Разбор задачи «Выстрел в голову»

Заметим, что если $m \cdot b \leqslant a$, то можно полностью перезарядить пистолет за $m \cdot b$ секунд, а не за a. Сделаем $a = \min(a, m \cdot b)$. После того, как мы так сделали, нам не имеет смысл m раз заряжать один патрон. Тогда нам надо либо сделать $\lceil \frac{n}{m} \rceil$ полных перезарядок, либо $\lfloor \frac{n}{m} \rfloor$ полных перезарядок и дозарядить $n \bmod m$ одиночных патронов. Тогда ответ это минимум из этих двух значений.

Разбор задачи «Сжатие изображения»

Рассмотрим какое-нибудь сжатие данного изображения. Пусть при этом сжатии изображение было разбито на прямоугольники $a \times b$. Тогда очевидно, что a является делителем n, а b является делителем m.

Научимся проверять, что в некотором прямоугольнике разбиения все пиксели имеют один цвет за время $\mathcal{O}(1)$. Заменим символы «.» на нули, а «X» — на единицы. Теперь нам нужно проверить, что сумма в прямоугольнике равна либо 0, либо площади этого прямоугольника. Чтобы быстро считать сумму в прямоугольнике, посчитаем аналог сумм на префиксах для прямоугольника. Обозначим за $pref[i][j]$ сумму всех значений, у которых первый индекс не превосходит i, а второй индекс не превосходит j. Тогда сумма в прямоугольнике с углами (x_1, y_1) и (x_2, y_2) равна $pref[x2][y2] - pref[x1-1][y2] - pref[x2][y1-1] + pref[x2-1][y2-1]$. С помощью этого довольно классического приема мы научились проверять, что все символы в прямоугольнике одинаковые.

Теперь переберем в качестве значений a и b все делители чисел n и m соответственно, и для каждой пары (a, b) проверим, что такие значения подходят за время $\mathcal{O}(\frac{n}{a} \cdot \frac{m}{b})$. Среди всех подходящих значений выберем значение, для которого $\frac{nm}{ab}$ минимально.

Оценим время работы получившегося алгоритма. Заметим, что время работы алгоритма равняется сумме по всем парам (a, b), где a — делитель n, а b — делитель m, величин $\frac{nm}{ab}$. Очевидно, эта сумма не превосходит суммы тех же самых величин по всем парам (a, b), где $1 \leqslant a \leqslant n, 1 \leqslant b \leqslant m$ (так как эта сумма содержит все слагаемые исходной суммы). Таким образом, время работы алгоритма не превышает величины $\sum\limits_{a=1}^{n} \sum\limits_{b=1}^{m} \frac{a}{n} \frac{b}{m} = \sum\limits_{a=1}^{n} \frac{a}{n} \cdot (\sum\limits_{b=1}^{m} \frac{b}{m})$.

Довольно известный факт, что $\sum\limits_{i=1}^{n} \frac{1}{i} = \mathcal{O}(\log n)$. Умножая это равенство на n, получаем, что $\sum\limits_{i=1}^{n} \frac{n}{i} = \mathcal{O}(n \log n)$.

Значит, наша искомая сумма равна $\sum\limits_{a=1}^{n} \frac{a}{n} \cdot \mathcal{O}(m \log m)$. Применив ещё раз этот же факт, получим, что время работы алгоритма составляет $\mathcal{O}(nm \log n \log m)$.

Разбор задачи «Суперагентское блюдо»

Построим граф зависимостей ингредиентов. Теперь для каждого ингридиента найдем минимальную стоимость, за которую его можно получить — тогда ответ будет равен сумме минимальных стоимостей для n ингредиентов, из которых состоит рецепт суперагентского блюда.

Минимальную стоимость получения ингредиента можно найти, запустив обход в глубину из этой вершины — зная минимальные стоимости получения всех ингредиентов из рецепта, мы знаем минимальную стоимость получения выбранного ингредиента.

Разбор задачи «Секретный код»

Заметим, что в результате обмена двух равных букв, строка не меняется. Если же мы меняем две разные буквы, то мы получаем строки, которые отличаются от исходной в фиксированных двух позициях, причем при любом таком обмене мы получим попарно разные строки, так как у нас поменяется разное подмножество позиций исходной строки (иначе это был один и тот же обмен). Значит ответ равен числу пар позиций, на которых находятся разные символы, плюс, возможно, один, если у нас есть пара одинаковых символов(только, так мы можем получить исходную строку). И то, и другое, легко считается за $O(|s|)$.

Разбор задачи «Игра в дженгу»

Заметим, что из ряда больше нельзя вынимать бруски в одном из двух случаев:

- Вынут брусок стоящий по центру.
- Вынуты два крайних бруска.

Назовем ряд «активным», если из него еще можно вынуть хотя бы один брусок. Изначально в башне $n-2$ активных ряда.

Рассмотрим победную стратегию второго игрока, если n четно. Будем рассматривать соседние ряды башни, кроме верхнего и нижнего (каждый ряд будет иметь пару, так как n четно). Если первый игрок вынимает брусок, то второй делает аналогичный ход в его парном ряду.

В результате таких действий, как только первый игрок совершит ход, который уменьшит количество активных рядов башни, второй игрок своим следующим ходом уменьшит количество активных рядов еще на единицу. Тогда активных рядов станет на два меньше, чем было. В случае, же если ход первого игрока не уменьшает количество активных рядов, то и у второго игрока будет ход, чтобы не уменьшить это количество.

Таким образом четность количества активных рядов никак не зависит от ходов первого игрока, и в его очередь оно всегда четно. В некоторый момент активных рядов станет ноль и первый игрок не сможет походить.

При нечетной начальной высоте башни, первый игрок может вынуть центральный брусок в одном из рядов, после чего пользоваться стратегией описанной выше ($n-1$ — четно), чтобы выиграть.

Разбор задачи «Кодовый замок»

Зафиксируем позицию j в массивах. Заметим, что если одно число встречается на позициях j, $m-j$ суммарно более двух раз, ответ на задачу `No`. Так как при разворотах любых массивов, это число по прежнему будет содержаться как минимум два раза в строке j или $m-j$.

Пусть A_i — i-ый массив во входных данных, A_{ij} — элемент, находящийся в i-ой строке и j-ом столбце таблицы. Далее, если ответ может существовать, построим граф из n вершин: для всех массивов A_i и A_k проведем ребро из вершины i в вершину k с цветом 0, если существует j, что $A_{ij} = A_{kj}$ и ребро с цветом 1, если $A_{ij} = A_{k(m-j)}$.

Теперь покрасим граф в два цвета — 0 (нужно переворачивать) и 1 (не нужно). При этом, если текущая вершина v покрашена в цвет 0, существует ребро цвета 0 из v в u, то нужно покрасить u в цвет 1, то есть развернуть один из массивов относительно другого. А для всех ребер из v в u цвета 1, наоборот, вершину u нужно покрасить в тот же цвет, что и v, то есть не разворачивать один массив относительно другого. При этом, если мы сейчас хотим покрасить вершину u, а она уже покрашена в другой цвет, то возникает конфликт и решения не существует, то есть ответ `No`. Это возникает вследствие того, что могут существовать проблемы для массивов A_i, A_k, для которых $A_{ij} = A_{i(m-j)}$ и $A_{ip} = A_{kp}$.

Массивы, номера которых соответствуют вершинам цвета 0, следует развернуть. Эти номера и будут являться ответом.

Разбор задачи «План шпионской сети»

Если есть хотя бы одна пара совпадающих точек, то одно множество — одна из этих точек, второе — все остальные точки.

Иначе построим выпуклую оболочку исходных точек. Если есть хотя бы одна точка, невходящая в выпуклую оболочку, то одно множество — эта точка, второе — все остальные точки, Если такой нет, то в одно множество включим две любые несоседние точки выпуклой оболочки, а в другое все остальные.

Разбор задачи «Минер»

Рассмотрим любой остов графа. Если можно его взорвать, то можно взорвать весь граф, так как остов содержит все вершины. Заметим, что остов можно полностью взорвать, если он не состоит из одной вершины, следующим алгоритмом:

- Подвесим остов за любую вершину.

- Пока в остове осталось хотя бы две вершины, взорвем предка самого глубокого листа, а также всех его потомков. Чтобы найти самый глубокий лист, отсортируем все вершины по убыванию глубины. Заметим, что самая глубокая вершина, которая ещё не была удалена, обязательно является листом.
- Если в конце остался один корень, то взорвем его вместе с любым из его сыновей (такой сын найдется, потому что корень не изолированная вершина).

Время работы:

1. Сортировка: $\mathcal{O}(n \log n)$ или $\mathcal{O}(n)$ подсчетом.
2. Алгоритм: $\mathcal{O}(1)$ на каждой итерации. Всего итераций $\mathcal{O}(n)$, поэтому суммарное время работы $\mathcal{O}(n)$.

Разбор задачи «Светский приём»

Для нахождения времени перемещения агента Инглиша и Бофа воспользуемся формулой $t = \frac{S}{u}$, где t — время, S — пройденное расстояние, а u — скорость перемещения. Поскольку в точку Боф и агент Инглиш должны прийти одновременно, и при этом скорость Инглиша p, а скорость Бофа q, то $\frac{S_p}{p} = \frac{S_q}{q} \Rightarrow \frac{S_p}{S_q} = \frac{p}{q} \Rightarrow$ в искомом маршруте его длина должна быть в $\frac{p}{q}$ раз больше перемещения (расстояния от первой точки маршрута до последней).

Под это условие подходит маршрут из четырёх точек: $(0, 0) \to (2q, 0) \to (2q, p - q) \to (2q, 0)$.