

Задача А. ВСОШ по Информатике

В этой задаче ключевое наблюдение в том, что каждый удлинитель даёт c_i разъёмов, но если мы хотим включить его в уже собранную цепочку, то один разъём этого удлинителя неизбежно тратится на подключение к предыдущему источнику питания. Значит, каждый удлинитель, кроме самого первого, даёт не c_i , а только $c_i - 1$ полезных разъёмов для ноутбуков или следующих удлинителей.

Первый удлинитель подключается прямо в единственную розетку, поэтому он сохраняет все свои c_i разъёмов. Если использовать все N удлинителей, общее число доступных разъёмов будет равно $c_1 + (c_2 - 1) + (c_3 - 1) + \dots + (c_N - 1)$, что после упрощения даёт формулу $\sum_{i=1}^N c_i - (N - 1) = \sum_{i=1}^N c_i - N + 1$. Именно столько ноутбуков можно подключить максимально. Поэтому авторское решение просто считывает все значения c_i , находит их сумму sum , а затем выводит $sum - n + 1$. Алгоритм работает за $\mathcal{O}(N)$, потому что каждый удлинитель обрабатывается ровно один раз.

Задача В. Отсортировать заказы

Главная идея одного из возможных решений — строить ответ цифра за цифрой, как будто мы спускаемся по лексикографическому дереву чисел. Если рассматривать все числа от 1 до n как строки, то лексикографический порядок соответствует обходу префиксного дерева: сначала идут числа с префиксом 1, потом с префиксом 10, 100 и так далее. Поэтому достаточно поддерживать текущий префикс t и на каждом шаге пытаться дописать к нему очередную цифру так, чтобы среди чисел от 1 до n , начинающихся с нового префикса, существовало хотя бы одно число, делящееся на k . Для этого для кандидата $q = 10t + i$ рассматриваются все числа с таким префиксом: это отрезки $[q, q]$, $[10q, 10q + 9]$, $[100q, 100q + 99]$, $\dots$, пока они не выходят за пределы n .

Необходимо проверять, есть ли в очередном таком диапазоне хотя бы одно кратное k . Это можно понять по трём условиям: либо сам левый конец диапазона уже кратен k , либо длина диапазона не меньше k , тогда по принципу остатков кратное точно найдётся, либо при переходе от q к правому концу r остаток по модулю k "перепрыгивает" через ноль что проверяется условием $r \bmod k < q \bmod k$. Если хотя бы для одной цифры такой префикс перспективен, её можно безопасно добавить в ответ, потому что лексикографически это минимальный возможный следующий шаг. Когда в какой-то момент сам префикс t становится кратным k , поиск заканчивается: это и есть минимальное в лексикографическом порядке подходящее число. По времени для одного запроса решение работает очень быстро: глубина ответа не больше 19, на каждом уровне перебирается не более 10 цифр, и для каждой цифры проверяется не более 19 возможных длин, так что сложность составляет фактически константное время на запрос.

Задача С. Суммы — это просто!

Пусть $S = 1 + 2 + \dots + N = \frac{N(N+1)}{2}$ — сумма всех чисел от 1 до N . Если S уже простое, то ответ очевиден: все числа можно поместить в одну группу, и это минимально возможное число групп. Если S составное, то одной группы недостаточно, и дальше идея решения состоит в том, чтобы разбить множество чисел на минимальное число групп, используя свойства простых сумм. Для чётного S необходимо искать такое число i , что и i , и $S - i$ простые. Тогда достаточно выделить число i в отдельную группу, а все остальные числа положить во вторую: суммы этих двух групп будут равны i и $S - i$, то есть обе простые.

Здесь используется идея, близкая к гипотезе Гольдбаха: чётное число представляется как сумма двух простых. Для нечётного S надо проверить, является ли $S - 2$ простым. Если да, то достаточно выделить число 2 в отдельную группу, а остальные поместить во вторую, потому что суммы групп будут 2 и $S - 2$, обе простые. Если это не получилось, из общей суммы можно вычесть 3, и если $S - 3$ простое, то аналогично достаточно выделить число 3. Если и это не сработало, остаётся разбить числа на три группы: отдельно число 3, отдельно некоторое число i , а все остальные — в третью группу. Тогда суммы групп равны 3, i и $S - 3 - i$, и необходимо искать такой i , чтобы числа i и $S - 3 - i$ были простыми. В результате число групп всегда получается минимальным и не превышает трёх.

Задача D. Преобразуйте массив

В этой задаче разрешённая операция — обмен соседних элементов, а минимальное число таких

обменов для преобразования одного массива в другой равно числу инверсий в некоторой перестановке позиций. Поэтому сначала нужно понять, можно ли вообще получить массив b из массива a : для каждого значения количество его вхождений в обоих массивах должно совпадать.

Для каждого числа можно хранить списки его позиций в первом и втором массивах и попарно сопоставлять эти вхождения: если элемент стоит на позиции i в массиве a , а соответствующее ему такое же вхождение в массиве b должно оказаться на позиции j , то записывать $perm[i] = j$. Если для какого-то значения числа вхождений не совпали, сразу выводить -1 .

После этого задача сводится к следующей: есть перестановка $perm$, и нужно за минимальное число соседних обменов превратить её в возрастающую последовательность $1, 2, \dots, n$. Известно, что это число в точности равно количеству инверсий, то есть числу пар (i, j) , где $i < j$, но $perm[i] > perm[j]$. Поэтому в решении лучше использовать сортировку слиянием: при слиянии двух половин, если элемент из правой части оказывается меньше текущего элемента из левой, то он образует инверсии со всеми ещё не взятыми элементами левой части, и к ответу прибавляется их количество.

Задача Е. Таблица

В этой задаче нужно минимальным числом одинаковых фигур покрыть всю таблицу, причём фигуры можно накладывать друг на друга и даже частично выводить за границы поля, поэтому задача сводится не к точной укладке, а именно к покрытию.

В качестве одного из вариантов решения можно использовать динамику по профилю. Состояние описывается функцией $f(j, a)$, где j — номер текущей строки, а a — битовая маска для трёх подряд идущих строк, начиная с j : в ней отмечено, какие клетки уже покрыты. Такого запаса в 3 строки достаточно, потому что сама фигура живёт внутри квадрата 3×3 , значит, влияния дальше этих трёх строк нет. Если вся текущая строка уже покрыта, то можно перейти к следующей строке: $f(j, a) = f(j + 1, a \gg w)$, где сдвиг убирает обработанную строку. Иначе находится первая непокрытая клетка в текущей строке, и дальше перебираются все способы поставить фигуру так, чтобы она покрывала именно эту клетку.

Для каждого возможного положения заранее строится маска покрытия, после чего новое состояние получается как $b = a \vee \text{mask}$, и переход имеет вид $f(j, a) = \min(f(j, b) + 1)$. Если ни одна постановка фигуры не покрывает выбранную клетку, то добавляется фиктивный переход $f(j, a \vee o) + 1$, что эквивалентно покрытию этой клетки отдельной фигурой "в лоб"; это гарантирует существование ответа. Так как ширина и высота не превосходят 9, состояние по сути хранит не более $3w \leq 27$ бит, а число реально достижимых состояний оказывается достаточно небольшим. Для ускорения можно сделать хеширование уже посчитанных состояний, чтобы не пересчитывать одинаковые подзадачи.