Задача A. Aware of depth

Авторы задачи: Кузляев Никита, Ревин Дмитрий

Это классическая задача на «жадность» + «два указателя». Отсортируем список l по невозрастанию, поставим правый указатель $right$ на самый правый элемент списка и начнем итерироваться левым указателем $left$ с начала списка.

Если мы встречаем веревку, которая не меньше d (то есть $d \leqslant l[left]$), то прибавляем к счетчику $+1$ и перемещаем левый указатель на $left + 1$; иначе двигаем правый указатель до тех пор, пока $left < right$ или пока не найдем $l[right]$, что $d \leqslant l[left] + l[right] - 1$ — в таком случае прибавляем к счетчику $+1$, и при любом исходе сдвигаем $left$ и $right$. Когда левый указатель «зайдет» за правый, останавливаемся и смотрим значение счетчика — если он меньше n, то выводим «NO», иначе «YES».

Также можно немного оптимизировать в случае $m < n$ — тогда ответ всегда «NO».

Сложность решения: $O(m \times logm)$.

Задача B. Bark Beetles

Автор задачи: Кузляев Никита

Оптимальной стратегией является «запустить» заражение из всех изначально зараженных гектаров и посмотреть, куда оно может добраться, при условии, что заражение распространяется только по болотистым гектарам. Тогда обработке подвергаются только лесные гектары, соседние по стороне с зараженными (так, например, нет необходимости обрабатывать весь «лес», а только его края, также возможен случай, когда гектары леса «экранируют» другие лесные гектары внутри).

Для хранения ответа выделим список res из $n + m - 2$ элементов, изначально равных нулю. res_i будет равно числу гектаров леса, которые будут заражены ровно через i дней.

Применим обход в ширину (bfs): добавим в очередь координаты изначально зараженных гектаров и отметим их как «посещенные» за 0 дней. Далее, пока очередь не пустая, будем брать из нее координату очередного гектара (такой гектар, который был заражен в прошлом и помещен в очередь), для этого гектара посмотрим на соседние с ним по стороне: если соседний гектар является «непосещенным» лесом, то увеличим на единицу res_i, где $i-1$ — это число дней, за которые заражение добралось до текущего гектара, помечаем гектар как «посещенный», но не добавляем в очередь; если же мы встречаем «непосещенный» гектар болота, то добавляем его в очередь, помечаем его как «посещенный» при этом сохраняя число ходов, за которые мы до него добрались.

На самом деле, нас просят вывести другие числа — не те, что сейчас в res, но мы можем сделать преобразование $res_i = res_i + res_{i-1}$ последовательно для всех i от 2 до $n + m - 2$.

Сложность решения: $O(nm)$.

Задача C. Call him ВИТЁК

Автор задачи: Кузляев Никита

Заметим, что нет смысла для каждой команды хранить ее балл от каждого члена жюри — нам важна только их сумма до прихода шестого члена жюри.

Переберем все команды (имена «V», «I», «T», «E», «K») в указанном порядке. Когда команда $name$ может иметь шанс на победу? — когда она получает 5 баллов, а остальные столько, чтобы не иметь в сумме больше нее. Как тогда раздавать баллы другим командам? — Из жадности (можете это доказать), выгодно дать другой команде, у которой больше всего баллов, 1 балл, затем второй по величине 2 балла и т. д. Если команда $name$ получает не меньше остальных, то она имеет шансы на победу (потому что раз существует хотя бы один способ раздать баллы, чтобы команда выиграла, то шансы есть).

Все, что нужно, — это найти команды с именами, не равными $name$, и отсортировать их по баллам. Далее, реализовать проверку, описанную выше. И так для каждой команды. Также не забываем выводить «!» в конце, если случилось так, что все команды имеют шансы на победу.

Сложность решения: $O(1)$.

Задача D. Dilemma

Автор задачи: Михайлов Глеб

Ключевая идея: если в строке есть хотя бы один дубликат (повторяющаяся буква), то ответ 1. Если все символы уникальны, то ответ – длина строки. Также необходимо рассмотреть особый случай: если длина строки $p > 26$, то ответ 1. Для проверки на уникальность символов можно использовать set.

Итоговая сложность: $O(n)$.

Задача E. EcoHomes

Автор задачи: Кузляев Никита

Сразу заметим, что решения нет в случаях $n = 2$ и $n = 3$, иначе решение существует, но мы не будем это доказывать. Существует множество подходов к решению этой задачи: рассмотрим здесь только один. Введем обозначение «серого» и «белого» треугольников — они располагаются так, что соседние по стороне треугольники всегда разного цвета, а самый верхний треугольник является «серым».

Шаг 1: «серые» треугольники на уровнях от 1 до $n - 1$ сверху вниз и слева направо заполним последовательными числами, начиная с единицы.

Шаг 2: продолжим заполнение со следующего числа — заполним «белые» треугольники на уровнях от 2 до $n - 1$ сверху вниз и слева направо также последовательными числами.

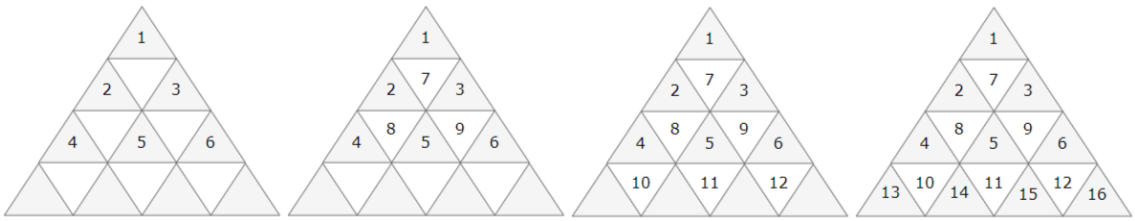

Иллюстрация к шагам 1—4.

Шаг 3: продолжим заполнять «белые» треугольники на самом нижнем уровне слева направо.

Шаг 4: заполним «серые» треугольники на самом нижнем уровне слева направо.

Сложность решения: $O(n^2)$.

Задача F. Finding Technodromes

Автор задачи: Кривеженко Иван

Задача решается динамическим программированием по подстрокам. Пусть $f(i, j)$ равно наименьшему числу символов, которое нужно исправить в подстроке $s[i \dots j]$ (то есть от i-го до j-го символа включительно), чтобы она стала палиндромом. В частности, $f(i, i) = 0$ — строка из одного символа является палиндромом.

Теперь вычислим $f(i, j)$, если $i < j$. Если $s[i] = s[j]$, то $f(i, j) = f(i + 1, j - 1)$. Иначе, если $s[i] \neq s[j]$, то $f(i, j) = f(i + 1, j - 1) + 1$.

Ответом является количество таких подстрок, то есть количество таких пар (i, j) $(0 < i < j < k)$, что $f(i, j) < n$.

Сложность решения: $O(k^2)$.

Задача G. Group guide

Автор задачи: Михайлов Глеб

Заранее вычислим все факториалы до 100! по модулю $10^9 + 7$, чтобы иметь быстрый доступ при вычислениях.

Переберем все возможные конфигурации групп: рассмотрим все возможные сочетания групп по 2 и 3 человека. Для каждой конфигурации проверим, что она допустима (не остаётся "лишних"студентов), убедимся, что количество групп не превышает количество работ.

Вычислим число перестановок. Для каждой допустимой конфигурации посчитаем: сколькими способами можно разбить студентов на группы и сколькими способами можно распределить работы между группами. Перемножим эти значения для получения числа вариантов для данной конфигурации.

Будем использовать деление через умножение на обратный элемент (по малой теореме Ферма), чтобы ускорить подсчет. Все вычисления выполняются по модулю $10^9 + 7$.

Итоговая сложность: O(n+m)

Задача H. Hack this shift

Автор задачи: Кузляев Никита

«Скрытым» алгоритмом является сдвиг каждой буквы вправо на значение ее индекса. Подразумевается, что после «d» идет «e», а после «e» снова «a». Так, строка «ae» преобразовывается в «bb», поскольку «a», так как имеет индекс 1, сдвигается на 1 и превращается в «b», а буква «e» сдвигается на 2 вправо и тоже превращается в «b».

Автор понимает, что задача не является в полной мере «олимпиадной», но убежден, что до решения способен дойти каждый, и что тестового примера для этого достаточно. Также намек на решение есть в названии задачи («shift» — англ. сдвиг, перенос).

Сложность решения: $O(|s|)$.

Задача I. Interactive measure

Автор задачи: Кузляев Никита

Можно задать $n - 1$ запросов вида «`? 0 x`» для всех x от 1 до $n - 1$. Можно доказать, что этого достаточно, чтобы получить полное представление о величине перегородок. После i-го такого запроса мы знаем высоту перегородок от 0 до i. Например, за первый запрос мы узнаем высоту первой перегородки с индексом 1 (правее перегородки с индексом 0 и высотой 100). Затем, зная высоту первой перегородки (и нулевой, равной 100), а также объем жидкости, который можно уместить между перегородками 0 и 2, мы можем определить высоту второй перегородки. И так далее. Алгоритм нахождения высоты очередной перегородки здесь приведен не будет, но вы можете его изучить из кода авторского решения (надеемся, у вас есть ссылка).

Забавно, что обратная задача, которую решает интерактор — а именно быстрое нахождение объема жидкости между двумя перегородками при их известной длине, является куда более сложной. (На самом деле не решает, потому что $n \leqslant 1000$)

Сложность решения: $O(n)$.

Задача J. Jackbox at the party!

Автор задачи: Кузляев Никита

Найдем все делители числа k перебирая возможные делители от 1 до $\sqrt{k}$. Сохраним их в словарь $count$, и посчитаем $count[q]$ — количество чисел в a, которые делятся на q для всех q (всех делителей k).

Пройдемся еще раз по списку a. Пусть текущее число a_i может участвовать в произведении с некоторым другим числом a_j ($i \neq j$). Сколько таких чисел j существует для i? — Если $a_i \times a_j$ делится на k, то a_i дает некоторый «вклад», а именно $\gcd(a_i, k)$, теперь достаточно узнать, сколько чисел в a делятся на $\frac{k}{\gcd(a_i,k)}$ — это $count[\frac{k}{\gcd(a_i,k)}]$.

Таким образом, для каждого i можем быстро считать число возможных «пар». Стоит учитывать, что индексы i и j могут быть посчитаны по два раза (как (i, j) и (j, i)), также необходимо не учитывать a_i в $count$.

После подсчета числа пар с произведением, которое делится на k, нужно поделить на общее число пар, чтобы получить вероятность.

Сложность решения: $O(n \times d(k) + nlog(k))$, где $d(k)$ — оценка роста числа делителей от величины k, но так как $k \leqslant 10^5$, можем считать $d(k) = 128$, поскольку максимальное число делителей среди чисел меньше 10^5 имеет число 83160 с 128 делителями.

Задача K. Kubiki for TikTok House

Автор задачи: Кузляев Никита

Самое важное — уместить все кубики того цвета, который встречается наибольшее число раз. Один из способов — начать ставить кубики этого цвета в перемешку с другими в первом слоте. Если такое получилось — ок! Иначе у нас есть еще возможность поставить один такой кубик в третий слот. Если получилось — ок! Иначе задача не имеет решения и надо вывести «NO»

Сложность решения: $O(n)$.

Задача L. Lunar debate

Автор задачи: Михайлов Глеб

Переберем все возможные пары точек. Для каждой пары точек (i, j) вычислим евклидово расстояние, сохраняя уникальные расстояния в массив distances, так как они будут использоваться в качестве возможных значений для h.

Поиск минимального h реализуем через бинарный поиск: Минимальное значение h будет равно 0, а максимальное будет равным максимальному расстоянию между любой парой точек. На каждой итерации бинарного поиска будем брать среднее значение $h = distances[(low + high)/2]$.

Проверку связности осуществим через поиск в глубину. Построим граф, добавляя ребро между двумя точками, если расстояние между ними $\leqslant h$.

Выведем найденное значение h с необходимой точностью (до 10^{-9}).

Итоговая сложность: $O(log(n^2) \cdot (1 + n^2))$.

Задача M. MSIT MStIT

Автор задачи: Кузляев Никита

Будем решать задачу, используя идею динамического программирования. Пусть $dp_{i,j}$ — минимальное количество денег для решения задачи, если у нас не n рядов, а только первые i, а высота последнего ряда равна j.

Но высоты рядов очень большие. Как нам эффективно перебирать j? — Во первых, заметим, что если мы изменяем высоты каких-то рядов, то до такой высоты, которая есть у другого ряда, иначе мы переплачиваем. Во вторых, заметим, что $n \leqslant 1000$, а значит, что уникальных j порядка не 10^9, как величины h_i, а 10^3.

Сделаем маппинг: отсортируем все h_i по неубыванию. Пусть такие отсортированные значения в G. Теперь j это не сама величина h, а индекс высоты в отсортированном массиве G. Тогда $dp_{i,j} = \min(dp_{i,j}, dp_{i,k \leqslant j} + c_i \cdot |G_k - h_i|)$. Итоговый ответ в $dp_{n,x}$, где x такое число от 1 до n, что $dp_{n,x}$ минимально (перебираем x).

Сложность решения: $O(n^2)$.