа в паре (то есть по возрастанию элементов заданных массивов). Затем перенесём значения второго числа каждой пары из первого массива во второй в том порядке, в котором элементы массива расположены после сортировки и отсортируем второй массив по возрастанию второго числа в паре.

Получится тот же самый порядок относительно возрастания, который исходно был в первом массиве.

Для первой подзадачи достаточно даже квадратичного алгоритма сортировки.

Для второй подзадачи достаточно отсортировать второй массив по возрастанию.

Задача B. Голубь и игра с числами

Давайте поймем, что у соседних чисел, всегда НОД будет равен 1, по этому давайте просто разобьем все наши числа парами вида $i * 2 - 1$ и $i * 2$, и теперь, когда Сегун будет вычеркивать число, мы будем вычеркивать число из его пары, по итогу в массиве останется 2 соседних числа и мы победим.

Задача C. 1.5 G-modem

Так как Боб получает строку в случайном порядке символов, то все строки, состоящие из одного и того же набора символов, можно считать эквивалентными. Выберем представление, в котором цифры в строке отсортированы по возрастанию.

То есть сообщение имеет вид «a_0 нулей, a_1 единиц... a_9 девяток».

Первая идея — задавать первую цифру числа как a_0, вторую — как a_1 и так далее. Тогда суммарная длина строки равна s, где s — сумма цифр. Максимальная длина строки тем самым равна 90. Остаётся число 0, которое кодируется пустой строкой, но его можно закодировать произвольной строкой длины 91. Это даёт более 50 баллов.

Заметим, что при таком кодировании строка, содержащая 10 нулей (или 10 каких-то других одинаковых цифр), не соответствует никакому числу.

Поэтому 10 одинаковых цифр можно использовать в качестве метки, обозначающей другой тип кодирования. Например, 10 единиц задают число 0, а 10 нулей задают «инвертированный» метод кодирования: первой цифре соответствует $9 - a_0$, второй — $9 - a_1$ и так далее. Тогда суммарная длина строки равна $10 + (90 - s)$, где s — сумма цифр.

Если мы будем кодировать числа с суммой цифр до 50 включительно основным способом, а числа с суммой цифр от 51 до 90 — инвертированным, то в первом случае используется не более 50 знаков, во втором — не более 10+(90-51)=49. Получается, что количество знаков не превосходит 50 (напомним, что ноль мы кодируем 10 единицами), что даёт полный балл.

Заметим, что оптимальное решение этой задачи строится комбинаторным путём: количество строк требуемого вида длины N вычисляется через «метод перегородок»; после чего подбирается наименьшее такое N, что количество строк превосходит 10^{10} и далее задача сводится к построению комбинаторного объекта по номеру и номера по комбинаторному объекту.

Однако в ограничениях задачи данный способ является слишком трудоёмким.

Задача D. Санитарный контроль

Геометрическая формулировка задачи выглядит следующим образом: дано множество квадратов со стороной $2n$, сторонами, параллельными осям координат и точкой пересечения диагоналей в точке $(0, 0)$. Требуется найти количество точек, в которых отрезок пересекает контуры заданных квадратов.

Введём на плоскости манхэттенскую метрику относительно прямых $x = y$ и $x = -y$ (то есть относительно диагоналей). Тогда i-й квадрат — это множество точек, для которых расстояние от

начала координат по заданной метрике (то есть сумма координат при переходе к прямым $x = y$ и $x = -y$ в качестве координатных осей) равно $i \cdot sqrt(2)$ (или i, если считать единичный отрезок равным диагонали единичного квадрата).

Очевидно, что все целочисленные точки (включая и концы отрезков) находятся на сторонах какого-то из таких квадратов (а именно, квадрата со стороной $2 \cdot max(|x|, |y|)$). Назовём номер этого квадрата «порядком» точки.

Если отрезок вертикальный или горизонтальный, то ответ либо равен бесконечности (если отрезок перекрывается с каким-то из отрезков вида $[(n, n), (n, -n)]$, $[(n, -n), (-n, -n)]$, $[(-n, -n), (-n, n)]$ или $[(-n, n), (n, n)]$), либо равен абсолютной величине разности порядков начала и конца отрезка плюс единица. Перекрытие легко проверяется классическим алгоритмом проверки перекрытия отрезков на прямой.

В противном случае содержащая отрезок прямая пересекается хотя бы с одной диагональю и расстояния от центра до точек пересечения различны (если точка пересечения одна, то второе расстояние можно считать бесконечно большим). Минимальное из этих расстояний, таким образом, всегда определяется однозначно и является расстоянием от начала координат до данной прямой.

Если соответствующая точка пересечения лежит вне отрезка или на его концах, то ответ равен абсолютной величине разности порядков начала и конца отрезка плюс единица. Если точка пересечения лежит внутри отрезка и является целой, то ответ складывается из разности порядков начала отрезка и точки пересечения, разности порядков конца отрезка и точки пересечения и самой точки пересечения. Если точка пересечения лежит внутри отрезка и не является целой, то наименьший пересекаемый квадрат — это квадрат со стороной $2 \cdot ([|x|] + 1)$, где x — x-координата точки и пересечение происходит в двух точках, то есть ответ складывается из разности порядка начала отрезка и $[|x|]$ и разности порядка конца отрезка и $[|x|]$.

Первая группа тестов, очевидно, не может содержать тесты с бесконечным ответом (так как прямые, содержащие стороны невырожденных квадратов из условия задачи, не проходят через центр), а малое количество точек позволяет найти ответ как количество пересечения отрезка с отрезками, заданными сторонами квадратов, не переходя к манхэттенской метрике, или просто как разность модулей наибольших координат минус единица, если догадаться в том или ином виде до основной идеи задачи.

Во второй группе тестов добавляются тесты с бесконечным ответом; впрочем, решение попрежнему возможно «в лоб» (проверкой пересечения отрезка со сторонами квадратов), при этом все вычисления остаются в 32-битном целом типе данных.

Третья группа — последняя, в которой ещё работает решение проверкой пересечения отрезка со сторонами квадратов; вычисления уже требуют 64-битного целого типа данных; при неоптимальных способах проверки пересечения возможно превышение лимита времени.

При реализации полного решения расстояние до прямой в манхэттенской метрике может быть найдено с помощью приближённых алгоритмов (двоичный или троичный поиск); в этом случае при неаккуратной реализации возможны проблемы с точностью, так что целочисленное решение в целом выглядит предпочтительнее.

Задача E. Эмалированное дерево

Сперва заметим, что ответ не может быть меньше, чем максимальная степень вершины в дереве + 1. Действительно, все соседи и сама вершина попарно связаны между собой в построенном графе G.

Оказывается, граф G можно раскрасить так, что данная нижняя граница достигается. Начнем раскрашивать граф в порядке поиска в глубину. Пусть мы находимся в вершине v. Рассмотрим все выходящие ребра из нераскрашенного ребенка u, которые связывают u с уже раскрашенной вершиной. Таких ребер не больше $deg\ v$. Раскрасим u в минимальный подходящий цвет, он точно не больше $deg\ v + 1$.

Задача F. Почти палиндромы

Первым делом избавимся от необходимости отдельно рассматривать случаи с «почти палиндромами» четной длины. Это можно сделать, если между любыми двумя символами в строке, а также

перед первым и после последнего вставить специальный символ, который в строке не встречается. Например, подойдет «#». В полученной строке каждому «почти палиндрому» нечетной длины с центром в обычном символе соответствует «почти палиндром» нечетной длины в обычной строке, а «почти палиндрому» с центром в специальном символе соответствует «почти палиндром» четной длины. Таким образом мы свели задачу к поиску «почти палиндромов» нечетной длины.

Переберем центр «почти палиндрома». Дальнейшее решение можно разбить на две части:

1. Найдем максимальный палиндром с заданным центром. Если мы попробуем добавить к нему еще один символ слева и справа от палиндрома, то столкнемся либо с границами строки – в таком случае ни одного «почти палиндрома» с данным центром не существует. Либо мы столкнемся с отличающимися символами. В таком случае «почти палиндром» с данным центром точно существует, и следует перейти к шагу 2:

2. После нахождения первого различия следует продолжить добавлять символы с обеих сторон от найденного «почти палиндрома» до тех пор, пока они будут одинаковы. Таким образом, можно найти общее количество «почти палиндромов» с данным центром.

Однако, такое решение будет работать за $O(n^2)$, что слишком долго в данных ограничениях. Заметим, что в каждом из пунктов решения нам необходимо начиная с некоторых номеров l, r найти максимальные совпадающие строки, идущие в разных направлениях: в первом случае $l = r =$ позиции центра, а во втором l и r задают границы найденной в первом случае строки. Такую задачу можно решать, используя различные строковые алгоритмы.

Самым простым решением является использование бинарного поиска и хешей. Бинарным поиском мы будем искать длину совпадающих строк, а при помощи хешей сравнивать их. Итоговое решение работает за $O(n \cdot \log(n))$.

Существуют и другие решения. Например, можно через разделитель записать перевернутую строку – получится строка «S» + «\$» + «S^R». В таком случае мы свели задачу к поиску наибольшего общего префикса у каких-то двух суффиксов. Её классическим решением является построение суффиксного массива, массива LCP на нем и запросами минимума на отрезке. Полученное решение также будет иметь асимптотику $O(n \cdot \log(n))$.

Можно также добиться линейной асимптотики, если над полученной строкой построить суффиксное дерево и искать наибольший общий префикс за $O(1)$ в оффлайне, используя алгоритм Тарьяна для поиска наименьшего общего предка.

Отдельно стоит заметить, что в любом решении первый шаг можно проделать при помощи алгоритма Манакера.

Задача G. Быстро? Ещё быстрее!

Будем искать N в виде $2^{2^k} - 1$.

Заметим, что

$$2^{2^k} - 1 = \prod_{i=0}^{k-1} (2^{2^i} + 1)$$

Для того, чтобы возвести в степень 2^x, требуется x умножений, так что (с учётом одного дополнительного умножения на $+1$), всего получается $\sum\limits_{i=0}^{k-1} (2^i + 1) = 2^n + n - 1$ умножение. Быстрое возведение в степень даст $2^{k+1} - 2$ умножений, то есть выигрыш составляет $2^k - k - 1$ умножение. Для $k = 6$ получается экономия 57 умножений (то есть та, которая требуется). При этом $N = 2^{64} - 1$.

Получить промежуточные результаты можно, например, взяв решение для максимального результата и заменив последнее умножение на необходимое число умножений на предыдущие множители.

Задача H. Опять математика...

Решение на 1 группу. Нужно в лоб выбрасывать i-й элемент в последовательности, а затем вычислять всё по рекурсивной формуле: $g^{a_1 a_2 \dots a_n} = g^{a_1^{a_2 \dots a_n}} = \dots$, используя на каждом шаге бинарное возведение в степень. Такое решение работает за $O(n^2 \log A)$.

Решение на 2 группу. Малая теорема Ферма утверждает, что если m — простое, а $g \neq 0$, то $g^a \equiv g^{a \bmod (m-1)} (\mathrm{mod} m)$. Воспользуемся этим фактом, вычислив префиксные $(a_1, a_1 a_2, \ldots)$ и суффиксные степени по модулю $m-1$. Далее лего вычислить ответ с помощью префиксных и суффиксных степеней, а так же с помощью бинарного возведения в степень. Работает за $O(n \log A)$.

Решение на 3 группу. Есть некоторое обобщение МТФ, которое называется теоремой Эйлера. Она утверждает, что если g и m взаимнопросты, то $g^a \equiv g^{a \bmod (\phi(m)-1)} (\mathrm{mod} m)$. Далее идея аналогична предыдущей подзадаче. Решение работает за $O(\sqrt{A} + n \log A)$ из-за вычисления функции Эйлера.

Полное решение. Для полного решения необходимо воспользоваться методом divide-and-conqueror. Рассмотрим решение на примере массива из 8 элементов. Разобьём его на 2 части и вычислим $B_{5..8} = g^{a_5 a_6 a_7 a_8}$ и $B_{1..4} = g^{a_1 a_2 a_3 a_4}$. Выпишем их следующим образом:

$B_{5..8} B_{5..8} B_{5..8} B_{5..8} B_{1..4} B_{1..4} B_{1..4} B_{1..4} B_{1..4}$

Все вычисления мы сделали за $O(n/2 \log A + n/2 \log A) = O(n \log A)$.

На следующем шаге разобьём массив уже на 4 части, в каждой части будем опять считать вспомогательные B: $B_{3..4,5..8} = g^{a_3 a_4 a_5 a_5 a_7 a_8} = (B_{5..8})^{a_3 a_4}$ и $B_{1..2,5..8}, B_{1..4,7..8}, B_{1..4,5..6}$ аналогично. Это вычисляется за $O(4 \cdot (n/4) \log A)$. Далее опять выпишем их в ряд следующим образом:

$B_{3..4,5..8} B_{3..4,5..8} B_{1..2,5..8} B_{1..2,5..8} B_{1..4,7..8} B_{1..4,7..8} B_{1..4,5..6} B_{1..4,5..6}$.

На следующем шаге мы аналогично уже вычислим ответ. Каждый шаг работает за $O(n \log A)$, всего нужно сделать $\log n$ шагов. Итого решение работает за $O(n \log n \log A)$.