Задача A. Асимметрия нежелательна

Покажем, что ни у какой фигуры ответ не может быть больше 4. Если у фигуры есть ось симметрии, то сторона единичного квадрата также должна переходить в какую-то сторону. Но угол между двумя отрезками, симметричными относительно прямой, равен удвоенному углу между отрезком и прямой, а так как отрезки могут быть или горизонтальными, или вертикальными, то оси симметрии могут или также быть горизонтальными и вертикальными, или должны идти под углом 45 градусов к линиям сетки. Так как в одном направлении может быть не более одной оси симметрии, то количество осей не может быть более 4. Четыре оси симметрии можно получить всегда.

Задача B. Больше команд в Питер!

Устанавливаем по умолчанию $B = 2$, $T = 0$. Если первая строка равна `org`, присваиваем $B = 4$, иначе проверяем, что t не менее 2 или что s не менее 30, в этом случае $B = 3$. Далее вычисляем T в соответствии с тем, что написано в правилах, и выводим в качестве ответа сумму $T + B$.

Задача C. Выплаты за сезон

При чтении данных каждого из 15 игроков считаем зарплату за каждый из четырёх сезонов. Если контракт на меньшее количество сезонов, чем 4, заполняем оставшиеся сезоны минимальными контрактами. После чего считаем сумму по каждому году и в случае, если она больше потолка, прибавляем к итоговому результату разность между суммой и потолком.

Задача D. Гарри Поттер и длинные числа

Будем считывать число как строку (слева направо) и достраивать по формуле $v(p + 1) = 10 \cdot v(p) + d_{p+1}$, где $v(i)$ — значение числа после прочтения i знаков, d_i — i-я цифра числа, после чего сразу же брать остаток от деления на k.

Если число можно разбить на блоки из ℓ цифр и меньше, то какой-то блок будет находиться с левого края, так что если ни после одной из первых ℓ цифр не получился остаток 0, то Гарри Поттер ответит, что число не делится. Иначе обнуляем счётчик позиций при первом обнулении остатка и снова считаем до ℓ. Если мы дошли до конца числа — Гарри даст правильный ответ. В противном случае проверяем, делится ли n на k. Если делится — то выводим 0 (так как ответы не совпадут), иначе выводим 1.

Задача E. Движение по гладкой траектории

Самолёт следует по гладкой траектории (в определении задачи) тогда и только тогда, когда он движется по звену ломаной между (x_i, y_i) и $(x_{i+1}, y_{i+1}$, обладающему следующим свойством: точки (x_i, y_i), $(x_{i+1}, y_{i+1}$ и (x_n, y_n) лежат на одной прямой. Поэтому считываем все точки и проверяем, что ориентированная площадь параллелограмма, то есть косое (псевдоскалярное) произведение векторов $(x_i - x_n, y_i - y_n)$ и $(x_{i+1} - x_n, y_{i+1} - y_n)$ равно нулю. Если это так — прибавляем длину отрезка к ответу. Заметим, что ответ всегда будет ненулевым, так как для отрезка перед посадкой один из концов отрезка совпадает с точкой (x_n, y_n), и длина этого отрезка будет обязательно учтена.

Задача F. Евклид и Большой Сфинкс

Всего есть 8 вариантов магического квадрата 3×3, получающихся из квадрата

```
2 9 4
7 5 3
6 1 8
```

композицией четырёх поворотов и одного отражения (это можно легко найти, перебрав все 9! вариантов расстановки чисел).

Рассмотрим позиции $(1, 2)$ и $(2, 1)$. На этих позициях могут находиться пары (7,9), (9,7), (1,3), (3,1), (9,3), (3,9), (7,1), (1,7). Заметим, что, так как Сфинкс может исказить информацию только один раз, минимум одно число из пары будет передано неискажённым. Второе будет либо передано неискажённым, либо поменяет чётность, то есть мы заведомо знаем, какое число искажено. Далее заметим, что для каждого числа его возможные соседи отличаются минимум на 4 (для 9 это 7 и 3,

для 7 – 9 и 1, для 3 – 9 и 1, для 1 – 7 и 3), то есть даже с искажением на 1 можно восстановить второе число пары. Ну а далее по двум позициям и обязательной 5 в центре восстанавливаются $(3, 2)$ и $(2, 3)$, а затем и значения в угловых клетках.

Заметим, что если в запросе будет фигурировать хотя бы одна угловая клетка (тем более центральная), то адаптивный интерактор всегда сможет создать ситуацию множественного ответа.

Задача G. Ёлочные слова

Количество попарно различных ёлочных слов длины k над алфавитом размера n — это количество способов выбрать упорядоченный набор из k элементов среди n, то есть $n!/k!$, или $n \cdot (n-1) \cdot \ldots \cdot (n-k+1)$.

То есть требуется проверить, можно ли представить число в таком виде. Предпросчитаем все числа для $k > 3$; для $k = 3$ их будет не более 10^6 (так как $(10^6+1) \cdot 10^6 \cdot (10^6-1) < 10^{18}$, а следующее уже больше), для последующих k ещё меньше, поэтому можно хранить `map`, значением которого является пара (n, k). Если при вычислении оказалось, что представление x уже есть, мы апдейтим значение только в случае, когда новое n меньше старого (если считаем «снизу вверх», то просто не трогаем те x, для которых значение уже есть).

Если число не нашлось в построенном `map`, то остаются варианты $k = 2$ и $k = 1$ (который возможен всегда, так как $n!/(n-1)! = n$). При $k = 2$ ответ будет меньше, так что надо проверить этот вариант, то есть проверить, имеет ли квадратное уравнение $a^2 - a - x = 0$ относительно a целые корни, и если имеет — вывести положительный корень в качестве n и 2 в качестве k, иначе вывести x в качестве n и 1 в качестве k.

Задача H. Жадный Скруджиус

Решим задачу для $n \leqslant 4\,000$ методом динамического программирования за $O(n^2)$: $dp_i = \min\limits_{j=1}^{\min(i,3999)} dp_{i-j} + rmn(j)$, где $rmn(j)$ — длина соответствующей римской записи.

Далее заметим, что, начиная с 4000, для всех n верно утверждение $dp_{i-1000} = dp_i + 1$ (то есть добавляем в сумму M за каждую 1000): это легко понять, если заметить, что для обычных римских чисел единственные три варианта, при которых можно выиграть от представления числа в виде суммы, это IX+IX вместо XVIII, а также аналогичные XC+XC вместо CLXXX и CM+CM вместо MDCCC (и производные от таких действий типа MCM+CM вместо MMDCCC). Поэтому просто представляем число в виде $x + 1000k$, где $3000 \leqslant x \leqslant 3999$, после чего ответ равен $k + dp[x]$.

Задача I. Заповедник

Давайте сначала с помощью алгоритма Дейкстры найдем расстояния от особых вершин до всех, а затем от всех до особых(для этого развернем ребра и запустим Дейктру от особых). Для ответа на каждый запрос будем использовать классическое дп по битмаскам, как в задаче про коммивояжера.

Задача J. История одного собеседования

Давайте решим задачу, когда $k = 0$. Пусть у нас не круг, а массив. Тогда можно посчитать динамику $dp(v, f)$ – наибольшая сумма ребер в паросочетании, если мы рассмотрели v вершин и $f = 0$ указывает на то, что последняя вершина не входит в паросочетание, а $f = 1$ – на то, что входит. Для того, чтобы решить задачу на круге, возьмем любое ребро, и сначала предположим, что оно не войдет в паросочетание – тогда у нас предыдущая подзадача, а потом возьмем его в паросочетание – тогда можно аналогично решить этот случай такой же динамикой, но с другими начальными значениями.

Пусть $k > 0$. Так как $k \leqslant 20$, мы можем перебрать подмножество дополнительных ребер, которое войдет в ответ. Проверим, что никакая вершина не принадлежит нескольким выбранным ребрам одновременно. Пусть мы выбрали $s \leqslant k$ ребер. Тогда весь круг разобьется на участки, концы которого – вершины, являющиеся концами ребер из выбранного подмножества, а между ними – цепочка вершин, для которой нужно знать ответ – наибольшее по весу паросочетание. Заметим, что таких цепочек не более $2 \cdot s$.

Нам нужно быстро находить ответ на каждой цепочке, для этого выделим все особые точки изначально – это такие вершины, которые являются концом хотя бы одного дополнительного ребра, а

затем, начиная из каждой особой вершины, предпросчитаем динамикой ответ до каждой следующей вершины(не обязательно особой). Сложность решения – $O(n \cdot k + 2^k \cdot k)$.

Задача K. Йогафон Лайт, Йогафон Про

Давайте решим более простую версию задачи, когда все стоимости равны 1. Будем идти слева направо и поддерживать стек открывающихся скобок. Если мы встречаем открывающуюся скобку, добавим её в стек. Если закрывающуюся, то посмотрим на стек – если он не пуст, удалим с вершины стека одну скобку и увеличим ответ на 2.

В нашей версии задачи стоимости могут быть любыми. Давайте попробуем действовать по похожему алгоритму. Если мы встречаем закрывающуюся скобку, нам нужно поставить ей в соответствие максимальную имеющуюся в стеке открывающуюся скобку. Но что делать, если стек пуст? Не всегда будет правильным решением просто пропустить текущую закрывающуюся скобку, так как мы могли найти пару какой-то встретившейся ранее закрывающейся скобке, которая по цене была меньше, и выгоднее сделать замену – поменять ту на нашу. Давайте поддерживать еще одну структуру для закрывающихся скобок, для которых мы нашли пару. Если для нашей закрывающейся скобки нет пары, то есть стек открывающихся скобок пуст, просто посмотрим на нашу структуру для закрывающихся скобок – если она не пуста, берем оттуда минимум и меняем его на нашу скобку в случае, если её стоимость меньше нашей. Вместо стека открывающихся скобок будем использовать мультисет, также мультисет будем использовать и для закрывающихся скобок, для которых нашли пару.

Сложность – $O(n \cdot logn)$.

Задача L. Квадратность

Будем перебирать значение k достаточности подотрезка, поддерживая массив s из нулей и единиц, где $s_i = 1$, если $a_i \geqslant k$, и $s_i = 0$ иначе. Массив можно разбить условно на нерасширяемые блоки подряд идущих единиц и нулей. При увеличении k единицы будут исчезать, а нули добавляться. Будем поддерживать также значение to_i – ближайшая справа позиция такая, что на отрезке $[i, to_i]$ ровно k единиц. Предположим пока, что k только растет, и удалений единиц нет(то есть зафиксирована некоторая последовательность 0 и 1). При этом при увеличении k значения to_i будут расти, в некоторых случаях они просто увеличатся на 1(когда $s_{to_i+1} = 1$), а в других нам нужно будет перепрыгнуть через блок из нулей в поисках следующей единицы. Для первого случая будем понимать, что мы увеличили на k суммарно значения to по умолчанию. Для второго создадим события, для каждой единицы введем значение $time_i$ – в какой момент времени k-я единица справа от позиции i должна искаться через блок нулей, а не быть следующей. $time_i$ легко посчитать для фиксированной единицы, достаточно найти ближайший справа 0 к to_i.

Наблюдение: мы можем проходиться по всем таким единицам в событиях. Это так, поскольку мы прыгаем через блок нулей, получается, каждый раз рассматривая новую инверсию, а их не более n суммарно.

Вернемся к исходной задаче.

Пусть теперь единицы убираются, нули добавляются. Справа от добавляемого нуля все так же – там случай выше. Теперь, мы можем пройтись по всем единицам слева от добавляемого нуля(опять же, каждая пара «единица слева – ноль справа» образует инверсию, а их суммарно не более n) и пересчитать значения to_i для них(можно поддерживать сет единиц и пересчитывать to_i с помощью пары $upper_bound$-ов). Что делать с нулями? Будем хранить в CHM компоненты из нулей. Значение to_i для $s_i = 0$ равно значению to_j для $s_j = 1$, где j – позиция ближайшей справа от i единицы. Мы можем пройтись также по всем компонентам нулей слева от добавляемого нуля, так как их не больше, чем единиц, и пересчитать to для компонент. В итоге, нам необходимо поддерживать сумму значений to, по ней можно восстановить ответ. Сложность – $O(n \cdot \log(n))$.

Задача M. Лидеры разбиений

В этой задаче нужно посчитать сумму максимумов по всем разбиениям числа n на m положительных слагаемых.

Посчитаем для каждого x количество разбиений, в которых все числа не превосходят x. По этой информации можно восстановить ответ, то есть нам нужно посчитать разбиения на слагаемые, в которых каждый элемент ограничен сверху.

Посчитаем, наоборот, разбиения, в которых есть слагаемое, большее x. Применим принцип включений-исключений: переберём количество элементов, больших x, пусть их k. Если у нас k слагаемых больших, чем x, давайте вычтем из n величину kx. Получим теперь, что нужно разложить число $n - kx$ на m слагаемых, каждое из которых больше нуля. Эта подзадача решается методом шаров и перегородок через количество сочетаний.

Заметим, что $k \cdot x \leqslant n \to k \leqslant n/x$, и сумма k по всем нужным слагаемым даёт итоговую сложность $n \log n$.