Разбор задачи «Туризм»

Докажем, что один из оптимальных ответов — это найти два самых далёких одинаковых символов и взять в качестве ответа отрезок между этими двумя символами, продленный на единицу влево и вправо. Эти отрезки будут иметь одинаковую сумму, поскольку сумма в каждом из них равна общей части двух отрезков вместе с добавленным символом, который одинаков для обоих отрезков.

Покажем теперь, что такое построение дает максимальную длину отрезков. Рассмотрим первый символ левого отрезка в оптимальном ответе. Заметим, что этот символ не лежит в другом отрезке, поскольку мы рассматриваем левый из двух отрезков. Пусть он стоит на месте x. Посмотрим на позиции, которые лежат в правом отрезке, но не лежат в левом. Среди них есть символ, который совпадает с символом на позиции x. Ответ, построенный на этих двух символах будет не короче, чем изначальный ответ. Значит, выбирать отрезки таким образом — оптимально.

Найдём такой ответ. Переберём, чему будет равен совпадающий символ, «`0`» или «`1`». Среди всех таких символов надо взять два самых далёких, то есть первый и последний. Найти их можно за время $\mathcal{O}(n)$, и общее время работы выходит равным $\mathcal{O}(n)$.

Разбор задачи «Открытый кубок»

Поскольку необходимо, чтобы существовал отрезок, перпендикулярный обеим выбранным сторонам, эти стороны должны быть параллельны.

Поскольку многоугольники выпуклые, если мы знаем одну из сторон, которые должны быть в ответе, остается всего не больше двух параллельных ей сторон во втором многоугольнике. Кроме того, вершины многоугольников уже заданы в порядке обхода против часовой стрелки, и мы можем рассматривать ориентированные стороны, в направлении от вершины i к вершине $i + 1$. На самом деле, если мы зафиксировали сторону первого многоугольника, которая будет в ответе, нас интересует лишь одна сторона другого многоугольника — не просто параллельная, но еще и противоположная по направлению к выбранной, так как если обе стороны сонаправлены, легко понять, что отрезок между ними обязательно будет пересекать один из многоугольников.

Таким образом, есть не больше n пар сторон, которые могут быть кандидатами на ответ. Для поиска этих пар можно использовать метод двух указателей — перебираем сторону первого многоугольника, двигаем вместе с ней кандидата на противоположную к ней в другом многоугольнике. Для сравнения углов между прямыми нужно смотреть на знак векторного произведения.

Для пары сторон можно проверить, является ли эта пара сторон решением задачи, за $O(1)$ — для этого нужно проверить, правда ли отрезок между ними не пересекает многоугольники (достаточно посмотреть, с правильной ли стороны соседние со стороной точки), и пересекаются ли проекции этих сторон на направляющий вектор, параллельный этим прямым — это можно сделать, используя скалярное произведение.

Итоговая сложность решения — $O(n)$.

Разбор задачи «Конфигурационный файл»

Будем читать файл построчно и поддерживать для каждой переменной стек значений, которые были присвоены. Кроме того, для каждого уровня вложенности будем поддерживать множество переменных, которые были изменены на этом уровне.

При выходе из блока необходимо для всех переменных, которые были изменены в этом блоке, убрать последнее число из стека.

Для того, чтобы узнать текущее значение переменной, необходимо посмотреть на вершину ее стека. При присвонии переменной нового значения, необходимо добавить его в стек. Причем, если в текущем блоке значение переменной уже обновлялось, со стека предварительно нужно убрать старое значение.

Разбор задачи «Совпадающие максимумы»

Для каждого элемента массива предподсчитаем (при помощи стека) индекс следующего (и предыдущего) строго большего элемента. Они понадобятся для вычисления формул и для проверки.

Переберём число T: значение максимума. Выделим в левом отрезке $(a_i, a_{i+1}, \ldots, a_j)$ правое вхождение T, а в правом — левое. Бывают три варианта того, что происходит между отрезками, то есть на промежутке $(a_{j+1}, \ldots, a_{k-1})$:

1. все числа строго меньше T,
2. есть числа, большие T, но нет равных,
3. хотя бы один раз встречается ровно T.

Пройдёмся по списку индексов в массиве, соответствующих вхождениям числа T (такие списки несложно построить заранее для всех значений). Для каждой пары соседних проверим (при помощи «следующего большего»), генерируют ли они случаи (1) или (2) — и прибавим к ответу число, полученное по соответствующей формуле.

В случае (3) к ответу надо прибавить произведение количества способов взять правый отрезок на количество способов взять левый отрезок. Они независимы (в случае (2) аналогичная ситуация) потому что разделены (хотя бы одним) вхождением числа T. Количество правых отрезков вычисляется по формуле, количество левых — сумма по элементу, который окажется правым вхождением в левом отрезке. Этот элемент можно было бы перебирать каждый раз (но тогда в некоторых случаях решение будет работать за $O(n^2)$), а можно сумму аккумулировать по ходу.

Разбор задачи «Городская олимпиада»

Отсортируем места проведения и дома школьников по возрастанию координаты. Тогда в оптимальном распределении школьников по местам проведения, первая часть школьников пойдет в первое место проведения, вторая часть — во второе и третья — в третье.

Переберем, какая часть школьников пойдет в первое место проведения. При этом, будем перебирать только те значения, при которых существует хотя бы какое-нибудь распределение школьников, удовлетворяющее условиям. А именно, количество школьников, которые пойдут в первое место, не должно превышать его вместительность, и количество оставшихся школьников не должно превышать суммарную вместительность оставшихся двух мест проведения. Разделим оставшихся школьников на две группы: те, кто ближе ко второму месту проведения, и те, кто ближе к третьему месту проведения. Если одна из групп имеет размер больше, чем вместительность соответствующего места проведения, переместим минимальное количество граничных школьников из этого места проведения во второе, чтобы вместительность стала не превышена. Обновим таким распределением школьников ответ. Чтобы посчитать значение для фиксированного распределения школьников, нужно отрезок школьников для каждого места проведения разбить на два: содержащий тех школьников, которые находятся левее этого места, и содержащий тех, которые находятся правее этого места. А затем, взять разность суммы координат на отрезке школьников и координаты места проведения, умноженной на количество школьников на отрезке.

Разбор задачи «Пазл»

Посчитаем количество различных насадок для каждого типа детали пазла:

- Для угловых деталей количество различных насадок равно $(2k)^2$;
- Для крайних деталей количество различных насадок равно $(2k)^3$;
- Для обычных деталей по лемме Бёрнсайда количество различных насадок будет равно $\frac{1}{4} \cdot \sum_{i=0}^{i=3} (2k)^{\gcd(i,4)}$, где $\gcd(a, b)$ — наибольший общий делитель чисел a и b.

  Можно было найти ответ и без помощи леммы Бёрнсайда — например, посчитать $(2 \cdot k)^4$, то есть общее количество насадок, а затем вычитать из этого числа детали, посчитанные несколько раз. В этом случае получается следующая формула:

$$(2k)^4 - 5 \cdot 2k \cdot (2k-1) - \frac{9 \cdot 2k \cdot (2k-1) \cdot (2k-2)}{2} - \frac{2k \cdot (2k-1) \cdot (2k-2) \cdot (2k-3)}{4}$$

Разбор задачи «Игра в 2-SAT»

Чтобы выиграть, нам достаточно добиться того, чтобы хотя бы один дизъюнкт стал $false$.

Пусть есть некоторая переменная x, которая входит в дизъюнкты с двумя различными переменными y и z. Есть два разных случая с точностью до замен переменных на отрицания:

1. $(x|y)\&(x|z)$
2. $(x|y)\&(!x|z)$

В пером случае мы можем выиграть, сделав первым ходом $x = false$. Тогда второму, чтобы не дать нам победить, придётся сделать y и z $true$, но он не может сделать это за один ход.

Во втором случае сделаем $y = false$. Теперь второй вынужден сделать $x = true$, иначе мы сможем сделать ложным первый дизъюнкт. После этого ходим $z = false$, и второй дизъюнкт становится ложным.

Таким образом, если есть переменная, которая входит в дизъюнкты с двумя различными переменными, то мы выиграли. Рассмотрим случай, когда это не так. В этом случае переменные разбиты на пары, и для каждой пары дизъюнкты задают некоторую булеву функцию. Мы должы занулить хотя бы одну из них. Кроме того, может быть сколько-то свободнх переменных.

Рассомтрим эти пары и заданные на них функции. Они бывают трёх типов:

1. Функции, которые мы можем занулить за свой первый ход. Это $f = 0$, $f = x\&y$, $f = x$, $f = y$, а также те, коорые получаются из них заменой переменных их отрицаниями. Наличие такой пары гарантирует нам победу.
2. Функция $f = x|y$ (и такие же с отрицаниями). С одной стороны, на любой наш ход в такую пару второй может ответить и не проиграть. С другой стороны, он сам может сходить в такую пару так, что мы не сможем занулить f. Поэтому такие пары ни на что не влияют.
3. Функция $f = x \oplus y$ или $f =!(x \oplus y)$. Второму нельзя первому ходить в эту пару, так как в этом случае мы можем ответить и занулить её. На наш первый ход в такую пару он всегда может ответить так, чтобы она не занулилась. Таким образом, если есть такая пара и n нечётно, то мы выиграем: если оба игрока будут избегать этой пары, то из-за нечётности n первым в неё сходит второй.

Таким образом, мы можем выиграть, если есть пара типа 1, либо если n нечётно и есть пара типа 3.

Разбор задачи «Электрическая цепь»

Докажем по индукции следующий факт: в хорошем графе из хотя бы двух ребер есть либо вершина степени 2, либо кратное ребро.

База. Для двух ребер существует два хороших графа — две вершины соединенные двумя ребрами и цепочка из трех вершин.

Переход. Посмотрим каким из способов получен граф. В обоих случаях, одна из половин состоит из хотя бы двух ребер, и применяя предположение к ней, получим нужное утверждение.

Решение будет состоять в следующем процессе. Найдем в графе либо два параллельных ребра либо вершину степени два, и заменим их на одно ребро. При этом будем поддерживать для каждого ребра две величины (f, p) — количество остовных деревьев и количество остовных деревьев без одного ребра для подграфа, который сжался в это ребро. В конце у нас останется одно ребро и число f для него будет ответом на задачу.

Если есть два параллельных ребра со значениями (f_1, p_1) и (f_2, p_2),то для их объединения $(f, p) = (f_1 \cdot p_2 + f_2 \cdot p_1, p_1 \cdot p_2)$. Если есть вершина степени два, для ребер которой значения (f_1, p_1) и (f_2, p_2), то для объединения $(f, p) = (f_1 \cdot f_2, f_1 \cdot p_1 + f_2 \cdot p_2)$.

Для эффективной реализации этого решения, будет хранить ребра каждой вершины в структуре данных позволяющей быстро добавлять и искать ребра (например map), а так же будем хранить очередь вершин степени два. При добавлении ребра, если такое уже было в графе, сразу склеим два найденных параллельных ребра. Добавим все вершины степени два в очредь. Достаем очередную

вершину, удаляем два инцидентных ей ребра, добавляем новое. Проверяем, не стала ли у соседей степень 2 (такое может быть, если между ними уже было ребро).

Сложность такого решения — $O(m)$ или $O(m \log m)$ в зависимости от аккуратности и выбранных структур.

Разбор задачи «Развлечение с копьями»

Никогда не выгодно бросать копьё, если его диаметр больше диаметра предыдущего брошенного копья. Значит нам интересны только подпоследовательности копий, в которых диаметр невозрастает. Среди них нужно выбрать подпоследовательность с суммарной прочностью не меньше m и с минимальной суммарной стоимостью.

Будем считать $D(i, j)$ ($1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant m + 2000$) — минимальная стоимость интересной подпоследовательности копий с номерами не больше i, которая включает в себя i-е копье и пробивает j слоёв мишени. $D(i, j) = \min_{k<i, d_k \geqslant d_i} D(k, j - c_i) + p_i$.

Для того, чтобы быстро это считать, будем для каждой суммарной прочности поддерживать структуру, в которой для каждого диаметра последнего копья будем хранить минимальную суммарную стоимость. К этой структуре будет два вида запросов: обновить значение для диаметра и взять минимум по всем диаметрам, меньших какого-то x. Дерево Фенвика умеет отвечать на эти запросы за $\mathcal{O}(\log n)$, если предварительно сжать диаметры, то есть перейти от самих диаметров к их номерам в упорядоченной последовательности.

Тогда для нахождения $D(i, j)$ сделаем запрос в структуру для суммарной прочности $j - c_i$, и после обновим в структуре для суммарной прочности j стоимость для диаметра d_i.

Ответ — $\min_{j \geqslant m} D(i, j)$. Номер последнего брошенного копья — это i. Номер предпоследнего брошенного копья — такое k, что $d_k \geqslant d_i$ и $D(k, j - c_i) + p_i = D(i, j)$. Аналогично найдём все остальные брошенные копья.

Все $D(i, j)$ посчитаем за $\mathcal{O}(n(m + 2000) \log n) = \mathcal{O}(nm \log n)$, ответ восстановим за $\mathcal{O}(n(m + 2000)) = \mathcal{O}(nm)$, значит суммарно решение работает за $\mathcal{O}(nm \log n)$.

Разбор задачи «Смишенное произведение»

Заметим, что остаток числа b_{ij}, полученного из чисел a_i и a_j по модулю $10^9 + 7$ равен остатку $a_i * 10^{k_j} + a_j$, где k_j — это минимальное натуральное k_j, такое, что $10^{k_j} > a_j$.

Значит остаток суммы равен остатку $\sum_{i=1}^{n}(a_i \cdot \sum_{i=1}^{n} 10^{k_i}) + \sum_{i=1}^{n} a_i \cdot n - \sum_{i=1}^{n}(a_i \cdot 10^{k_i} + a_i)$. Этот остаток легко считается за $\mathcal{O}(n \cdot \log_{10} n)$.

Разбор задачи «Простые суффиксы»

Для решения задачи необходимо заметить, что если Сене нравится некоторое число x, то ему нравятся и все его суффиксы — числа, получающиеся отбрасыванием произвольного количества первых цифр.

Для хранения подходящих чисел будем использовать очередь. Начнём с однозначных чисел — 2, 3, 5 и 7 (если они не превышают b). Доставая число из очереди, попробуем приписать к нему слева все цифры от 1 до 9. Каждый раз, когда получилось простое число, не превышающее b, положим его в очередь. Количество чисел в очереди, не меньших a, будет ответом на задачу.

Оказывается, что подходящих чисел до 10^{11} не так уж и много — всего 3131. Благодаря этому такой алгоритм работает достаточно быстро. Однако, эффективность зависит от алгоритма проверки числа на простоту. Например, чтобы узнать, является ли число x простым, можно проверить, делится ли оно на простые числа, не превышающие $\sqrt{x}$, предварительно найдя все простые числа не более $\sqrt{10^{11}}$.

Разбор задачи «Восстановление последовательности»

Если осталось не $n - 1$ число, то ответ — «No».

Сравним каждое число с предыдущим, если они отличаются не на 1 и не на 2, то ответ «No». Если они отличаются ровно на 2, то между ними кандидат на ответ.

Если таких кандидатов больше одного, то ответ «No», иначе мы нашли искомое число.