Поиск фальшивых монет

Для того, чтобы решить задачу на n запросов, спросим все префиксные суммы. Найдем настоящие элементы последовательности как разности соседних префиксных сумм.

Для того, чтобы решить задачу за $\frac{n}{2} + k$ запросов, спросим все префиксные суммы $n, n-2, n-4, \ldots, n \bmod 2$. Если $pref_i = i + i - 1 + pref_{i-2}$, то числа $i-1, i$ не заменяли. Если это равенство не выполнено спросим $i-1$. Дополнительно мы сделаем $\leqslant k$ запросов. Аналогично сможем узнать все замененные числа.

Далее все ответы на запросы a будем заменять на $\frac{p(p+1)}{2} - a$. Таким образом мы будем узнавать префиксную сумму замененных чисел.

Рассмотрим решение за $k \log n$ запросов. Будем находить замененные числа по одному слева направо. Для того чтобы найти очередное число $\geqslant i$ сделаем бинарный поиск и найдем минимальное j, такое что $pref_j > pref_i$.

Это решение можно реализовать так: рассмотрим функцию `func(l, r,` s_l`,` s_r`)`. Она ищет все замененные числа на отрезке $[l, r]$, если мы знаем что их сумма равна $s = s_r - s_l$.

- Если $s = 0$, то чисел на отрезке нет.
- Если $l \leqslant s \leqslant 2l$, то число на отрезке обязательно одно и оно равно s.
- Иначе спросим $mid = \lfloor \frac{l+r}{2} \rfloor$ и вызовем:
  `func(l, mid,` s_l`,` s_{mid}`)`,
  `func(mid + 1, r,` s_{mid}`,` s_r`)`

Операций все еще $O(k \log n)$, но константа решения становится меньше.

Рассмотрим решение за $2k \log k$ запросов. Мы реализуем `func(l, r,` s_l`,` s_r`)`. Допустим что мы знаем, что чисел на отрезке c штук. Тогда если мы спросим $p = \lfloor \frac{s}{c} \rfloor$, то хотя бы одно замененное число будет слева и хотя бы одно замененное число будет справа. Чтобы найти c сделаем бинарный поиск по нему. Нам не нужно найти точное значение c, главное разделить замененные числа на две части. За $\leqslant \log k$ запросов сможем разделить. Всего будет не более $2k$ вызовов `func`, потому что мы всегда разделяем замененные числа.

В предыдущем решении будем делать бинарный поиск по c в границах $[c_l, c_r]$, где c_l и c_r это min/max возможное количество чисел с суммой s из отрезка $[l, r]$. Как только получится разделить замененные числа, вызываем `func` от половинок рекурсивно. Такое решение сделает $\leqslant 2k + \log n$ запросов. Доказательство: можно аккуратно придумать потенциал, который всегда будет уменьшаться на $\geqslant 1$ после каждого запроса.

Для того, чтобы получить полное решение, нужно вставить в прошлое решение все возможные оптимизации и отсечения. Будем реализовывать функцию `func(l, r,` c_l`,` c_r`,` s_l`,` s_r`)`. Изначально `func(1, n, k, k, 0,` s_n`)`. Проверяем что чисел нет, либо одно число, либо весь отрезок, либо весь отрезок без одного числа. Иначе $c_{mid} = \lceil \frac{c_l + c_r}{2} \rceil$, запрос $p = \lfloor \frac{s}{c_{mid}} \rfloor$. Если разделить не смогли, вызываем `func` двигая соответствующие параметры. Если разделить смогли, то находим c_l, c_r для левой и правой половинок. Сначала вызовем `func` от той где $c_r - c_l$ меньше. Когда мы из нее выйдем мы уже будем знать сколько чисел было там и возможно подвинем c_l, c_r второй половинки.

Добавление параметров c_l, c_r в `func` позволяет использовать информацию о том, что замененных чисел было ровно k. Наша функция содержит много `break`-ов и ограничений параметров. За счет этого количество раз, которое мы не сможем разделить множество будет мало. На практике получается $\leqslant k + 30$ операций.