Разбор задачи «Урок арифметики»

Будем поддерживать множество битов, которые одинаковы у всех чисел. Изначально это множество пустое. Заметим, что это множество изменяется не более $\log(2^{20}) = 20$ раз, потому что когда какой-то бит у всех чисел стал одинаковым, он никогда не станет вновь разным. Изменять это множество может только запрос `and` — он выставляет значения некоторых битов у всех чисел равными 0. Пусть $mask$ — маска битов, которые одинаковы у всех чисел. Обозначим за cur_i i-е число, в котором стоит 0 на тех позициях, где в $mask$ стоит 1. Тогда будем отдельно хранить значение общей для всех чисел части b, отдельно a_i — массив чисел длины n, и отдельно c — число, такое, что a_i `xor` $c = cur_i$.

Теперь, когда поступает запрос `and`, если он не изменяет множество битов, одинаковых для всех чисел, то он изменяет только b. Это изменение можно обработать за $O(1)$. А именно, $b = b$ `and` x, где x — число из запроса. Если текущий запрос `and` изменяет множество битов, одинаковых для всех чисел, пробежимся по всем числам и суммарно за $O(n)$ обновим все a_i ($a_i = (a_i$ `xor` $c)$ `and` x). А также, обновим b, $mask$ и c ($b = b$ `and` x, $mask = mask$ `or` (`not` x), $c = 0$).

Если поступает запрос `xor`, мы сразу можем обновить значение b, $b = b$; `xor` (x `and` $mask$). А значения a_i мы оставим без изменения, но сделаем $c = c$ `xor` (x `and` (`not` $mask$)).

Когда поступает запрос '`?`', нужно найти количество чисел a_i, таких что $l - b \leqslant (a_i$ `xor` $c) \leqslant r - b$. Для этого, будем поддерживать дерево отрезков. Если бы c было равно 0, то требовалось бы просто найти сумму на отрезке значений с $l - b$ по $r - b$. Но когда c не равно 0, заметим, что нам нужно находить сумму на отрезке массива, к которому предварительно применили перестановку $p_i = i$ `xor` c. Применение такой перестановки к листам полного двоичного дерева равносильно изменению порядка детей у некоторых вершин. А именно, если у c i-й бит равен 1, нужно поменять порядок детей у всех вершин на глубине i. Конечно, явно менять порядок детей не нужно. Нужно лишь в функции, которая считает сумму на отрезке, в таком случае, вместо левого сына идти в правого, а вместо правого в левого. Эта модификация дерева отрезков не изменяет его время работы.

Итоговая асимптотика складывается из $O(\log(n))$ перестроений дерева отрезков, каждое из которых работает за $O(n)$. И $O(q \cdot \log(n))$ на все запросы к дереву отрезков.