Разбор задачи «Сумасшедшие транспортные налоги»

Заметим, что таблица налоговых ставок упорядочена по строгому возрастанию мощностей, задающих диапазоны. Тогда для ответа на каждый запрос можно использовать двоичный поиск для поиска последней строки таблицы такой, что в ней мощность строго меньше, чем в запросе.

Далее нужно было умножить мощность из запроса на найденную ставку, и не забыть про 64-битный тип данных для результата. Итоговая сложность решения $O(m \log(n))$.

Разбор задачи «Безумный танец»

Пусть в массиве c_i мы будем считать, сколько раз Джокер произнёс i-ю цифру. Изначально этот массив заполнен нулями. Заметим, что каждую секунду каждый элемент этого массива не уменьшается, а хотя бы один элемент увеличивается. Поэтому сумма всех элементов этого массива каждую секунду строго увеличивается. Поэтому, Если танец Джокера конечен, то он не может длиться дольше секунд, чем сумма элементов массива b_i. А также заметим, что так как сумма элементов массива c_i строго увеличивается, мы можем с помощью бинарного поиска найти первый момент, когда она стала хотя бы такой же, как сумма элементов массива b_i. Тогда либо в этот момент эти массивы равны, и тогда ответ это этот момент, либо они не равны, и тогда Джокер будет танцевать вечно.

Осталось понять, как по моменту времени восстановить массив c_i. Пусть момент времени равен x. Для чисел с длиной меньше длины x мы знаем, что каждое из них будет произнесено. Для чисел с длиной равной длине x, нужно перебрать длину префикса числа, которая совпадает с префиксом x, перебрать следующую цифру, которая должна быть строго меньше соответствующей цифры в x, и тогда число может заканчиваться на любую последовательность цифр правильной длины.

Разбор задачи «Лечебница Аркхем»

Дано множество точек. Требуется провести минимальное количество прямых, чтобы любые две точки были разделены хотя бы одной прямой. Каждая прямая разбивает точки на два множества точек — с одной стороны от прямой, и с другой. Понятно, что в рамках задачи прямые, которые образуют одинаковые множества, лежат в одном классе эквивалентности. Научимся получать все возможные классы прямых. Переберем подмножество точек, которые будут лежать по одну сторону от прямой. Проверим, что их выпуклая оболочка не имеет общих точек с выпуклой оболочкой оставшихся точек. Теперь, для двух непересекающихся выпуклых оболочек нужно найти прямую, которая их разделяет — она будет представителем класса эквивалентности. Несложно доказать, что количество классов не превышает n^2.

Рассмотрим состояние, которое мы имеем после проведения нескольких прямых. Есть некоторые множества точек, которые еще не отделены друг от друга. Причем, если точки a и b не отделены, и точки b и c не отделены, не сложно заметить, что a и c тоже не отделены. Осталось реализовать аналог bfs-а по состояниям, где переходом будет добавление одной из прямых. При ограничении $n \leqslant 12$, количество различных состояний оказывается порядка 140 000. Поэтому, данное решение укладывается в ограничение по времени.

Разбор задачи «Ограбление банка»

Эта задача может быть решена методом динамического программирования. Вычислим значение $dp[i][j]$ — количество последовательностей длины i, заканчивающихся на символ j, и соответствующих первым i числам из подсказки. Понятно, что $dp[1][a_1] = 1$, а все остальные $dp[1][i] = 0$. Для того, чтобы посчитать значение $dp[i][j]$, нужно сложить $dp[i-1][j-a_i]$ и $dp[i-1][j+a_i]$ (каждое из значений прибавляется только в том случае, если существует соответствующее состояние).

Разбор задачи «Странная игра на графе»

Для начала заметим, что у графа может быть несколько компонет связности. При этом первый игрок, первым ходом выбирая ребро, фиксирует ее. Значит можно рассматривать компоненты связности независимо друг от друга.

Теперь рассмотрим, какая компонента связности будет выигрышной. Можно доказать, что если в компоненте связности нечетное число ребер, первый игрок может выиграть. Для этого, сделаем

лемму: В любом связном графе, содержащем нечетное число ребер, для любой вершины можно выбрать смежное с ней ребро, что после его удаления, во всех получившихся компонентах связности (одной или двух), число ребер будет четно. Назовем такое ребро *хорошим*.

Тогда, первому игроку нужно будет играть следующим образом:

1. Выбрать компоненту с нечетным числом ребер
2. Выбрать в этой компоненте любую вершину v
3. По лемме, выбрать в текущей компоненте из вершины v любое *хорошее* ребро, и удалить его
4. Второй на своем ходу либо проигрывает, если не может сходить, либо удаляет ребро (a, b)
5. Если второй удалил ребро, он удалил его в компоненте с четным число ребер (по свойству *хорошего ребра*). И тогда возможны два случая:
   - Компонента, в которой находилось ребро (a, b) осталась связна. Тогда там стало нечетное число ребер, и первый игрок выбирает в качестве вершины v любую из вершин a и b
   - Компонента, в которой находилось ребро (a, b), распалась на две. Тогда ровно одна из двух получившихся компонент содержит нечетное число ребер. А a и b лежат в разных компонентах. Тогда в качестве v первый игрок выберет из вершин a и b ту, которая лежит в компоненте с нечетным числом ребер
6. Переходим к пункту 3

Первый игрок не может проиграть, потому что на его ходу в текущей компоненте нечетное число ребер (и поэтому по лемме он может выбрать *хорошее* ребро).

Доказательство леммы:

- Если есть ребро, смежное v, которое не является мостом, удалим его. Тогда компонента не распадется, и значит в этой одной компоненте будут все ребра, кроме удаленного — четное число
- Остался случай, когда все ребра, ведущие из вершины — мосты. Пусть их k. Пойдем от противного: пусть ни одно из этих ребер не является *хорошим*. Тогда, после удаления любого ребра, в двух получившихся компонентах будет нечетное число ребер. Значит, в каждой из k компонент, которые получаются, если удалить v со всеми смежными с ней ребрами, будет нечетное число ребер. Тогда посчитаем количество ребер в исходном графе по модулю 2: $k + \sum\limits_{i=1}^{k} e_i \underset{2}{\equiv} k + \sum\limits_{i=1}^{k} 1 \underset{2}{\equiv} k + k \underset{2}{\equiv} 0$ (e_i — количество ребер в i-й компоненте). Противоречие.

Можно заметить, что второй случай покрывает в том числе случай, если степень вершины v равна 1.

Таким образом, чтобы решить задачу, нужно пройтись по всем компонентам связности, например обходом в глубину, и проверить есть ли среди них та, что содержит нечетное число ребер. Итоговая сложность решения $O(n + m)$.

Разбор задачи «Беспорядочное выступление»

Будем решать задачу жадным алгоритмом. Очевидно, что порядок уменьшения порядочности не играет роли, а уменьшения порядочности разных людей не зависят друг от друга, поэтому суммарное внимание будет минимально, если мы будем уменьшать порядочность «самых наблюдаемых» зрителей, то есть тех, за которыми следит наибольшее число полицейских.

Для этого, для каждого зрителя найдем количество следящих за ним полицейских: заведем события $(+1, l)$ и $(-1, r)$ для каждого наблюдаемого отрезка зрителей $[l, r)$. Эти события можно отсортировать за $\mathcal{O}(n \log n)$ или за $\mathcal{O}(n)$ с помощью подсчета, так как ограничения на n позволяют это сделать. После этого, обработав события в порядке их следования и посчитав на них префиксные суммы, мы получаем для каждого зрителя количество полицейских, следящих за ним.

Осталось только в порядке уменьшения этих значений изменять порядочность зрителей на минимум из их текущей порядочности и оставшейся харизмы k. Алгоритм завершаеися, если все зрители имеют порядочность 0 или если $k = 0$.

Разбор задачи «Сумасшедшее домино»

Несложно привести подходящую расстановку шашек. Например, для чётных n:

```
#..#
....
#..#
....
```

Для нечётных n:

```
#....
....#
#....
....#
#....
```

Шашки в крайних столбцах делают замощение доминошками однозначным.

Разбор задачи «Этажи»

Если изначально Артур оказался на этаже x, на котором есть табличка, он сразу знает номер текущего этажа и сделает $|x - k|$ переходов. Иначе, у Артура есть всего два варианта стратегии: идти вниз, пока он не дойдет до этажа с табличкой, или идти вверх, пока он не дойдет до этажа с табличкой. Поэтому, в решении нужно перебрать оба варианта стратегии, вычислить ответ для каждой, и выбрать лучший вариант. Пусть изначально Артур был на этаже x, решил идти вверх, и ближайший этаж сверху, на котором есть табличка, это y. Тогда он сделает $|x-y|+|y-k|$ переходов.

Разбор задачи «Вырваться из окружения»

Разделим задачу на 4 квадранта относительно клетки с Джокером. Найдем клетки, Манхэттенское расстояние до которых равно d, находящиеся на границе. Внутри квадранта искомые клетки лежат на одной диагонали, а значит, зная две крайние клетки, можно найти количество клеток, принадлежащих этой диагонали

Разбор задачи «Убийственная математика»

Заметим, что из каждой пары (a, b) мы можем получить четыре различных результата. Сделаем *bfs* по такому неявному графу и получим минимальное «расстояние» от исходной пары до пары с двумя одинаковыми элементами.

Данный алгоритм работает за время $\mathcal{O}(b^2)$. Количество операций до достижения ситуации $a = b$ меньше $\log_2 b$, так как есть последовательность изменений, в которой расстояние между числами каждый раз уменьшается хотя бы в два раза (всегда заменять b на меньшее из двух средних, например). А тогда максимальное число состояний *bfs* равно $4^{\log_2 b} \sim b^2$.

Так же за ту же асимптотику работает метод динамического программирования по подотрезкам, где $\mathtt{dp}_{i,j}$ равно минимальному числу операций до достижения пары равных чисел. Пересчет происходит за $\mathcal{O}(1)$, так как из каждого состояния есть не более четырех переходов.

Тем не менее, есть жадный алгоритм, так же решающий эту задачу, но за $\mathcal{O}(b)$, который каждый раз заменяет b на минимальное из двух средних, однако сложность его доказательства не позволяет рассматривать его как ожидаемое решение, и поэтому все правильные решения за $\mathcal{O}(b^2)$ получали вердикт «OK».

Разбор задачи «Безумные расстановки»

Докажем, что ответ не зависит от дерева. Введем переменную h_v, равную исключающему ИЛИ значений на пути от вершины v до корня (за корень можно взять любую вершину). Тогда исключающее ИЛИ на пути между вершинами u и v равно $h_u \oplus h_v$.

По значениям h_v, для любого дерева и корня r, веса ребер восстанавливаются однозначно, если $h_r = 0$. Таким образом, мы просто свели задачу к подсчету правильных массивов h.

Зафиксируем границу, которая разделяет пути, на которых исключающее ИЛИ равно 0, и пути, на которых исключающее ИЛИ равно 1.

Тогда у нас есть условия $h_{u_i} \oplus h_{v_i} = q_i$ (где $q_i = 0$ для путей слева от границы, и $q_i = 1$ для путей справа от границы), и нужно посчитать число массивов, подходящих под такие ограничения.

За $O(n + m)$ это можно сделать алгоритмом, подобным покраске графа в два цвета. Тогда решение будет работать за $O(m \cdot (n + m))$. Чтобы оптимизировать это решение дальше, предлагается воспользоваться техникой «`разделяй и властвуй`». Будем делать аналогично алгоритму Dynamic Connectivity. Когда идем в левую половину отрезка, добавляем в DSU правую половину с весами 1. Когда идем в правую половину, добавляем в DSU левую половину с весами 0.

Таким образом, время составит $O(m \log^2)$, а также может быть соптимизировано до $O(m \log)$ используя обход в глубину и явное сжатие графа вместо DSU с откатами.