

Разбор задач

1 Чайная лавка в Чэнду

Условие задачи

Путешествуя по Китаю, Ильнур оказался в старинной чайной лавке в городе Чэнду. В этой лавке чай готовят особым способом.

Сначала мастер наливает в чашку несколько одинаковых порций крепкого чайного настоя. Каждая порция имеет объём D миллилитров. Мастер может налить любое количество порций, но итоговый объём настоя всегда должен быть кратен D .

После этого мастер доливает горячую воду до полного объёма чашки.

У каждого посетителя есть своя чайная чашка объёма C миллилитров. Также посетитель сообщает, сколько воды он хотел бы получить в напиток: не меньше A миллилитров и не больше B миллилитров.

Таким образом, если мастер налил k порций настоя, то:

- объём настоя равен $k \cdot D$ миллилитров,
- объём воды равен $C - k \cdot D$ миллилитров.

Ильнуру стало интересно, может ли мастер выбрать такое количество порций настоя, чтобы объём воды оказался в желаемом диапазоне.

Вам даны числа A , B , C и D . Определите, существует ли целое число $k \geq 0$, для которого выполняется условие

$$A \leq C - kD \leq B.$$

Идея решения

Переформулируем условие. Требуется определить, существует ли целое число $k \geq 0$ такое, что

$$A \leq C - kD \leq B.$$

Здесь kD — объём настоя, а $C - kD$ — объём воды.

Ключевое преобразование

Перенесём выражение с k в другую сторону:

$$A \leq C - kD \leq B.$$

Это равносильно системе неравенств

$$C - B \leq kD \leq C - A.$$

Таким образом, задача сводится к проверке существования неотрицательного целого k , для которого число kD лежит в отрезке

$$[C - B, C - A].$$

Наблюдение

Все возможные значения объёма настоя имеют вид

$$0, D, 2D, 3D, \dots$$

то есть это кратные числа D .

Нужно проверить, существует ли кратное D , которое попадает в интервал

$$[C - B, C - A].$$

Минимальное кратное D , не меньшее чем $C - B$, равно

$$\left\lceil \frac{C - B}{D} \right\rceil \cdot D.$$

Если это значение не превосходит $C - A$, то существует подходящее значение k , иначе — нет.

2 Чередующийся маршрут

Условие задачи

Ильнур планирует путешествие по Китаю и выписал города в том порядке, в котором хочет их посетить.

Каждый город относится к одному из двух типов:

- R — город с историческими достопримечательностями;
- B — город с парками.

Типы городов заданы строкой s длины n , состоящей только из символов R и B.

Ильнуру нравится, когда маршрут выглядит разнообразно. Поэтому он хочет найти как можно более длинный участок маршрута, на котором типы городов чередуются. Это означает, что на таком участке любые два соседних города имеют разные типы.

Необходимо определить длину самого длинного непрерывного отрезка, на котором города чередуются.

Идея решения

Требуется найти максимальную длину непрерывного отрезка строки, на котором соседние символы различны, то есть выполняется чередование типов городов.

Наблюдение

Будем идти по строке слева направо и поддерживать длину текущего максимального суффикса, заканчивающегося в позиции i , на котором символы чередуются.

Обозначим эту длину через cur .

- Если $s_i \neq s_{i-1}$, то чередование продолжается, и можно увеличить текущую длину:

$$cur = cur + 1.$$

- Если $s_i = s_{i-1}$, то чередование ломается, и новый подходящий отрезок может начинаться только с позиции i , поэтому

$$cur = 1.$$

На каждом шаге достаточно обновлять ответ как максимум из всех значений cur .

Почему этого достаточно

Для каждой позиции i значение cur равно длине наибольшего чередующегося отрезка, который оканчивается в этой позиции.

Действительно:

- если текущий символ отличается от предыдущего, то любой корректный отрезок, заканчивавшийся в $i - 1$, можно продолжить;
- если символы совпадают, то никакой отрезок длины больше 1, заканчивающийся в i , уже невозможен.

Значит, максимум по всем таким значениям и будет ответом задачи.

Особые случаи

- Если $n = 1$, ответ равен 1.
- Если все символы одинаковы, ответ также равен 1.
- Если вся строка полностью чередуется, ответ равен n .

3 Чайная церемония

Условие задачи

Путешествуя по Китаю, Ильнур посетил традиционную чайную церемонию. Во время церемонии чайные мастера расставляют гайвани на специальном подносе, имеющем форму прямоугольной решётки размером $N \times M$.

В каждой позиции (i, j) находится одна гайвань. В неё кладут некоторое количество мини-точа — небольших прессованных чайных шариков. Количество мини-точа в гайвани обозначим через $A_{i,j}$.

Мастер объяснил Ильнуру важное правило расстановки. Для любых двух гайваней, расположенных в соседних клетках подноса (то есть имеющих общую сторону), суммарное количество мини-точа в них должно быть нечётным.

Формально, если клетки (i, j) и (x, y) являются соседними по стороне, то есть выполняется $|i - x| + |j - y| = 1$, то должно выполняться условие, что $A_{i,j} + A_{x,y}$ — нечётное число.

Иногда в некоторых гайванях оказывается недостаточно мини-точа, и правило может нарушаться. Ильнур может добавлять мини-точа в любую гайвань (то есть увеличивать значение $A_{i,j}$), однако удалять мини-точа нельзя.

Требуется изменить значения $A_{i,j}$ так, чтобы указанное правило выполнялось для всех соседних пар гайваней, при этом суммарное количество добавленных мини-точа было минимальным.

После внесения изменений выведите итоговую таблицу значений $A_{i,j}$.

Идея решения

Сумма двух чисел нечётна тогда и только тогда, когда одно из них чётное, а другое нечётное.

Значит, в любой корректной итоговой таблице чётности соседних клеток должны различаться. Но на прямоугольной решётке это означает ровно шахматную раскраску: все клетки одного цвета должны иметь одну чётность, а все клетки другого цвета — противоположную.

Следовательно, возможны только два варианта:

- клетки с $(i + j)$ чётным имеют чётные значения, а клетки с $(i + j)$ нечётным — нечётные;
- наоборот, клетки с $(i + j)$ чётным имеют нечётные значения, а клетки с $(i + j)$ нечётным — чётные.

Как минимально исправить одну клетку

Если для клетки требуется чётность $p \in \{0, 1\}$, то:

- если текущее число уже имеет чётность p , его менять не нужно;
- иначе достаточно увеличить его на 1.

Больше увеличивать никогда невыгодно, потому что нужно минимизировать суммарную прибавку.

Значит, для фиксированного шахматного шаблона каждая клетка исправляется независимо.

Два кандидата

Построим два варианта ответа.

Первый вариант. Требуемая чётность клетки (i, j) равна

$$(i + j) \bmod 2.$$

Второй вариант. Требуемая чётность клетки (i, j) равна

$$1 - ((i + j) \bmod 2).$$

Для каждого из двух вариантов:

- строим итоговую матрицу;
- считаем суммарное число прибавленных единиц.

Как выбрать ответ

- Если у одного варианта суммарная прибавка меньше, он лучше.
- Если суммы прибавок равны, нужно выбрать лексикографически меньшую матрицу.

Так как матрицы записываются построчно, обычное лексикографическое сравнение двух двумерных массивов даёт именно то, что требуется по условию.

4 Чайные наборы

Условие задачи

Путешествуя по Китаю, Ильнур посещает различные чайные магазины и покупает разные сорта чая. Каждый купленный пакетик он кладёт в чемодан. Если записать сорта чая в том порядке, в котором они лежат в чемодане, получится строка P , состоящая из строчных латинских букв. Каждая буква обозначает сорт чая.

Иногда Ильнур покупает чай не по одному пакетiku, а сразу небольшими наборами. В каждом таком наборе может быть несколько пакетиков чая, причём порядок сортов внутри набора может отличаться.

После возвращения из поездки Ильнур посмотрел на строку P и задумался: какими могли быть размеры наборов, которые он покупал.

Будем говорить, что строка P является *полиграммой*, если существуют целое число $k \geq 2$ и строки $S_1, S_2, \dots, S_k$, такие что

$$P = S_1 S_2 \dots S_k,$$

и все строки $S_1, S_2, \dots, S_k$ являются анаграммами друг друга.

Напомним, что две строки называются анаграммами, если одну из них можно получить из другой перестановкой символов.

Строка S_1 обозначается через S и называется *корнем полиграммы*. Её длина соответствует размеру чайного набора.

Вам дана строка P длины N . Необходимо определить все возможные длины корня полиграммы.

Идея решения

Пусть дана строка P длины N . Требуется найти все такие длины L , что строку можно разбить на

$$k = \frac{N}{L} \geq 2$$

подстрок длины L , и каждая из них является анаграммой первой.

Иными словами, если разбить строку на блоки длины L , то для всех блоков должны совпадать частоты всех букв.

Ключевое наблюдение

Если длина корня равна L , то обязательно выполняются два условия:

- $L < N$, так как блоков должно быть хотя бы два;
- L делит N , иначе разбить строку на равные блоки невозможно.

Следовательно, проверять нужно только делители числа N , меньшие N .

Для фиксированного L разобьём строку на блоки

$$P = B_1 B_2 \dots B_k, \quad |B_i| = L.$$

Тогда длина L подходит тогда и только тогда, когда для всех i строки B_i являются анаграммами друг друга.

Метод 1: префиксные частоты

Построим префиксные суммы по каждой букве: для каждой буквы c и каждой позиции i будем знать, сколько раз буква c встречается в префиксе длины i .

Тогда количество вхождений буквы c на отрезке $[l, r]$ можно получить за $O(1)$:

$$cnt_c(l, r) = pref_c(r) - pref_c(l - 1).$$

Следовательно, частотный вектор любого блока можно получить за

$$O(|\Sigma|),$$

где $|\Sigma|$ — размер алфавита. Сравнение двух блоков также занимает

$$O(|\Sigma|).$$

Для выбранного делителя L :

- находим частотный вектор первого блока;
- для каждого следующего блока длины L сравниваем его частоты с первым;
- если хотя бы для одного блока частоты отличаются, длина L не подходит;
- если все блоки совпали по частотам, длина L подходит.

Метод 2: весовой фильтр

Можно использовать более быстрый способ предварительной проверки. Назначим каждой букве некоторый вес w_c . Тогда вес блока определим как сумму весов его символов.

Если блок содержит символы $x_1, x_2, \dots, x_L$, его вес равен

$$W = w_{x_1} + w_{x_2} + \dots + w_{x_L}.$$

Если два блока являются анаграммами, то множества букв в них совпадают с учётом кратностей, а значит их веса тоже совпадают.

Чтобы быстро вычислять веса блоков, строится префиксный массив

$$pref[i] = w_{P_1} + w_{P_2} + \dots + w_{P_i}.$$

Тогда вес подстроки $[l, r]$ равен

$$pref[r] - pref[l - 1].$$

Для фиксированного L можно:

- посчитать вес первого блока;
- сравнить его с весами всех остальных блоков;
- если какой-то вес отличается, длина L точно не подходит.

Почему это только фильтр

Совпадение весов ещё не гарантирует, что строки являются анаграммами. Например, различные наборы букв могут иметь одинаковую сумму весов.

Следовательно, равенство весов является только необходимым условием.

Поэтому на практике удобно применять следующий подход:

- сначала отбрасывать большинство неподходящих длин по несовпадению весов;
- затем для оставшихся кандидатов выполнять точную проверку частот букв.

Особые случаи

- Если $N = 1$, строка не может быть полиграммой, так как требуется как минимум два блока.
- Если ни один делитель $L < N$ не проходит проверку, необходимо вывести -1 .
- Если подходящих длин несколько, их следует вывести в порядке возрастания.

5 Однотипные города

Условие задачи

Путешествуя по Китаю, Ильнур составил карту перемещений между городами. Каждый город относится к одному из двух типов:

- 0 — исторический город;
- 1 — современный город.

Между некоторыми парами городов существуют односторонние маршруты (например, скоростные поезда или внутренние авиарейсы). Известно, что по этим маршрутам невозможно вернуться в тот же город, то есть сеть маршрутов образует ориентированный ациклический граф.

Ильнур хочет добраться из города A в город B .

Он заметил, что ему быстро надоедают однотипные города, поэтому он не хочет посещать слишком много городов одного типа подряд.

Назовём *серией* максимальный подряд идущий фрагмент маршрута, в котором все города имеют одинаковый тип.

Маршрут считается допустимым, если длина любой серии на нём не превышает L .

Требуется определить количество допустимых маршрутов из города A в город B .

Поскольку число маршрутов может быть большим, выведите ответ по модулю $10^9 + 7$.

Идея решения

Дан ориентированный ациклический граф. У каждой вершины есть тип 0 или 1. Нужно посчитать количество путей из A в B , таких что длина любой максимальной подряд идущей серии вершин одного типа не превосходит L .

Иными словами, если на пути подряд идут несколько вершин одного цвета, то их количество не должно быть больше L .

Ключевая идея

Ограничение зависит не только от текущей вершины, но и от того, сколько вершин того же типа подряд уже набрано в текущей серии.

Поэтому естественно ввести динамику по состояниям:

$$dp[v][k]$$

— количество допустимых путей из A в вершину v , для которых текущая серия однотипных вершин, оканчивающаяся в v , имеет длину ровно k .

Здесь:

- $1 \leq k \leq L$;
- все значения считаются по модулю $10^9 + 7$.

Переходы

Пусть есть ребро $u \rightarrow v$.

- Если $c_u \neq c_v$, то при переходе в v начинается новая серия длины 1. Тогда все способы попасть в u можно перенести в состояние

$$dp[v][1].$$

- Если $c_u = c_v$, то текущая серия продолжается. Если в u серия имела длину k , то в v она станет длины $k + 1$. Такой переход допустим только при

$$k + 1 \leq L.$$

То есть переходы имеют вид:

- если $c_u \neq c_v$, то для всех k

$$dp[v][1] += dp[u][k];$$

- если $c_u = c_v$, то для всех $k < L$

$$dp[v][k + 1] += dp[u][k].$$

Начальное состояние:

$$dp[A][1] = 1,$$

поскольку путь из одной вершины A состоит из одной серии длины 1.

Ответ:

$$\sum_{k=1}^L dp[B][k].$$

Почему DAG сильно помогает

Так как граф ацикличен, его вершины можно обработать в топологическом порядке.

Когда мы рассматриваем вершину u , все значения динамики для неё уже полностью посчитаны, и можно безопасно распространять их по исходящим рёбрам.

Именно поэтому здесь получается обычная динамика по графу без рекурсии с мемоизацией и без риска заиклиться.

Замечания по подзадачам

- При $n \leq 20$ или $n, m \leq 2000$ можно было бы даже делать более прямолинейную динамику или поиск по путям.
- При $L = 1$ разрешены только пути, в которых типы соседних вершин всегда различны. В этом случае остаётся обычный подсчёт путей в DAG, но только по рёбрам между вершинами разных типов.
- Если граф является цепочкой, задача сводится к обычной динамике по массиву.

6 Два путешественника

Условие задачи

Путешествуя по Китаю, Ильнур и Глеб решили составить маршрут, чтобы посетить как можно больше интересных мест. Они изучили карту и выбрали n городов, которые хотят увидеть: от шумных мегаполисов до небольших древних поселений, расположенных вдоль восточного побережья страны.

Для каждого города известны его координаты (x_i, y_i) на карте. Города уже упорядочены по неубыванию координаты x , то есть

$$x_1 \leq x_2 \leq \dots \leq x_n.$$

Путешественники начинают своё путешествие вместе в городе 1. Изначально оба находятся в этом городе, поэтому город 1 считается посещённым ими обоими.

После старта каждый из путешественников может независимо от другого перемещаться между городами. Разрешается переходить только из города с меньшим номером в город с большим номером.

Каждый город с номером от 2 до n должен быть посещён хотя бы одним из путешественников.

Если путешественник перемещается из города i в город j , то длина такого перехода равна

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Суммарная длина маршрута равна сумме длин всех переходов, совершённых Ильнуром и Глебом.

Путешественники не обязаны заканчивать маршрут в одном и том же городе, и каждый из них может посетить любое количество городов.

Необходимо определить минимальную возможную суммарную длину маршрутов, при условии, что каждый город с номером от 2 до n будет посещён хотя бы одним из них.

Идея решения

Ключевая идея решения состоит в том, что для продолжения маршрута важно знать только текущие позиции двух путешественников. Все ранее посещённые города уже не влияют на стоимость будущих переходов.

Поэтому состояние решения можно описывать только двумя вершинами — городами, в которых сейчас находятся путешественники.

Ключевое наблюдение

Так как разрешено двигаться только из города с меньшим номером в город с большим номером, любой маршрут каждого путешественника представляет собой возрастающую по номерам последовательность городов.

Рассмотрим некоторый момент, когда уже покрыты все города с номерами от 1 до k , а путешественники находятся в городах i и j . Тогда обязательно

$$\max(i, j) = k.$$

Действительно, если бы оба номера были меньше k , то город k ещё не был бы посещён. Если же какой-то номер был бы больше k , то тогда уже был бы посещён и этот больший город, что невозможно.

Значит, состояние можно описывать парой последних посещённых городов:

$$dp[i][j]$$

— минимальная суммарная длина маршрутов после того, как посещены все города с номерами от 1 до $\max(i, j)$, а путешественники находятся в городах i и j .

Что означает состояние

Поскольку все города от 1 до $\max(i, j)$ уже покрыты, следующий город, который ещё не посещён, имеет номер

$$k = \max(i, j) + 1.$$

Именно этот город и нужно добавить следующим шагом. Посетить его может либо первый путешественник, либо второй.

Переходы

Пусть

$$k = \max(i, j) + 1.$$

Если $k \leq n$, то возможны два перехода:

- первый путешественник идёт из i в k , тогда новое состояние:

$$dp[k][j] = \min(dp[k][j], dp[i][j] + dist(i, k));$$

- второй путешественник идёт из j в k , тогда новое состояние:

$$dp[i][k] = \min(dp[i][k], dp[i][j] + dist(j, k)).$$

Здесь

$$dist(a, b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}.$$

Начальное состояние:

$$dp[1][1] = 0,$$

потому что оба путешественника стартуют в городе 1, и никаких переходов ещё не сделано.

Почему этого достаточно

В любой допустимой конфигурации после покрытия городов $1, 2, \dots, k$ важно только следующее:

- где сейчас находится первый путешественник;
- где сейчас находится второй путешественник.

Вся более ранняя история уже не влияет на стоимость дальнейших переходов, потому что:

- все будущие города имеют номера больше k ;
- стоимость следующего шага зависит только от текущего положения того путешественника, который делает этот шаг.

Поэтому состояние $dp[i][j]$ содержит всю необходимую информацию для продолжения построения оптимального решения.

Почему каждый следующий город можно добавлять сразу

Так как города можно посещать только в порядке возрастания номеров, после того как уже покрыты все города до k , нельзя “перепрыгнуть” через город $k + 1$ и сначала посетить какой-то более поздний город.

Следовательно, на каждом шаге следующий ещё не покрытый город определяется однозначно:

$$k + 1.$$

Это и делает динамику квадратичной, а не экспоненциальной. Очень приятный момент: геометрия тут нужна только для расстояний, а вся комбинаторика сидит в порядке индексов.

Финальный ответ

Когда покрыты все города, должно выполняться

$$\max(i, j) = n.$$

Значит, ответом будет минимум по всем состояниям, в которых хотя бы один из путешественников находится в городе n :

$$\min(dp[n][i], dp[i][n])$$

по всем $1 \leq i \leq n$.

Так как таблица симметрична по смыслу, на практике можно просто взять

$$\min_{1 \leq i \leq n} dp[i][n].$$

или

$$\min_{1 \leq i \leq n} dp[n][i],$$

в зависимости от реализации.

Порядок вычисления

Состояния удобно обрабатывать в порядке возрастания

$$\max(i, j).$$

Например, можно перебирать все пары (i, j) , для которых значение уже определено, и делать переходы в город

$$\max(i, j) + 1.$$

Так как в каждом переходе максимум увеличивается ровно на 1, все зависимости направлены вперёд, и циклов нет.

Замечания по подзадачам

- При $n \leq 10$ или $n \leq 34$ можно было бы даже рассматривать перебор с мемоизацией, но общая динамика уже работает достаточно быстро.
- В случае $y_i = 0$ все города лежат на одной прямой, и расстояние между городами равно $|x_i - x_j|$, однако сама структура решения не меняется.