Оглавление

Локальные минимумы

Дан массив a длины n. Необходимо посчитать количество элементов, являющихся локальными минимумами.

Элемент a_i является локальным минимумом, если он строго меньше всех своих соседей. Рассмотрим три случая:

1. Для $i = 0$ (первый элемент): проверяем условие $a_0 < a_1$ (если $n > 1$).
2. Для $i = n - 1$ (последний элемент): проверяем условие $a_{n-1} < a_{n-2}$ (если $n > 1$).
3. Для $1 \le i \le n - 2$: проверяем $a_i < a_{i-1}$ и $a_i < a_{i+1}$.

Алгоритм линейно проходит по массиву, для каждого элемента выполняет $O(1)$ проверок и подсчитывает количество подходящих элементов.

Левый минимум

Дан массив a длины n. Для каждого индекса i необходимо найти индекс минимального элемента на префиксе $[0, i]$. Если таких элементов несколько, выбрать наименьший индекс.

Введем переменную min для хранения текущего минимального значения на префиксе и массив ans, где $ans[i]$ — искомый индекс для позиции i.

Базовый случай: $ans[0] = 0$, $min = a_0$.

Для каждого следующего i от 1 до $n - 1$:

- Если $a_i < min$: обновляем $min = a_i$, устанавливаем $ans[i] = i$
- Иначе: $ans[i] = ans[i - 1]$ (минимальный элемент остался тот же, его индекс — предыдущий ответ)

Таким образом, за один проход по массиву вычисляются все ответы.

Конфеты и дети

Учительница имеет a конфет и n детей. Необходимо раздать все конфеты так, чтобы каждый ребенок получил целое положительное число конфет и все дети получили разное количество.

Минимальное суммарное количество конфет, необходимое для раздачи n детям с различными положительными количествами, достигается при распределении $1, 2, 3, \ldots, n$ конфет. Сумма такой последовательности равна:

$$S_{\min} = \frac{n(n+1)}{2}$$

Если $a \ge S_{\min}$, то конфет хватит (можно выдать минимальные порции, а оставшиеся конфеты добавить, например, последнему ребенку, сохраняя условие различности). Если $a < S_{\min}$, то конфет не хватит, так как даже минимально возможная сумма превышает имеющееся количество.

Для каждого теста достаточно вычислить $S_{\min}$ и сравнить с a.

+ или -

Дана строка S длины n, состоящая из символов '+' и '-'. Требуется для каждого запроса $[l, r]$ найти разницу между количеством плюсов и минусов на этом отрезке.

Закодируем каждый символ числовым значением val: '+1' для '+' и '-1' для '-' и построим массив префиксных сумм p длины $n + 1$, где $p[0] = 0$, а $p[i] = p[i - 1] + val(s[i - 1])$ для i от 1 до n.

Тогда сумма на отрезке $[l, r]$ (в 0-индексации) равна: $\text{sum}(l, r) = p[r+1] - p[l]$. Эта сумма и есть искомая разница.

Игра с камнями

Дана куча из n камней. За ход можно взять 1, 3 или 4 камня. Проигрывает тот, кто не может сделать ход. Необходимо определить победителя при оптимальной игре.

Обозначим $dp[i]$ как состояние игры при i камнях: 1 — выигрышная позиция для текущего игрока, 0 — проигрышная. Базовые случаи:

- $dp[0] = 0$ (камней нет, текущий игрок проигрывает)
- $dp[1] = 1$ (можно взять 1 камень и выиграть)
- $dp[2] = 0$ (можно взять только 1 камень, остаётся 1 камень — выигрышная позиция для противника)
- $dp[3] = 1$ (можно взять 3 камня и выиграть сразу)

Переход: позиция i является выигрышной, если существует такой ход $x \in \{1, 3, 4\}$, что $i \geq x$ и $dp[i-x] = 0$ (противник попадает в проигрышную позицию). Иначе позиция проигрышная.

Вычисляем $dp[i]$ для всех i от 4 до максимального n по формуле:

$$dp[i] = \max(dp[i-1] == 0,\ dp[i-3] == 0,\ dp[i-4] == 0)$$

Для каждого запроса выводим "First"при $dp[n] = 1$, иначе "Second".

Ладьи и препятствия

Дана доска $n \times n$. На ней расположены ладьи и препятствия. Ладья бьёт все клетки по горизонтали и вертикали до первого препятствия или другой ладьи. Необходимо подсчитать количество клеток, не находящихся под атакой ни одной ладьи.

Обозначим состояния клеток:

- 0 — пустая клетка
- 1 — ладья
- -1 — препятствие

Для каждой клетки (i, j) проверяем, находится ли она под атакой. Для этого просматриваем четыре направления:

1. Влево: $j \to 0$, останавливаемся при встрече препятствия или ладьи
2. Вправо: $j \to n-1$, останавливаемся при встрече препятствия или ладьи
3. Вверх: $i \to 0$, останавливаемся при встрече препятствия или ладьи
4. Вниз: $i \to n-1$, останавливаемся при встрече препятствия или ладьи

Если в любом из направлений до препятствия встречается ладья, клетка находится под атакой.

После подсчёта всех пустых клеток, не находящихся под атакой, вычитаем количество препятствий p, так как они не являются свободными клетками и не должны учитываться в ответе.

Минимальный максимум

Требуется построить массив длины n из неотрицательных целых чисел, сумма которого кратна k, а максимальный элемент минимален.

Пусть x — искомое минимальное значение максимального элемента. Тогда каждый элемент массива не превосходит x, а сумма всех элементов не превосходит $n \cdot x$. Чтобы сумма делилась на k, необходимо $n \cdot x \geq k$ (иначе максимальная возможная сумма меньше k).

При фиксированном x мы можем выбрать элементы так, чтобы сумма была в точности равна k (или $k, 2k, 3k, \ldots$, но минимальная достигается при k). Для этого нужно $n \cdot x \geq k$ и, кроме того, k должно быть достижимо как сумма n чисел от 0 до x.

Если $n > k$, то можно взять массив из k единиц и $n - k$ нулей. Тогда сумма равна k (делится на k), а максимальный элемент равен 1.

Если $n \leq k$, то минимальный максимум достигается при равномерном распределении k по n элементам. Наименьшее целое x, при котором $n \cdot x \geq k$, равно $\lceil k/n \rceil$. Именно это значение и будет ответом.

Таким образом:

$$\text{Ответ} = \begin{cases} 1, & n > k \\ \left\lceil \dfrac{k}{n} \right\rceil, & n \leq k \end{cases}$$

Жадность

Дан массив A длины n и число x. Требуется найти максимальное количество элементов, сумма которых не превосходит x.

Чтобы взять как можно больше элементов с ограничением на сумму, необходимо выбирать наименьшие элементы. Отсортируем массив по возрастанию и будем накапливать сумму элементов в порядке возрастания. Как только добавление очередного элемента делает сумму больше x, останавливаемся. Количество взятых элементов до этого момента и будет ответом. Если сумма всех элементов не превосходит x, ответ равен n.

Освещение сцены

Дана сцена $n \times m$ с актёрами (1) и пустыми клетками (0). Прожектор можно установить в любую пустую клетку и направить вверх, вниз, влево или вправо. Позиция считается хорошей, если в направлении луча есть хотя бы один актёр. Необходимо подсчитать общее количество таких позиций (учитывая направление).

Построим двумерный префиксный массив p размера $(n+1) \times (m+1)$, где $p[i][j]$ — количество актёров в прямоугольнике от $(0,0)$ до $(i-1, j-1)$:

$$p[i][j] = p[i-1][j] + p[i][j-1] - p[i-1][j-1] + v[i-1][j-1]$$

Для каждой пустой клетки (i, j) проверяем четыре направления:

1. **Влево**: клетки $(i, 0..j-1)$
   Количество актёров: $p[i+1][j] - p[i][j]$
2. **Вверх**: клетки $(0..i-1, j)$
   Количество актёров: $p[i][j+1] - p[i][j]$
3. **Вправо**: клетки $(i, j+1..m-1)$
   Количество актёров: $p[i+1][m] - p[i][m] - p[i+1][j+1] + p[i][j+1]$
4. **Вниз**: клетки $(i+1..n-1, j)$
   Количество актёров: $p[n][j+1] - p[n][j] - p[i+1][j+1] + p[i+1][j]$

Если в соответствующем направлении количество актёров больше нуля, увеличиваем счётчик на 1.

Альтернативный отбор

Даны две шеренги спортсменов длины n с результативностями a_i и b_i. Необходимо выбрать подмножество спортсменов в порядке возрастания номеров, чередуя шеренги (нельзя брать двух подряд из одной шеренги), максимизируя суммарную результативность.

Рассмотрим динамическое программирование. Обозначим:

- $dpA[i]$ — максимальная суммарная результативность при рассмотрении первых i позиций, если последний взятый спортсмен из первой шеренги (или спортсменов ещё не брали)
- $dpB[i]$ — максимальная суммарная результативность при рассмотрении первых i позиций, если последний взятый спортсмен из второй шеренги

Базовые случаи:

- $dpA[0] = a_0$ (берём первого из первой шеренги)
- $dpB[0] = b_0$ (берём первого из второй шеренги)

Переходы для $i \geq 1$:

- $dpA[i] = \max(dpB[i-1] + a_i,\ dpA[i-1])$
  - Либо берём a_i после спортсмена из второй шеренги
  - Либо пропускаем a_i, сохраняя предыдущее состояние
- $dpB[i] = \max(dpA[i-1] + b_i,\ dpB[i-1])$
  - Либо берём b_i после спортсмена из первой шеренги
  - Либо пропускаем b_i, сохраняя предыдущее состояние

Ответ для всего массива: $\max(dpA[n-1], dpB[n-1])$.

Подотрезок с ограниченным разнообразием

Дан массив a длины n. Требуется найти максимальную длину непрерывного подотрезка, содержащего не более k различных элементов.

Применим бинарный поиск по длине подотрезка. Пусть len — текущая проверяемая длина. Необходимо определить, существует ли подотрезок длины len, содержащий не более k различных элементов.

Для фиксированной длины len используем скользящее окно: проходим по всем левым границам i от 0 до $n - len$. Для каждого окна $[i, i + len - 1]$ подсчитываем количество различных элементов. Если найдено окно с $\leq k$ различными элементами, длина len достижима.

Для эффективного подсчёта различных элементов при сдвиге окна поддерживаем словарь частот: изначально считаем частоты для первого окна $[0, len - 1]$; при сдвиге окна вправо: удаляем $a[i]$, добавляем $a[i + len]$, обновляем словарь и количество уникальных элементов.

Если длина len достижима, ищем в правой половине, иначе — в левой. Бинарный поиск выполняется по длине от 1 до n.

Разделение массива

Дан массив a длины n из положительных целых чисел. Необходимо определить, можно ли разбить его на две группы так, чтобы суммы элементов в группах отличались не более чем на 1.

Обозначим $S = \sum_{i=1}^{n} a_i$ — общая сумма элементов. Условие "отличаются не более чем на 1"означает, что суммы групп должны быть $\lfloor S/2 \rfloor$ и $\lceil S/2 \rceil$. Таким образом, достаточно проверить, можно ли набрать сумму $\lfloor S/2 \rfloor$, а это классическая задача о рюкзаке.