# Разбор задачи «Bananas Packing»

Для решения задачи требуется использовать жадную стратегию: пока можно, нужно собирать набор с минимальным $d$. Для этого можно пойти по массиву слева направо, и для каждого $d$ проверять сколько наборов можно собрать с таким $d$ — это $\min(a_d, a_{d+1}, \ldots, a_{d+6})$. Прибавляем это число к ответу и вычитаем его же из $a_d, a_{d+1}, \ldots, a_{d+6}$. А сколько бананов осталось — сумма всех $a_i$ в конце.

# Разбор задачи «Permutations»

Для каждого столбца найдем число, которого ему не хватает, чтобы стать перестановкой, либо определим, что в столбце есть повтор, и перестановкой он уже стать не может. Это можно сделать, считая в дополнительном массиве $col\_cnt[i]$ длины $n$, сколько раз число $i$ встречается в столбце.

Заметим, что если нескольким столбцам не хватает одного и того же числа, то после добавления перестановки только один из этих столбцов может стать перестановкой. Тогда в остальные столбцы мы можем дописать любое число, и это не ухудшит ответ. То есть если есть $k$ столбцов, которым не хватает числа $x$, то все перестановки, где $x$ стоит на одном из этих $k$ мест, дадут нам плюс один столбец-перестановку.

Посчитаем в массив $cnt[i]$ длины $n$, скольким столбцам не хватает числа $i$.

- Если $cnt[x] == 0$, значит нам неважно, где будет стоять число $x$ в дописанной перестановке. Посчитаем количество нулей в массиве $cnt$ в переменную $zeroes$.
- Если $cnt[x] \neq 0$, значит у нас есть $cnt[x]$ вариантов поставить число $x$ в перестановку и получить один дополнительный столбец-перестановку. Посчитаем произведение ненулевых значений $cnt[x]$ в переменную $ways$.

Тогда наибольшее достижимое число столбцов перестановок — это $n - zeroes$, а количество возможных перестановок $ways \cdot zeroes!$.

# Разбор задачи «Equation»

Пусть $l = r$, тогда неравенство $l \leqslant b + c \leqslant r$ примет вид $b + c = l$. Теперь надо вспомнить теорему Виета $b = -x_1 - x_2$, а $c = x_1 \cdot x_2$.

Преобразуем равенство:

$$x_1 \cdot x_2 - x_1 - x_2 = l$$

$$x_1 \cdot (x_2 - 1) - x_2 = l + 1 - 1$$

$$x_1 \cdot (x_2 - 1) - x_2 + 1 = l + 1$$

$$x_1 \cdot (x_2 - 1) - (x_2 - 1) = l + 1$$

$$(x_1 - 1) \cdot (x_2 - 1) = l + 1$$

Получается, что $x_1 - 1$ и $x_2 - 1$ должны быть делителями числа $l + 1$, а наша задача заключается в том, чтобы найти количество способов разложить $l + 1$ как произведение двух целых чисел. Пар корней бесконечно, когда $l = -1$, так как при $x_1 = 1$ уравнение примет вид $1 \cdot x_2 - 1 - x_2 = -1$, а значит, равенство верно для всех $x_2 \in \mathbb{Z}$. Для остальных $l$ ответ конечен.

Поймем, чему равно число способов представить $l + 1 = x_1 \cdot x_2$. В качестве $x_1$ можно взять любой делитель $l + 1$ (включая отрицательный), однако, разные способы выбрать $x_1$ могут давать одинаковые пары. Все пары делителей $l + 1$ разбиваются на пары $(x_1, x_2)$ и $(x_2, x_1)$ кроме случая, когда эти две пары совпадают. Такое случается, когда $x_1^2 = l + 1$, то есть, $l + 1$ является полным квадратом. Таким образом, тогда ответом для числа $l$ является количество делителей числа $l + 1$, отдельно добавим единицу в случае полного квадрата.

Для подсчета ответа на отрезке, используем модифицированное решето Эратосфена, чтобы найти делители всех чисел на отрезке $[l, r]$. Воспользуемся тем фактом, что количество делителей числа равно произведению степеней простых чисел, увеличенных на один. Сначала найдем все простые числа до $10^6$. Будем перебирать простые числа и идти с шагом $prime_i$ на отрезке $[\lceil \frac{l}{prime_i} \rceil \cdot prime_i, \lfloor \frac{r}{prime_i} \rfloor \cdot prime_i]$ и делить текущее число, пока оно делится на $prime_i$. Домножаем количество делителей для этого числа на то, сколько мы раз поделили на $prime_i$ плюс один. Первое и последнее число на отрезке $[l, r]$, которые делятся на $prime_i$, равны $\lceil \frac{l}{prime_i} \rceil \cdot prime_i$ и $\lfloor \frac{r}{prime_i} \rfloor \cdot prime_i$ соответственно. После того, как мы обработали все простые до $10^6$, осталось обработать все оставшиеся простые, для каждого числа такое осталось максимум одно.

# Разбор задачи «Fantastic Three»

Зафиксируем $j$. Переберем, какой префикс битов чисел $(a_i \oplus a_j)$ и $(a_j \oplus a_k)$ совпадает. Заметим, что в таком случае префикс такой же длины совпадает и у чисел $a_i$ и $a_k$. Следующий после совпадающего префикса бит в числе $(a_i \oplus a_j)$ должен быть меньше, чем в числе $(a_j \oplus a_k)$. Значит, если в числе $a_j$ на позиции этого бита написан 0, то в $a_i$ должен быть 0, а в $a_k$ — 1. Если же в числе $a_j$ на этой позиции 1, то в $a_i$ — 1, а в $a_k$ — 0.

Пусть мы перебираем $j$ в порядке возрастания, и поддерживаем два множества чисел: $P = \{a_1, a_2, \ldots, a_{j-1}\}$, $S = \{a_{j+1}, a_{j+2}, \ldots, a_n\}$. Тогда нам нужно научиться обрабатывать следующие операции:

1. Добавить/удалить элемент в множестве $P$.
2. Добавить/удалить элемент в множестве $S$.
3. Узнать количество пар $p \in P$, $s \in S$, таких что префиксы $p$ и $s$ до $k$-го бита не включительно совпадают, а в $k$-м бите $p$ больше/меньше, чем $s$.

На третью операцию мы будем отвечать просто возвращая значение, записанное в массиве. А при обработке первых двух операций, будем поддерживать значения в массиве актуальными. Будем хранить элементы и из $P$, и из $S$ в одном битовом боре (биты в боре идут от корня вниз от старших битов к младшим). Если бор уже построен, мы можем для каждой вершины $v$ посчитать количество чисел из $P$, которые заканчиваются в поддереве $v$, и аналогично количество чисел из $S$, которые заканчиваются в поддереве $v$. А чтобы заполнить массив ответов, нужно для каждой вершины $v$ вычислить произведение количества чисел из $P$, которые заканчиваются в поддереве левого сына $v$ и количества чисел из $S$, которые заканчиваются в поддереве правого сына $v$. Либо наоборот — у левого сына числа из $S$ и у правого сына числа из $P$. Когда мы производим добавление/удаление одного числа, посчитанные значения изменяются только для вершин по которым мы пройдем, добавляя число. Пересчитаем для них эти значения, и используя корректно посчитанные значения, обновим посчитанные ответы в массиве. Итого, одну операцию удаления/добавления мы обрабатываем за $O(\log C)$, где $C$ — максимальное значение $a_i$. А все решение работает за $O(n \cdot \log C)$.

# Разбор задачи «Reconstructing Pairs»

Задача состояла в том, чтобы «ориентировать» каждую из входных $n$ пар таким образом, чтобы первые $n$ элементов образовывали набор $A$, а вторые $n$ элементов образовывали набор $B$. За $O(2^n)$ можно перебрать ориентацию каждой пары и проверить наборы первых и вторых элементов напрямую.

Сформулируем эту задачу в терминах графов. Возьмем все значения, которые встречаются во входных данных, в качестве вершин графа, а пары будут обозначать ребра, соединяющие два числа (возможны петли и кратные ребра). Изначально неориентированные ребра нужно превратить в ориентированные таким образом, чтобы исходящая степень каждой вершины была равна количеству вхождений этого числа в набор $A$, а входящая – в набор $B$. В частности, если степень вершины в этом графе не равна суммарному числу вхождений числа в $A$ и $B$, то ответа не существует. Сделаем эту проверку в начале решения.

Заметим, что если мы сделали проверку на суммарную степень, а также ориентировали ребра таким образом, чтобы удовлетворить ограничения на исходящую степень (на вхождения в набор $A$), то и ограничения на входящую степень (на набор $B$) автоматически удовлетворены. Итого, текущая версия задачи звучит следующим образом: для каждого ребра нужно выбрать один из двух концов таким образом, чтобы каждая вершина была выбрана нужное число раз (для определенности, выберем для этого набор $A$).

Такую задачу можно свести к нахождению совершенного сочетания в следующем двудольном графе: вершины в левой доле соответствуют элементам $A$ (скопированным столько раз, сколько соответствующее число встречается в $A$), а вершины в правой доле соответствуют входным парам и соединены с двумя числами, равным элементам этой пары. В этом графе существует совершенное паросочетание тогда и только тогда, когда у задачи есть решение. В таком графе $O(n)$ вершин и $O(n^2)$ ребер, однако алгоритм Куна работает достаточно быстро, чтобы пройти предпоследнюю подзадачу.

Если объединить вершины левой доли, соответствующие одинаковым числам, можно получить граф, в котором $O(n)$ ребер, однако вершины левой доли теперь имеют определенную кратность: со сколькими вершинами в правой доле их нужно соединить. Для решения такой версии задачи можно модифицировать алгоритм Куна, который с оптимизациями работает достаточно быстро. Иначе можно алгоритм Диница нахождения максимального потока. В таком графе применима теорема Карзанова, поэтому алгоритм Диница доказуемо работает за $O(n\sqrt{n})$.