Xorset

СЕЗОН 7 – ЧЕТВЪРТИ РУНД

Наскоро Лора, както всеки прави рано или късно, прояви интерес към побитовата операция XOR[1]. За да я разбере по-добре тя иска да има програма, която да може да отговаря на няколко заявки.

Вашата задача е да поддържате заявките ефективно. За да се отговаря на заявките се налага да се поддържа множество от числа, позволявайки повторения. Следва кратко обяснение на какво разбираме под множество и подмножество:

Подмножество от множеството се получава като се изберат няколко от числата в множеството, без да има значение редът на избиране. Празното множество както и множеството от всички числа в дадено множество са валидни подмножества. Две множества са различни тогава и само тогава когато има поне един елемент, който присъства в едното множество, но не и в другото. Тъй като позволяваме повтарящи се елементи, то е важно да отбележим, че елементите се различават дори и ако стойностите им са еднакви – например множеството {1, 2, 2} има две различни подмножества {1, 2}.

XOR на дадено множество ще наричаме резултата от прилагане на операция XOR на всичките му елементи. Например XOR-ът на множеството {1, 2, 5} е равен на (1 xor 2 xor 5) = 6

За да помогнете на Лора трябва да напишете програма, която ефективно да поддържа следните 3 вида заявки, започвайки с празно множество:

- *"1 x"* – добавя се числото *x* в поддържаното множеството
- *"2 x e"* – иска се да разберем колко различни подмножества на поддържаното множество съдържат **четен брой елементи** и имат XOR равен на *x*. При e=0, това е цялата заявка. При e=1 се налага да дадем и пример за едно такова подмножество.
- *"3 x e"* – иска се да разберем колко различни подмножества на поддържаното множество съдържат **нечетен брой елементи** и имат XOR равен на *x*. При e=0, това е цялата заявка. При e=1 се налага да дадем и пример за едно такова подмножество.

(виж обяснението на примера за повече детайли)

Забележка: Тъй като извеждането на дълги примери при заявки 2,3 би забавило програмата – имаме няколко допълнителни ограничения (виж "Ограничения")

Забележете също, че считаме празното множество за валидно подмножество от 0 (четен брой) елемента.

Вход

От първия ред на файла `xorset.in` се въвежда числото **N** – броят заявки.

[1] Wikipedia - XOR

Xorset

СЕЗОН 7 – ЧЕТВЪРТИ РУНД

На всеки от следващите N реда се въвежда по една заявка от един от трите гореописани вида.

Изход

За всяка заявка от тип 2 и 3 изведете на един ред в изходния файл `xorset.out` броя подмножества отговарящи на условието. **Тъй като това число може да е голямо – изведете го по модул 1 000 000 007**!

Ако e=0, то не извеждайте нищо повече. Извеждането на примерно подмножество когато това не се изисква се счита за грешен отговор!

Ако e=1, то на следващия ред изведете броя елементи в намереното от вас подмножество, последван от списък с елементите, разделени с интервал. **Не е нужно** да намерите най-малкото подмножество, но подмножеството ви трябва да не съдържа повече от 50 елемента. Гарантирано е, че ако съществува поне едно подмножество отговарящо на условието в заявката, то съществува такова с под 50 елемента. **Ако не съществува нито едно такова подмножество (т.е. броя начини е 0), то изведете -1 на този ред.**

Ограничения

$1 \leq N \leq 100\ 000$

$0 \leq$ стойности на елементите в коя да е заявка $< 2^{30}$

$0 \leq$ брой заявки с e=1 $\leq 5\ 000$

Броя елементи във всеки ваш пример(подмножество) не трябва да надхвърля 50.

Ограничение за време: 2 сек
Ограничение за памет: 256 MB

Xorset

СЕЗОН 7 – ЧЕТВЪРТИ РУНД

Примерен тест

Вход (xorset.in)	Изход (xorset.out)
10 2 0 0 1 1 1 2 1 4 2 3 1 3 3 1 1 3 1 6 2 3 1 3 3 1	1 1 2 2 1 0 -1 2 4 2 3 4 6 2 1 3

Пояснения

Първата заявка е зададена докато множеството е празно. Има единствено решение – празното подмножество {}. Не се иска да го изведем (извеждането му би изглеждало просто като “0” на следващия ред).

Следващите три заявки добавят числата 1, 2 и 4 – правейки множеството {1, 2, 4}. Заявката “2 3 1” иска да намерим броя подмножества с четен брой елементи и XOR равен на 3. Има само едно такова – {1, 2}. Понеже e=1, то го извеждаме.

Следващата заявка пита отново за XOR равен на 3, но с нечетен брой елементи в подмножеството. Няма такова подмножество в текущото ни множество, затова извеждаме 0. Не можем да дадем пример, но e=1, така че извеждаме -1.

Следва добавяне на 3 и 6 в множеството – правейки го {1, 2, 3, 4, 6}.

Отново имаме по-ранните две заявки. Този път обаче вече имаме по две подмножества отговарящи на всяка:

С четен брой елементи и XOR равен на 3:

{2, 3, 4, 6} (3 = 2 xor 3 xor 4 xor 6)

{1, 2} (3 = 1 xor 2)

С нечетен брой елементи и XOR равен на 3

{3} (3 = 3)

{1, 4, 6} (3 = 1 xor 4 xor 6)

Понеже e=1, извеждаме и по едно примерно подмножество (без значение кое, тъй като всички са с по-малко от 50 елемента).