Задача A. Жизни цветов важны

Идем по массиву и смотрим на наш элемент и предыдущий, возможны 4 случая:

- $a_i == 1$ и $a_{i-1} == 1$ — k += 5
- $a_i == 1$ и $a_{i-1} == 0$ — k += 1
- $a_i == 0$ и $a_{i-1} == 1$ — k += 0
- $a_i == 0$ и $a_{i-1} == 0$ — k = -1, break

Задача B. Выворачивание массива

Лемма: Если x — максимальный элемент массива, то выворачивание не изменит массив.

Доказательство: Так как разделение стабильное и весь массив уйдёт в левую часть, относительный порядок элементов не изменится.

Лемма: После выворачивания последним становится самый правый элемент, находящийся слева от x в исходном массиве и больший x.

Доказательство: Рассмотрим все элементы большие x. Так как разделение стабильно, самым правым элементом станет самый правый элемент правой части. Он не мог быть правее x по построению выворачивания. Он точно больше чем x. Лемма доказана.

Построим последовательность $x_a, x_{a-1}, \dots x_0$, где $x_0 = a_n$, x_{i+1} — самый правый элемент левее x_i, который больше него. Ответом является число a так как $\{x_i\}$ — последовательность самых правых элементов массива во время выворачивания.

Пример:

6 10 4 17 9 2 8 1

Последовательность $\{x_i\}$ — это $1, 8, 9, 17$. Ответ на задачу — 3.

Задача C. Минимизируйте расстояние

Это задача может быть решена при помощи жадного алгоритма.

Заметим, что выгодно решать задачу для положительных x^p и отрицательных x^n точек по отдельности, так как при переходе через начало координат мы можем взять новый набор коробок, а не продолжать нести старый.

Заметим, если мы донесли коробку до какого-то склада, то мы прошли и все склады ближе. Поэтому за один поход мы выгодно покрывать ближайшие непокрытых k складов (с одной стороны от начала координат), либо все оставшиеся, если их меньше.

Поэтому мы можем отсортировать все x^p, x^n и найти ответ по следующим правилам:

$$sum_{pos} = \sum_{i=0}^{|pos|-k\cdot i>=0} x^p_{|pos|-k\cdot i}$$

$$sum_{neg} = \sum_{i=0}^{|neg|-k\cdot i>=0} x^n_{|neg|-k\cdot i}$$

Ответом на задачу будет $2(sum_{pos}+sum_{neg})$ минус максимум из расстояний от начала координат до складов так как мы не обязаны возвращаться в начало координат в конце.

Задача D. Ещё одна задача на сортировку

Множество 3-циклов порождает группу чётных перестановок A_n. Поэтому ответ «YES» тогда и только тогда, когда существует чётная перестановка, сортирующая массив a.

Если все элементы a различны, то сортирующая перестановка уникальна, и её четность совпадает с чётностью перестановки a.

Если в a есть совпадающие элементы, то давайте посмотрим на произвольную сортирующую перестановку. Если она нечётная, то добавим в неё транспозицию одинаковых элементов, она останется

сортирующей, но станет чётной. Таким образом, в этом случае, чётная сортирующая перестановка всегда существует.

В итоге, нам сначала надо проверить, что все элементы a различны. Если это не так, то ответ «YES». Иначе проверим, что перестановка a чётная. Это можно сделать множеством способов, в том числе и за $\mathcal{O}(n)$.

Задача E. Запросы частот

Давайте запустим обход в глубину от корня, поддерживая масссив-счётчик (cnt_x := «кол-во вхождений x в последовательность»). Когда dfs заходит в вершину v, то увеличиваем cnt_{a_v} на 1 и обрабатываем все запросы, относящиеся к вершине v. Когда dfs покидает вершину, то уменьшим cnt_{a_v} на 1.

Давайте поддерживать следующие величины:

- Сортирующую перестановку p массива cnt, изначально — $1, 2, \ldots, n$.
- Обратную ей перестановку p^{-1}.
- Для каждого $x \in \{0, 1, \ldots, n\}$, «lower_bound» lb_x в отсортированном массиве cnt. Более формально, наименьшее такое i, что $cnt_{p_i} \geqslant x$.

Когда мы хотим увеличить cnt_x на 1:

- Переместим x в конец блока с такими же значениями в сортированном массиве. То есть, мы должны поменять местами (p_i^{-1})-ю и $(lb_{cnt_x+1} - 1)$-ю позиции p.
- Изменим p^{-1} согласно изменению p.
- Уменьшим lb_{cnt_x+1} на 1. Заметим, что это единственное значение lb, которое изменится при этой операции.
- Увеличим cnt_x.

Операция уменьшения cnt_x на 1 выполняется симметрично.
Заметим, что если ответ существует, то один из возможных ответов — p_{lb_l+k-1}.
Итоговая асимптотика: $\mathcal{O}(n + q)$.

Задача F. Неравные соседи

Будем решать задачу с помощью формулы включений-исключений. Пусть i-е свойство означает, что элементы b_i и b_{i+1} совпали. Тогда для каждого $k = 1, \ldots, n-1$ массив разбивается на $n - k$ подряд идущих отрезков, где все числа на каждом из отрезков равны.

Дальше будем использовать метод динамического программирования $dp[i][j]$ − разбили префикс массива b длины i на некоторое число отрезков, где j обозначает четность числа отрезков.

Будем перебирать индекс $i = 1, \ldots, n$. Теперь для каждого j $(1 \leqslant j < i)$ нужно прибавить к $dp[i][0] += dp[j-1][1] \cdot f(j, i)$, где $f(j, i)$ - минимум из чисел в массиве a на отрезке $[j, i]$. Аналогично $dp[i][1] += dp[j-1][0] * \cdot f(j, i)$.

Получаем решение за $O(n^2)$. Чтобы ускорить его до $O(n)$ достаточно поддерживать стек минимумов на префиксе и делать пересчет $dp[i][0/1]$ с его помощью.

Задача G. Браконьеры

Описанное здесь решение использует теорию Шпрага-Гранди.

Можно заметить, что все различные позиции, которые появляются в игре, это какие-то поддеревья изначальных деревьев леса, и их прямые суммы. Для того, чтобы решить задачу нам достаточно найти гранди-значения для всех деревьев.

Будем использовать динамику по поддеревьям. Будем отождествлять вершину с её поддеревом. В вершине будем хранить массив гранди-значений по всем возможным ходам, в игре на её поддереве. Заметим, что длина этого массива равна рангу поддерева вершины. Будем считать, что чем меньше глубина разреза, тем позже ход встречается в массиве. Также будем хранить гранди-значение для самой вершины, оно равно MEX-у от гранди-значений ходов. Пересчёт у динамики будет такой:

- Если вершина лист, то у неё единственный ход в пустое дерево.
- Если у вершины ровно один ребёнок, то для получения массива родителя мы должны просто добавить гранди-значения ребёнка в конец массива ребёнка. Надо также пересчитать *MEX*, чтобы получить гранди-значение для родителя. Заметим, что *MEX* здесь мог только увеличится.
- Если у вершины несколько детей, то найдем наименьшую длину для массивов динамики детей. Мысленно обрежем все массивы по этой длине. В массиве родителя положим массив такой же длины, где лежат *XOR*-ы значений по соответствующим позициям массивов детей. В конец надо также добавить *XOR* всех гранди-значений детей. В конце надо честно посчитать *MEX* с нуля, чтобы получить гранди-значение для родителя.

Почему же это работает быстро? Будем использовать метод монеток для амортизационного анализа. Будем поддерживать инвариант, что на каждом рассматриваемом массиве лежит хотя бы ($3\cdot$ *ДЛИНА* $-$ *MEX*) монеток:

- Всегда когда мы считаем *MEX* мы будем снимать монетки при его увеличении, с массива на котором мы его считаем.
- В случая листа мы создаём массив длины 1, кладём на него 3 монетки. И честно считаем *MEX*.
- В остальных случаях мы всегда будем брать и изменять один из массивов детей, убирая из рассмотрения остальные.
- В случае одного ребёнка, мы увеличиваем его массив, докладывая на него 3 монетки.
- В случае нескольких детей: зафиксируем h, как длину кратчайшего из массивов детей. На этом массиве лежит ($\geqslant 2\cdot h$) монеток. Так как массивы всех остальных детей не короче, то на каждом лежит хотя бы ($\geqslant 2\cdot h$) монеток. Переложим h из них на кратчайший, теперь на нём лежит ($\geqslant 3\cdot h$) монеток. Потратим по h монеток с каждого из массивов, кроме кратчайшего, чтобы насчитать *XOR*-ы гранди-значений. При добавлении *XOR*-а гранди значений детей, положим на него 3 монетки. Заметим, что все массивы, кроме кратчайшего из нашего рассмотрения в этот момент пропадают.

Заметим, что добавим мы суммарно $3\cdot n$ монеток. Также заметим, что при любом действии кроме добавления мы снимаем монетку, так что их тоже будет не более $3\cdot n$.

Так как нам надо пересчитывать *MEX*, то параллельно массивы стоит поддерживать множество-значений. Количество добавлений/удалений/запросов к мультимножеству ограниченно $4\cdot n$, так как мы снимаем монетку за каждое из них, кроме последней неудачной попытки увеличить *MEX* в каждой вершине.

Итоговая асимптотика: $\mathcal{O}(n\cdot\log(n))$.