Задача A. Смена стиля

Автор задачи: Николай Будин, разработчик: Ильдар Загретдинов

Одно из решений представляет из себя построчное считывание данных, то есть считываем очередную строку и изменяем ее по следующим правилам: если есть заглавная буква, которая находится в начале строки, то заменяем ее на последовательность из символа «_» и соответствующей строчной буквы, если же она в начале строки, то только на соответствующую строчную букву. После этой операции выводим измененную строку.

Задача B. Результаты контеста

Автор задачи: Николай Будин, разработчик: Григорий Хлытин

Заметим, что мы можем игнорировать посылки, которые содержат вердикт `CE` или посылки по задаче, которая уже была сдана. Давайте для каждой задачи хранить количество неудачных попыток и была ли она сдана к текущему моменту. Таким образом, если мы видим первую успешную посылку по задаче, мы прибавим к ответу штраф за эту задачу, вычисленный по заданной формуле $t + 20 \cdot k$, где t — время первой успешной сдачи задачи в минутах, k — количество неправильных попыток перед первой успешной сдачей.

Задача C. Собака, предатель и кабеля

Автор задачи: Владислав Кузнецов, разработчик: Дмитрий Гнатюк

Решим задачу методом динамического программирования. Посчитаем для каждой клетки следующее значение — сколько проводов может максимум съесть собака на пути до этой клетки. Обозначим это значение за $\texttt{dp}[i][j] = \max(\texttt{dp}[i-1][j] + x, \texttt{dp}[i][j-1] + y)$, где x и y обозначают наличие или отсутствие провода между соответствующими клетками.

Теперь для каждого расположения человека посчитаем сколько ходов сделает собака. Это можно сделать простой формулой: если человек был в клетке x, y, то собака сделает $\left\lfloor \frac{x+y+1}{2} \right\rfloor$ ходов.

Осталось только посчитать максимум динамики на соответствующей диагонали, не забывая, что собака не может пройти выше или правее изначального положения человека. Ограничения в задаче позволяют считать этот самый максимум за линейное время, что дает нам полное решение задачи.

Задача E. Электронный замок

Автор задачи: Николай Будин, разработчик: Арсений Кириллов

Заметим, что мы всегда хотим получить более длинное число, так как длинное число больше короткого. А это значит, что на каждую цифру мы хотим включать как можно меньше сегментов. Меньше всего сегментов у цифры 1, их всего два. А значит длина итого числа будет $\lfloor \frac{n}{2} \rfloor$. Однако если число n нечётное, то у можно включить ещё один сегмент. Тогда надо превратить одну из единиц в семёрку, и разумеется, чтобы получить максимальное число, мы превратим первую единицу в семёрку. И того для чётных n мы получим число $111\ldots111$, а для нечётных число $7111\ldots111$.

Задача F. Похожие имена

Автор задачи: Николай Будин, разработчик: Даниил Орешников

Первым действием стоит применить стандартный трюк, который используется, когда надо работать с циклическими сдвигами строки s — взять конкатенацию строки с самой собой (ss) и рассматривать все ее подстроки длины $|s|$. В частности, если нас интересует префикс некоторого циклического сдвига, то в строке ss нас просто интересуют любые подстроки длины не больше $|s|$.

С меньшими ограничениями эту задачу можно было бы решить следующим образом: перебрать циклический сдвиг самой короткой строки и применить префикс-функцию. С имеющимися ограничениями задачу можно было решить, применив хеши или суффиксный массив и алгоритм Касаи. Далее будет приведено решение с суффиксным массивом.

Давайте сконкатенируем вместе продублированные строки из ввода, разделив их специальным символом, не встречающимся в алфавите (например, `'$'`). Заведем также массив `from`, отвечающий за номер исходной строки, которой соответствует каждый символ конечной строки, полученной в результате конкатенации.

Построим суффиксный массив. Заметим, что если ответ на задачу — a, то в полученной строке должны быть одинаковые подстроки длины a. Переформулируем это как «в строке есть суффиксы с общим префиксом длины a, с началами в каждой из исходных строк». Если применить алгоритм Касаи, мы сможем посчитать `lcp` — наибольший общий префикс соседних суффиксов в суффиксном массиве. Получается, что ответ — это такое a, что в массиве `lcp` есть отрезок лежащих подряд (потому что суффиксный массив лексикографически отсортирован) значений $\geqslant a$, соответствующие которым индексы из суфф. массива покрывают все множество исходных строк.

Для поиска максимального подходящего a можно воспользоваться двоичным поиском. Внутри цикла для конкретного значения a_0 достаточно пройтись по массиву `lcp` и для каждого отрезка из значений не менее a_0 проверить, что значения `from` на нем покрывают все множество $1 \dots n$. Также стоит искусственно ограничить ответ сверху минимальной из длин исходных строк, чтобы избежать ситуации, когда трюк с конатенацией увеличил потенциальный ответ.

Время работы решения — $\mathcal{O}(n \log n)$ на построение суффиксного массива и столько же на проход по `lcp` с бинпоиском.

Задача G. Теория Рамсея

Автор задачи и разработчик: Михаил Иванов

Теория Рамсея говорит, что для любых двух чисел L и K существует такое число R, что в любом графе на R вершинах есть либо L-клика, либо K-антиклика. Наименьшее такое число называется *числом Рамсея* и обозначается $R(L, K)$.

Мы предлагаем такое решение: взять любые $R(L, K)$ вершин графа (а если в графе меньше вершин — то взять все вершины), оставить только рёбра, проведённые между выбранным подмножеством, и в полученном графе перебрать все L-множества и K-множества вершин и проверить, что они подходят. Этого достаточно, так как, если $N < R(L, K)$, то мы переберём все нужные множества, а если $N \geqslant R(L, K)$, то мы в любом случае найдём требуемое множество.

Для всех $1 \leqslant K, L \leqslant 5$ выполняется $R(K, L) \leqslant 48$, поэтому перебрать достаточно C_{48}^5 множеств, что меньше двух миллионов и запросто можно успеть сделать за отведённое программе время.

Более того, если просто поверить, что $R(5, 5)$ достаточно мало, то можно не знать, насколько именно оно мало. Достаточно было поверить, что ограничения допускают решение с перебором всех множеств вершин до какого-то предела R, и написать программу, которая в бесконечном цикле перебирает подмножества всё большего набора вершин, каждый раз проверяя, что у неё ещё есть, скажем, полсекунды до истечения ограничения по времени.

Задача H. Супер-счастливые билетики

Автор задачи: Орешников Даниил, разработчик: Николай Будин

Так как билетик имеет длину не превышающую 6, а различных билетиков 10^n, можно перебрать все билетики и для каждого проверить, является ли он супер-счастливым.