

## Разбор задачи «Космический корабль»

**Решение первой подзадачи (20 баллов):** переберем врага, который может быть боссом и рассмотрим сумму оставшихся, которую будем считать каждый раз заново. Решение будет работать за  $O(n^2)$ .

**Решение второй подзадачи (30 баллов):** в данной подгруппе нет отрицательных чисел, значит босс — это враг с наибольшей силой. Найти максимум в массиве можно за  $O(n)$ .

**Полное решение (100 баллов):** заметим, что если сила босса равна  $x$ , то сумма сил всех врагов равна  $2x$ . Следовательно, чтобы найти силу босса, необходимо посчитать суммарную силу всех врагов и разделить ее на 2.

Чтобы восстановить порядок достаточно найти позицию босса и, например, поменять его местами с врагом на последней позиции.

## Разбор задачи «Рейнджеры в автобусе»

Будем обрабатывать пассажиров по очереди и для каждого определять, кем он мог бы быть. Сразу заметим, что розовый рейнджер может быть куда угодно, поэтому каждый пассажир мог бы быть розовым.

**Решение первой подзадачи (25 баллов):** для каждого места будем помнить, свободно ли оно (массив `bool` длины  $2n$ ). Проверим, мог бы очередной пассажир быть красным рейнджером. Для этого найдём место, на которое сел бы красный. Переберём циклом ряд от 1 до  $n$ , если попался ряд, в котором есть свободное место, выбираем левое, если оно свободно, или правое, если нет — это и есть место, куда сел бы красный. Поскольку он выбирает место однозначно, мы можем определить, мог бы этот пассажир быть красным — нужно проверить, что он сел именно на это место. Таким же образом узнаем, куда сели бы синий, жёлтый и чёрный — разница будет в порядке перебора рядов (от 1 до  $n$  или наоборот) и в порядке выбора места в ряду (сначала левое или правое). После обработки пассажира отметим его место как занятое.

Решение работает за  $O(nk)$  и использует  $O(n)$  памяти.

**Решение второй подзадачи (50 баллов):** на самом деле предыдущее решение работает за  $O(k^2)$  времени. Дело в том, что каждый цикл по рядам останавливается, когда находит ряд со свободным местом — а количество занятых рядов не более  $k/2$ . Однако оно использует  $O(n)$  памяти, а это слишком много для второй подзадачи. Есть два способа решения этой проблемы:

- Вместо массива использовать `std::set`, в котором хранить занятые места.
- Хранить только  $k/2 + 1$  первых рядов и столько же последних.

Тогда решение будет использовать  $O(k)$  памяти.

**Полное решение (100 баллов):** вспомним, что каждый из циклов останавливается, когда находит ряд, в котором есть свободное место. Это означает, что нужно быстро находить первый и последний незанятые ряды. Будем поддерживать эти значения — *first* и *last*. Изначально *first* = 1, *last* =  $n$ . После каждой итерации цикла обновим эти значения. Будем увеличивать *first* на 1, пока ряд *first* занят, затем уменьшать *last* на 1, пока ряд *last* занят. Занятых рядов не может оказаться больше  $k/2$ , поэтому *first* и *last* сделают  $O(k)$  шагов. Таким образом, решение работает за  $O(k)$ .

## Разбор задачи «Мегазорды»

**Решение первой подзадачи (15 баллов):** переберем зорды трех различных цветов и проверим условия. Для проверки достаточно сравнить первую цифру красной модели с последней цифрой зеленой, а также последнюю цифру красной модели с первой цифрой синей. Решение будет работать за  $O(n^3)$ .

**Решение второй подзадачи (25 баллов):** заметим, что количество различных номеров моделей мало. Для каждого цвета посчитаем количество моделей с фиксированным номером. Нам достаточно перебрать номера моделей для всех трех рейнджеров и прибавить к ответу произведение их количества.  $O(90^3 + n)$

**Решение третьей подзадачи (31 балл):** предподсчитаем массив  $R[a][b]$ , где элемент с номерами  $a$  и  $b$  равен количеству моделей у красного рейнджера, у которых первая цифра равна  $a$  и

последняя равна  $b$ . Теперь переберем номер модели, который взяли зеленый и синий рейнджеры. Зафиксируется первая и последняя цифра номера красной модели. Используя массив  $R$ , за  $O(1)$  узнаем количество подходящих моделей красного рейнджера.

**Решение четвертой подзадачи (46 баллов):** в этой подзадаче, существуют модели с одинаковыми номерами. Заметим, что решение для третьей группы посчитает способы, в которых могут повторяться номера моделей. Для начала учтем в рассмотрении вариантов, что номера моделей зеленого и синего рейнджера не должны совпадать.

Теперь для получения верного ответа вычтем случаи, в которых модели красного рейнджера совпадают либо с зеленым, либо с синим. Посчитаем количество моделей, которые совпадают у красного и синего, а также красного и зеленого рейнджеров и вычтем эти числа из ответа, если они были посчитаны в нашем решении. Это можно сделать при помощи структуры *map* или сортировки.

**Полное решение (100 баллов):** посчитаем массивы  $G[a]$  и  $B[b]$ . Элемент  $a$  в массиве  $G$  равен количеству моделей у зеленого рейнджера, у которых последняя цифра номера равна  $a$ . Элемент  $b$  в массиве  $B$  равен количеству моделей синего рейнджера, у которых первая цифра равна  $b$ . Теперь переберем первую и последнюю цифру номера модели красного рейнджера. К ответу прибавим  $G[a] \cdot R[a][b] \cdot B[b]$ . Не забудем вычесть варианты, в которых у какой-то пары совпадают номера моделей. Для этого воспользуемся идеей из четвертой подгруппы. Также заметим, что вариант, где все три рейнджера имеют одинаковый номер модели будет вычитаться 3 раза, поэтому необходимо прибавить это количество увеличенное в 2 раза.

## Разбор задачи «Объединенная армия»

**Частичные решения (0-100 баллов):** Можно заметить, что случаев, когда в армии могут находиться не только солдаты из Глиняного патруля всего 15. Некоторые из них довольно легко рассмотреть на листочке. Например, для  $x = 0$  и  $y = 2$  для  $k > 1$  и наименьший, и наибольший ответ будут равны двум, а расстановка будет выглядеть примерно так:

$$\begin{array}{c} 000 \dots 001 \\ 100 \dots 000 \end{array}$$

Более того, разбором случаев можно набрать до 100 баллов включительно.

**Полное решение (100 баллов):** Будем решать задачу динамическим программированием по профилю.

$dp[i][mask\_prev][mask\_cur]$  — минимальное/максимальное количество солдат из Зедд патруля, которые могут быть в расстановке из первых  $i$  столбцов,  $mask\_cur$  — маска  $i$ -го столбца, а  $mask\_prev$  —  $(i - 1)$ -го.

Затем переберем маску для  $(i + 1)$ -го столбца и проверим выполняются ли ограничения на соседей, для солдат из  $i$ -го столбца, так как теперь известны все их соседи.

Для первого и последнего столбца ограничения нужно проверять отдельно, потому что у них нет одного из соседей.

База:  $dp[2][mask\_prev][mask\_cur]$  равно количеству единиц в  $mask\_prev$  и  $mask\_cur$ , если  $mask\_prev$  может стоять слева от  $mask\_cur$ .

Переход:  $dp[i + 1][mask\_cur][mask\_next] = dp[i][mask\_prev][mask\_cur] + ones(mask\_next)$ , где  $ones(x)$  — количество единиц в маске  $x$ .

Ответ будет минимумом/максимумом среди всех  $dp[k][mask\_prev][mask\_cur]$ , таких что  $mask\_cur$  может находиться справа от  $mask\_prev$ .

Так как размерность маски константа и равна двум, данное решение будет работать за  $O(n)$ .

## Разбор задачи «Тюрьма для Зедда»

Заметим, что противоположные грани параллелепипеда должны быть равны, поэтому можно поворотом перевести каждый прямоугольник к виду  $a_i \leq b_i$ , и искать уже три прямоугольника. Далее следует понять, что если длины сторон, исходящих из одной вершины, равны  $A$ ,  $B$  и  $C$ , то параллелепипед должен быть составлен из трех прямоугольников со сторонами  $A \times B$ ,  $A \times C$ ,  $B \times C$ , не забыв, учесть два аспекта: для каждой грани должно быть хотя бы по два соответствующих прямоугольника, кроме того, если два или три размера равны друг другу, некоторые грани совпадают и количество необходимых прямоугольников возрастает до 4 и 6, соответственно.

**Решение первой подзадачи (27 баллов):** можно перебрать три прямоугольника, которые войдут в ответ, и явно проверить условие, описанное ранее. Данное решение работает за  $O(n^3)$ .

**Решение второй подзадачи (63 балла):** можно перебрать один прямоугольник. Пусть он имеет размер  $A \times B$  — у нас уже известно две размерности параллелепипеда, осталось лишь перебрать третью размерность параллелепипеда  $C$  и проверить, есть ли прямоугольники размера  $A \times C$  и  $B \times C$ . На этом моменте можно допустить ошибку, когда  $A = B$ ,  $A = C$  или  $B = C$ , попытавшись использовать один прямоугольник дважды, хотя его можно взять не больше одного раза. Для простоты можно считать что  $A$ ,  $B$  и  $C$  различны, а случаи, когда какая-то пара совпадает, разобрать отдельно, это несложно сделать за  $O(n)$ . Данное решение работает за  $O(n^2)$ .

**Полное решение (100 баллов):** будет удобно перейти к графовой интерпретации задачи. Наличие прямоугольников со сторонами  $A \times B$ ,  $A \times C$ ,  $B \times C$  эквивалентно нахождению цикла длины 3 в графе, построенном на числах, отвечающих за размерности параллелепипеда — прямоугольник  $A \times B$  отвечает в таком графе за ребро  $AB$ .

Решение за  $O(n^2/64)$ : перебираем одно из ребер  $uv$  треугольника, и смотрим на пересечение множества соседей вершин  $u$  и  $v$ . Это можно сделать пересечением bitset-ов за  $n^2/64$ , и перебрать третью вершину треугольника как все единички в пересечении bitset-ов.

Решение за  $O(n\sqrt{n})$ : можно упорядочить все вершины по возрастанию их степени, количества смежных ребер. Переберем вершину с минимальной степенью, входящую в треугольник. После этого можно перебрать ее соседа — вершину, которая будет второй по степени среди вершин нашего цикла. После этого переберем третью вершину, как соседа второй, и проверим, есть ли ребро между первой и третьей вершиной.

Можно показать, что если перебирать вершины именно в таком порядке увеличения степени на графе с  $m$  ребрами, будет рассмотрено не более  $m\sqrt{m}$  троек чисел. Для этого можно разбить отсортированный массив степеней  $deg$  на две части: префикс, в котором сумма не больше  $\sqrt{n}$ , и суффикс. Когда первая вершина из первой части массива, мы переберем  $\sqrt{n}$  кандидатов на вторую вершину и  $n$  кандидатов на третью, что дает  $n\sqrt{n}$ . Во второй же части не больше  $\sqrt{n}$  вершин, поэтому кандидатов на третью вершину для них будет не больше  $\sqrt{n}$ , что снова приводит нас к тому, что треугольников в графе не больше  $O(n\sqrt{n})$ .