Оглавление

Корень четвертой степени

Дано целое неотрицательное число x. Требуется найти $\lfloor \sqrt[4]{x} \rfloor$ — наибольшее целое y, такое что $y^4 \le x$.

Корень четвёртой степени можно вычислить как квадратный корень от квадратного корня: $\sqrt[4]{x} = \sqrt{\sqrt{x}}$. Для каждого запроса вычисляем $\sqrt{x}$ и от полученного значения снова берём квадратный корень, округляя вниз до целого числа.

Подъем по лестнице

Мальчик поднимается по лестнице из n ступенек, начиная с нулевой. За один шаг можно перешагнуть на 1, 2 или 3 ступеньки вверх, но нельзя делать два шага подряд одинаковой длины. Требуется найти количество способов подняться на n-ю ступеньку по модулю $10^9 + 7$.

Рассмотрим динамическое программирование, где состояние $dp[i][j]$ — количество способов достичь ступеньки i, сделав последним шагом длины j ($j = 0$ соответствует шагу на 1 ступеньку, $j = 1$ — на 2 ступеньки, $j = 2$ — на 3 ступеньки). Тогда переходы учитывают запрет на повтор длины:

- $dp[i][0] = dp[i-1][1] + dp[i-1][2]$ (шаг длины 1, предыдущий — длины 2 или 3)
- $dp[i][1] = dp[i-2][0] + dp[i-2][2]$ (шаг длины 2, предыдущий — длины 1 или 3)
- $dp[i][2] = dp[i-3][0] + dp[i-3][1]$ (шаг длины 3, предыдущий — длины 1 или 2)

Базовые случаи задаются для первых трёх ступенек с учётом ограничений. Ответом для ступеньки n будет сумма $dp[n-1][0] + dp[n-1][1] + dp[n-1][2]$ по модулю $10^9 + 7$.

Белочка и орехи

Белочка раскладывает n орехов по дуплам так, чтобы в каждом следующем дупле было на один орех больше, чем в предыдущем. Требуется найти количество различных способов такого разложения (различные длины последовательности и начальные значения).

Пусть последовательность имеет длину k и начинается с числа a. Тогда сумма орехов равна:

$$a + (a+1) + \cdots + (a+k-1) = k \cdot a + \frac{k(k-1)}{2} = n$$

Отсюда:

$$k \cdot a = n - \frac{k(k-1)}{2}$$

Для существования целого неотрицательного a необходимо, чтобы $n - \frac{k(k-1)}{2} \ge 0$ и делилось на k. Перебираем k от 1 до тех пор, пока $\frac{k(k-1)}{2} < n$, и проверяем условие делимости. Каждое подходящее k даёт ровно один способ (значение a определяется однозначно).

Заметим, что k не может превышать $\sqrt{2n}$, поэтому перебор выполняется за $O(\sqrt{n})$ на запрос.

Шах королю

На шахматной доске 8×8 расположены белый король (W), чёрный король (B) и белый ферзь (Q). Требуется определить, находится ли чёрный король под шахом белого ферзя.

Ферзь атакует по горизонтали, вертикали и обеим диагоналям. Необходимо проверить четыре направления:

1. **По горизонтали:** чёрный король и ферзь на одной строке ($x_2 = x_3$). Если белый король не находится между ними по столбцу (y_1 не лежит между y_2 и y_3), то объявляется шах.
2. **По вертикали:** чёрный король и ферзь в одном столбце ($y_2 = y_3$). Если белый король не находится между ними по строке (x_1 не лежит между x_2 и x_3), то шах.
3. **По главной диагонали:** $x_2 - x_3 = y_2 - y_3$. Если белый король не стоит на той же диагонали между ними ($x_1 - x_2 = y_1 - y_2$ и x_1 между x_2 и x_3), то шах.
4. **По побочной диагонали:** $x_2 - x_3 = y_3 - y_2$. Если белый король не стоит на той же диагонали между ними ($x_1 - x_2 = y_2 - y_1$ и x_1 между x_2 и x_3), то шах.

Если ни одно из условий не выполнено, шах отсутствует.

Мат королю

На шахматной доске 8×8 расположены белый король (W), чёрный король (B) и белый ферзь (Q). Требуется определить, стоит ли мат чёрному королю.

Мат чёрному королю означает, что:

1. Чёрный король находится под шахом от белого ферзя.
2. У чёрного короля нет ни одного допустимого хода (все соседние клетки либо заняты, либо находятся под боем белого ферзя или белого короля, либо выходят за пределы доски).

Проверим, находится ли чёрный король под шахом от ферзя (из предыдущей задачи). Если шаха нет, выводим NO MATE, иначе перебираем все 8 соседних клеток чёрного короля (с учётом границ доски). Для каждой клетки проверяем не занята ли она белым королём или ферзём; не находится ли она под боем ферзя; не находится ли она под боем белого короля (клетки рядом с белым королём).

Если найдена хотя бы одна клетка, куда чёрный король может пойти, то мата нет — выводим NO MATE. Если все соседние клетки недоступны, выводим MATE. Здесь важно не забыть о том, что черный король может съесть белого ферзя.

Интервалы и точки - легкая версия

Заданы n отрезков $[l_i, r_i]$ и m точек x_j. Для каждой точки требуется определить количество отрезков, которые её покрывают. Здесь $n, m \leq 2000$.

Для каждой точки x_j перебираем все отрезки и проверяем условие $l_i \leq x_j \leq r_i$. Если условие выполняется, увеличиваем счётчик для данной точки. После проверки всех отрезков выводим полученное значение.

Интервалы и точки - сложная версия

Заданы n отрезков $[l_i, r_i]$ и m точек x_j. Для каждой точки требуется определить количество отрезков, которые её покрывают. Здесь $n, m \leq 2 \cdot 10^5$.

Используем метод сканирующей прямой. Создадим список событий и для каждого отрезка $[l_i, r_i]$ добавим два события:

- В точке l_i происходит событие "начало отрезка которое увеличивает количество активных отрезков на 1.
- В точке r_i происходит событие "конец отрезка которое уменьшает количество активных отрезков на 1.

Для каждой точки x_j добавим событие "запрос"с координатой x_j и индексом j для записи ответа.

Отсортируем все события по координате. При равенстве координат порядок обработки может быть важен: сначала обрабатываются события начала отрезков, затем запросы, а затем события концов отрезков.

Будем поддерживать переменную cnt — текущее количество активных отрезков. Пройдём по отсортированным событиям:

- Если событие — начало отрезка, увеличиваем cnt на 1.
- Если событие — конец отрезка, уменьшаем cnt на 1.
- Если событие — запрос, записываем cnt как ответ для этой точки.

Обработка диапазонов запросов - легкая версия

Имеется массив a длины n, изначально заполненный нулями, и m запросов прибавления (l_i, r_i, x_i). Затем поступает q итоговых запросов (L_j, R_j), каждый из которых требует независимо выполнить все запросы прибавления с номерами от L_j до R_j включительно и вывести полученный массив. Здесь $n, m, q \leq 500$.

В данной версии задачи ограничения малы, поэтому можно решать "в лоб"для каждого итогового запроса. Для очередного запроса (L_j, R_j) создаём массив ans из n нулей и проходим по всем запросам прибавления с индексами от L_j до R_j. Для каждого такого запроса явно прибавляем x_i ко всем элементам ans на отрезке $[l_i, r_i]$. После обработки всех запросов выводим полученный массив.

Обработка диапазонов запросов - средняя версия

Имеется массив a длины n, изначально заполненный нулями, и m запросов прибавления (l_i, r_i, x_i). Затем поступает q итоговых запросов (L_j, R_j), каждый из которых требует независимо выполнить все запросы прибавления с номерами от L_j до R_j включительно и вывести полученный массив. Здесь $n, m, q \leq 10^5$, но выполняются дополнительные ограничения: $m \cdot q \leq 10^6$ и $n \cdot q \leq 10^6$.

Для решения этой версии заметим, что операцию прибавления x_i ко всем элементам ans на отрезке $[l_i, r_i]$ можно выполнять с помощью разностного массива. После обработки всех запросов выводим полученный массив, составив префиксные суммы на разностном массиве.

Обработка диапазонов запросов - сложная версия

Имеется массив a длины n, изначально заполненный нулями, и m запросов прибавления (l_i, r_i, x_i). Затем поступает q итоговых запросов (L_j, R_j), каждый из которых требует независимо выполнить все запросы прибавления с номерами от L_j до R_j включительно и вывести полученный массив. Здесь $n, m, q \le 10^5$, с дополнительным ограничением $n \cdot q \le 10^6$.

Применим технику корневой декомпозиции (sqrt-декомпозицию) по запросам прибавления. Разобьём все m запросов на блоки размера $B \approx \sqrt{m}$. Для каждого блока предподсчитаем его суммарное влияние на массив a, то есть какой массив получится, если применить все запросы данного блока. Для этого используем разностный массив: для каждого запроса блока выполняем $diff[l_i] += x_i$ и $diff[r_i + 1] -= x_i$, затем вычисляем префиксные суммы, получая результирующий массив для блока. Накопим эти результаты в массиве $pref$: $pref[k][pos]$ — значение в позиции pos после применения всех запросов из первых k блоков. Предподсчёт выполняется за $O\left(\frac{m}{B} \cdot n\right) = O(\sqrt{m} \cdot n)$.

Для каждого итогового запроса (L, R) переведём границы в 0-индексацию и определим номера блоков bl и br, содержащих L и R. Возможны два случая:

- Если $bl = br$, то все запросы находятся в одном блоке. Обрабатываем их напрямую: создаём разностный массив, применяем все запросы с L по R, вычисляем префиксные суммы и получаем ответ за $O(B + n)$.
- Иначе запрос покрывает несколько блоков. Тогда:
  1. Берём предподсчитанное влияние всех полных блоков между $bl + 1$ и $br - 1$ — это разность $pref[br] - pref[bl + 1]$ (поэлементно), выполняемая за $O(n)$.
  2. Для левого неполного блока (запросы с L до конца блока bl) и правого неполного блока (запросы с начала блока br до R) применяем прямое добавление через разностный массив, аналогично первому случаю.
  3. Суммируем полученные массивы.

Асимптотика: предподсчёт $O(\sqrt{m} \cdot n)$, обработка одного запроса $O(B + n)$.

Марта и делители - легкая версия

Дан массив p длины n, состоящий из различных целых чисел от 1 до n. Требуется ответить на m запросов (l, r): количество пар (q, w) таких, что $l \le q, w \le r$ и p_q делит p_w. Здесь $n, m \le 5000$.

Идея решения заключается в предподсчёте для каждой возможной пары индексов (i, j), является ли p_i делителем p_j. Так как $n \le 5000$, можно построить матрицу смежности размером $n \times n$, где $div[i][j] = 1$, если p_i делит p_j, и 0 иначе. Для каждой пары (i, j) проверка делимости выполняется за $O(1)$.

Тогда ответ на запрос (l, r) — это сумма всех $div[i][j]$ для i, j в диапазоне $[l, r]$. Эту сумму можно находить напрямую за $O((r - l + 1)^2)$, но это будет слишком медленно. Вместо этого предподсчитаем двумерные префиксные суммы на построенной матрице: $pref[i][j] = \sum_{x=1}^{i} \sum_{y=1}^{j} div[x][y]$. Тогда ответ на запрос (l, r) вычисляется за $O(1)$ как:

$$ans = pref[r][r] - pref[l-1][r] - pref[r][l-1] + pref[l-1][l-1]$$

Асимптотика: предподсчёт $O(n^2)$, ответ на запрос $O(1)$.

Марта и делители - сложная версия

Дан массив p длины n, состоящий из различных целых чисел от 1 до n. Требуется ответить на m запросов (l, r): количество пар (q, w) таких, что $l \le q, w \le r$ и p_q делит p_w. Здесь $n, m \le 2 \cdot 10^5$.

Ключевая идея заключается в переходе от проверки делимости элементов к работе с их позициями. Для каждой пары (i, j) такой, что p_i делит p_j, можно рассматривать отрезок $[\min(i, j), \max(i, j)]$, который эта пара покрывает. Тогда задача сводится к подсчёту количества таких отрезков, полностью лежащих внутри запроса $[l, r]$.

Предварительно построим все пары (i, j), где p_i делит p_j. Для каждого значения x от 1 до n переберём его кратные $y = k \cdot x$ и, зная позиции $pos[x]$ и $pos[y]$ (благодаря уникальности элементов), добавим отрезок $[\min(pos[x], pos[y]), \max(pos[x], pos[y])]$. Количество таких отрезков равно сумме гармонического ряда $O(n \log n)$.

Будем обрабатывать запросы в порядке увеличения правой границы r. Для каждого r добавляем все отрезки с правым концом r в структуру данных, отмечая их левую границу. Используем дерево Фенвика для поддержания количества активных отрезков, левая граница которых $\ge l$. Тогда ответ на запрос (l, r) — это сумма на отрезке $[l, r]$ в дереве Фенвика.

Асимптотика: построение отрезков $O(n \log n)$, обработка запросов $O((n + m) \log n)$.