Оглавление

MEX массива

Дан массив a длины n. Разрешено заменять любой элемент на текущее значение mex массива. Требуется найти максимально возможное значение mex, которое можно получить.

Можно показать, что mex массива не может превышать n. mex = n достижимо при массиве $0, 1, \ldots, n-1$ после последовательного применения операций ко всем элементам.

Максимальное произведение

Дан массив целых чисел a длины n. Требуется найти максимальное произведение двух элементов.

Отсортируем массив по возрастанию. Рассмотрим два кандидата на максимальное произведение:

1. Произведение двух наибольших элементов: $a_{n-1} \cdot a_{n-2}$
2. Произведение двух наименьших элементов: $a_0 \cdot a_1$

Максимальное произведение равно максимуму из этих двух вариантов.

Палиндром

Дан набор больших латинских букв. Требуется составить палиндром максимальной длины, а среди таких — лексикографически наименьший.

Для получения лексикографически наименьшего палиндрома строим левую половину, перебирая буквы от A до Z и добавляя половину их количества. Если есть буква с нечётной частотой, выбираем наименьшую такую букву для центра. Правая половина является обратной к левой.

Если нечётных букв нет, центр отсутствует, и палиндром состоит из левой половины и её отражения.

Подготовка к экзаменам - легкая версия

У студента есть n выученных билетов, i-й билет имеет сложность a_i. Сессия состоит из n экзаменов, которые сдаются по порядку. Для сдачи i-го экзамена требуется b_i билетов. Использованный билет "забывается"и не может быть использован повторно. Сессия заканчивается, когда студент сдал все экзамены или на очередной экзамен не хватает билетов.

Перед началом студент выбирает пороговую сложность x. Все билеты со сложностью меньше x становятся непригодными (их нельзя использовать). Результат сессии вычисляется как $x \cdot k$, где k — количество успешно сданных экзаменов. Нужно выбрать x так, чтобы максимизировать результат. Здесь $a_i \leq 100$.

Поскольку все $a_i \leq 100$, можно перебрать все возможные x от 100. Пусть cnt — количество билетов со сложностью $\geq x$. Экзамены сдаются по порядку, на i-й экзамен требуется b_i билетов. Процесс сдачи останавливается, когда для очередного экзамена недостаточно оставшихся билетов.

Таким образом, для каждого x находим cnt — количество билетов $\geq x$, после чего проходим по экзаменам от 1 до n, вычитая b_i из cnt, пока $cnt \geq b_i$. Параллельно можно поддерживать k — количество успешно пройденных экзаменов, обновляя ответ: $ans = \max(ans, x \cdot k)$.

Подготовка к экзаменам - сложная версия

У студента есть n выученных билетов, i-й билет имеет сложность a_i. Сессия состоит из n экзаменов, которые сдаются по порядку. Для сдачи i-го экзамена требуется b_i билетов. Использованный билет "забывается"и не может быть использован повторно. Сессия заканчивается, когда студент сдал все экзамены или на очередной экзамен не хватает билетов.

Перед началом студент выбирает пороговую сложность x. Все билеты со сложностью меньше x становятся непригодными (их нельзя использовать). Результат сессии вычисляется как $x \cdot k$, где k — количество успешно сданных экзаменов. Нужно выбрать x так, чтобы максимизировать результат. Здесь $a_i \leq 10^9$.

Отсортируем билеты по возрастанию сложности. Для фиксированного x подходят билеты с $a_i \geq x$, то есть суффикс массива a. Пусть выбрано cnt билетов. Чтобы сдать k экзаменов, необходимо:

$$\sum_{i=1}^{k} b_i \leq cnt$$

Заметим, что x может принимать только значения, равные некоторому a_i, так как увеличение x ($a_{i-1} < x < a_i$) до a_i не уменьшает количество доступных билетов, но увеличивают ответ. Поэтому достаточно рассмотреть все a_i в качестве кандидатов.

Для каждого i (индекс в отсортированном массиве a) количество доступных билетов равно $n - i$. Построим префиксные суммы pb от массива b: $pb[k] = \sum_{j=1}^{k} b_j$.

Для фиксированного i найдём максимальное k, такое что $pb[k] \leq n - i$. Это можно сделать бинарным поиском по k, так как pb монотонно возрастает. Тогда результат для данного i равен $a_i \cdot k$. Перебираем все i от 0 до $n - 1$ и максимизируем ответ.

Удаление подстроки с сохранением подпоследовательности - легкая версия

Даны строки s и t, причём t является подпоследовательностью s. Требуется удалить из s непрерывную подстроку максимальной длины так, чтобы t осталась подпоследовательностью полученной строки. Здесь $|s|, |t| <= 200$.

Пусть $n = |s|$, $m = |t|$. Рассмотрим все возможные отрезки $[l, r]$ (в 0-индексации), которые мы хотим удалить. Для каждого такого отрезка проверим, остаётся ли t подпоследовательностью после удаления и обновляем ответ: $ans = max(ans, r - l + 1)$, если проверка успешна.

Проверка выполняется двумя указателями и проходом по исходной строке s: идём по символам s от 0 до $n-1$, пропуская символы, попадающие в удаляемый отрезок $[l, r]$. Для остальных символов проверяем, совпадают ли они с текущим символом t. Если удаётся пройти все символы t, то условие выполнено

Перебираем все пары (l, r) с $0 \leq l \leq r < n$. Для каждой вычисляем длину удаляемой подстроки $r - l + 1$ и обновляем ответ, если проверка успешна.

Удаление подстроки с сохранением подпоследовательности - сложная версия

Даны строки s и t, причём t является подпоследовательностью s. Требуется удалить из s непрерывную подстроку максимальной длины так, чтобы t осталась подпоследовательностью полученной строки. Здесь $|s|, |t| <= 200000$.

Применим бинарный поиск по длине удаляемой подстроки. Для фиксированной длины len необходимо проверить, существует ли отрезок $[l, r]$ длины len, после удаления которого t остаётся подпоследовательностью. Для эффективной проверки предварительно построим два массива:

- $pre[i]$ — позиция в s, на которой заканчивается префикс $t[0..i]$ при поиске слева направо (индекс последнего символа префикса)
- $suff[i]$ — позиция в s, на которой начинается суффикс $t[i..m-1]$ при поиске справа налево (индекс первого символа суффикса)

Тогда после удаления отрезка $[l, r]$ строка t останется подпоследовательностью, если найдётся такой индекс i в t, что:

$$pre[i] < l \quad \text{и} \quad suff[i+1] > r$$

Это означает, что префикс $t[0..i]$ полностью помещается до удаляемого отрезка, а суффикс $t[i+1..m-1]$ — после него. Условие можно переписать как: $r-l+1 \geq len$ и существует i с $suff[i+1]-pre[i]-1 \geq len$.

Для проверки фиксированной длины len достаточно проверить, существует ли такой i, что $suff[i+1]-pre[i]-1 \geq len$.

Минимальное суммарное расстояние - легкая версия

На плоскости заданы n точек с целочисленными координатами. Требуется найти количество точек с целыми координатами, минимизирующих сумму манхэттенских расстояний до всех заданных точек. Здесь координаты точек находятся в диапазоне $[0, 500]$.

Так как координаты ограничены ($0 \leq x_i, y_i \leq 500$), можно перебрать все возможные точки-кандидаты с целыми координатами в этом же диапазоне. Для каждой точки (i, j) вычислим сумму расстояний:

$$S(i,j) = \sum_{p=1}^{n} (|i - x_p| + |j - y_p|)$$

В процессе перебора поддерживаем минимальную найденную сумму и счётчик точек, на которых этот минимум достигается: если текущая сумма меньше глобального минимума — обновляем минимум и сбрасываем счётчик в 1, а если равна минимуму — увеличиваем счётчик на 1. После перебора всех 501×501 точек выводим полученное количество.

Минимальное суммарное расстояние - сложная версия

На плоскости заданы n точек с целочисленными координатами. Требуется найти количество точек с целыми координатами, минимизирующих сумму манхэттенских расстояний до всех заданных точек. Здесь координаты точек находятся в диапазоне $[0, 10^9]$.

Заметим, что манхэттенское расстояние разделяется по координатам:

$$\sum_{i=1}^{n} (|x - x_i| + |y - y_i|) = \sum_{i=1}^{n} |x - x_i| + \sum_{i=1}^{n} |y - y_i|$$

Следовательно, минимизация по x и y происходит независимо. Для одномерного случая известно, что множество точек, минимизирующих сумму расстояний до заданных точек, представляет собой отрезок между двумя медианами:

- Если n нечётно, медиана единственна — это $\lfloor n/2 \rfloor$-й элемент в отсортированном массиве.
- Если n чётно, любая точка между $\lfloor (n-1)/2 \rfloor$-м и $\lfloor n/2 \rfloor$-м элементами дает одинаковую минимальную сумму.

Таким образом, количество оптимальных x и y координат равно:

$$\text{count}_x = x_{\lfloor n/2 \rfloor} - x_{\lfloor (n-1)/2 \rfloor} + 1$$

$$\text{count}_y = y_{\lfloor n/2 \rfloor} - y_{\lfloor (n-1)/2 \rfloor} + 1$$

Общее количество оптимальных точек на плоскости равно произведению $\text{count}_x \cdot \text{count}_y$.

Максимальная пустая полка

На числовой прямой заданы n отрезков $[l_i, r_i]$, символизирующих занятые книги. Требуется найти максимальную длину непрерывной свободной области, не покрытой ни одним отрезком.

Отсортируем отрезки по левой координате. Затем объединим пересекающиеся или касающиеся отрезки, поддерживая текущий объединённый отрезок $[cur\_l, cur\_r]$.

Для каждого следующего отрезка $[l, r]$:

- Если $l \leq cur\_r$ (отрезки пересекаются или касаются), расширяем правую границу: $cur\_r = \max(cur\_r, r)$
- Иначе между ними образуется промежуток: $l - cur\_r$. Обновляем максимальную длину промежутка и начинаем новый объединённый отрезок $[l, r]$

После обработки всех отрезков выводим максимальную найденную длину промежутка. Если промежутков не было, ответ равен 0.