Быстрый исполнитель

Автор задачи и разработчик: Даниил Орешников

Первая подгруппа, стандартно, позволяет набрать баллы, написав реализацию описанного в условии процесса. Присвоим $b \leftarrow a$, и честно выполним все $m \cdot p$ операций. Каждая операция выполняется для каждого элемента, поэтому время работы получается $\mathcal{O}(nmp)$.

Во **второй подгруппе** $m = 1$, что, в частности, означает, что всегда применяется одна и та же операция. Сразу приведем решение второй и **третьей подгруппы** вместе, потому что они почти ничем не отличаются. Посмотрим на каждый элемент последовательности отдельно. Всего применяется $m \cdot p$ операций, при каждой операции a сдвигается на d влево. Таким образом, b_i можно вычислить как

$$b_i = a_i \circ a_{(i+d) \bmod n} \circ a_{(i+2d) \bmod n} \circ \ldots \circ a_{(i+mpd) \bmod n},$$

где за $\circ$ обозначена применяемая операция.

Заметим, что для каждого элемента массовое применение такой операции можно вычислять быстро. А именно, разобьем весь a на «циклы» по сдвигу на d. Таких циклов будет ровно $\texttt{gcd}(n, d)$, и для каждого i указанная выше последовательность элементов a будет образовывать некоторый отрезок на цикле (возможно, с несколькими повторениями цикла целиком). Для операций `and` и `or` можно заметить, что целиком вошедший в последовательность много раз цикл можно отбросить, и оставить только «хвост» не больше цикла. Для операции `xor` надо учесть лишний раз `xor` всех элементов цикла, если цикл вошел в последовательность нечетное число раз.

Теперь, чтобы для каждого i посчитать значение операции на отрезке фиксированной длины, достаточно пройтись по всем циклам «окном» фиксированной ширины, поддерживая определенную информацию. Для `xor` это будет просто `xor` всех элементов на окне, для `or` и `and` — наличие в окне в каждом разряде единицы или нуля, соответственно. Время работы такого решения равно $\mathcal{O}(n \log \max(a_i))$.

Решения следующих подгрупп являются модификациями описанного решения. Будем рассматривать каждую из m операций отдельно, тогда для нее последовательность элементов, образующих в конечном итоге b_i, образует свой цикл, уже с шагом md вместо d.

В **четвертой подгруппе** нет операций `xor`, поэтому достаточно в каждом бите независимо найти последнюю операцию вида $b_i \leftarrow b_i$ `or` 1 или $b_i \leftarrow b_i$ `and` 0, которые задают фиксированное новое значение b_i вне зависимости от старого. Остальные операции b_i не меняют. Для каждой из m операций найдем соответствующее ей последнее имеющее значение применение этой операции в цикле. Опять же, для этого можно воспользоваться «скользящим окном» фиксированной ширины, либо же за $\mathcal{O}(n)$ найти для каждого элемента позицию ближайшего нуля и ближайшей единицы в каждом разряде.

Пятая подгруппа гарантирует, что достаточно обработать только один бит в каждом числе. Это позволяет неэффективным решениям с той же идеей набрать баллы за счет того, что надо обработать в ≈ 30 раз меньше независимых бит в каждом числе.

Для **полного решения** так же требовалось находить `xor` на отрезках в цикле. Однако надо было заметить, что нас интересует только множество операций `xor` после последнего действующего на значение `and` или `or`. Все до этой последней действующей операции не имеет значения, а после нее имеют значение только `xor`'ы. Чтобы находить `xor` на произвольных отрезках в цикле, достаточно посчитать аналог префиксных сумм и вычислять его как `xor` двух префиксов. Время работы всего решения получается $\mathcal{O}(nm \log \max(a_i))$.

Есть альтернативное полное решение, работающее за время $\mathcal{O}(n \log \max(a_i) + n \log p)$. Можно обрабатывать все биты одновременно, преобразовав применяемые к числам операции.

Заметим, что относительно каждого бита любая композиция трех данных операций с заданными вторыми аргументами — просто некоторая булева функция от изначального первого аргумента. Воспользуемся идеей двоичного подъема и найдем для каждого i функцию $\texttt{op}[q][i]$, соответствующую применению 2^q блоков по m операций, используя в качестве вторых аргументов элементы с шагом d, начиная с i-го. Переход от q к $q + 1$ осуществляется как вычисление композиции левой и правой

половины операций.

Если теперь аналогичным образом посчитать композиции не только 2^q блоков, а $p >> q$ блоков по всем целым неотрицательным q, останется только для каждого бита исходного числа за $\mathcal{O}(1)$ применить полученную операцию.