Задача A. Сбалансированное соседство

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Вам дано целое число n. Требуется построить неориентированный граф на n вершинах, пронумерованных от 1 до n. Граф должен удовлетворять следующим двум условиям:

- Граф связный, не содержит петель и кратных рёбер.
- Существует целое число s, такое что для каждой вершины i сумма индексов вершин, соединённых с вершиной i, равна s.

Можно показать, что существует хотя бы один граф, удовлетворяющий ограничениям задачи.

Формат входных данных

В единственной строке вводится целое число n ($3 \leqslant n \leqslant 100$) — требуемый размер графа.

Формат выходных данных

В первой строке выведите единственное число m — количество рёбер в построенном вами графе.
В следующих m строках требуется вывести описание рёбер построенного вами графа.
Строка i должна содержать два целых числа a_i и b_i — очередное ребро вашего графа.

Пример

стандартный ввод	стандартный вывод
3	2 1 3 2 3

Замечание

Для каждой вершины i сумма индексов вершин, соединённых с вершиной i, равна 3.

Задача B. Странный прибор

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

Вася обожает решать загадки и разгадывать головоломки. На этот раз он нашел странный прибор и хочет выяснить принцип его работы.

Этот прибор зашифрован с помощью дерева (связного неориентированного графа без циклов), состоящего из n вершин, пронумерованных целыми числами от 1 до n. Чтобы решить головоломку надо угадать это дерево.

К счастью прибор умеет выполнять одну операцию, исходя из которой надо разгадать его шифр. Можно ввести в прибор последовательность $d_1, d_2, \ldots, d_n$ целых неотрицательных чисел. На приборе есть n лампочек, i-я из которых отвечает за i-ю вершину дерева-шифра. Для всех i из этих лампочек i-я загорится, если существует такая вершина дерева-шифра с номером $j \neq i$, что $dist(i, j) \leqslant d_j$. Здесь $dist(i, j)$ обозначает расстояние между вершинами i и j в дереве-шифре, то есть количество ребер в простом пути между вершинами i и j.

Вася хочет за $\leqslant 80$ операций с прибором решить головоломку и угадать дерево-шифр. Помогите ему!

Протокол взаимодействия

В начале вашей программе вводится единственное целое число n — количество вершин в дереве-шифре прибора ($2 \leqslant n \leqslant 1000$).

Далее вы можете выполнять операции в следующем формате. Сначала выведите символ "`?`" (без кавычек) и за ним n целых чисел $d_1, d_2, \ldots, d_n$, разделенных пробелами. Заметьте, что в операциях для всех i должно быть выполнено неравенство $0 \leqslant d_i < n$. В ответ будет выведена строка s длины n, состоящая из символов "`0`" и "`1`" (без кавычек). Для всех i символ s_i будет равен "`0`", если у прибора не включилась лампочка, соответствующая i-й вершине дерева-шифра и "`1`", иначе.

После нескольких запросов вы должны вывести угаданное дерево. Для этого в первой строке выведите единственный символ "`!`" (без кавычек). В следующих $n - 1$ строках выведите по 2 целых числа a_i, b_i — номера вершин, соединяющих i-е ребро дерева. Выведенные числа должны удовлетворять условиям $1 \leqslant a_i, b_i \leqslant n$ и $a_i \neq b_i$. Эти ребра должны образовывать дерево, совпадающее с загаданным. Выводить ребра можно в любом порядке. После этого ваша программа должна завершиться.

Гарантируется, что в каждом тесте дерево-шифр будет зафиксировано заранее и не будет меняться в зависимости от выполняемых операций.

Ваша программа может выполнить от 0 до 80 операций с прибором и после этого сообщить дерево, совпадающее с загаданным.

Если ваша программа сделает больше 80 операций, то она может получить любой вердикт, потому что будет считывать данные из закрытого потока ввода. Также, если ваша программа сделает операцию или выведет ответ в неверном формате, она может получить любой вердикт. **Будьте внимательны.**

Не забудьте сбрасывать буфер вывода после того, как выведете данные для операции или ответ.

Чтобы сбросить буфер вывода вы можете использовать:

- fflush(stdout) в C++.
- System.out.flush() в Java.
- stdout.flush() в Python.
- flush(output) в Pascal.
- Прибегните к документации других языков.

Пример

стандартный ввод	стандартный вывод
5 00000 11011 11100 10010	? 0 0 0 0 0 ? 1 1 2 0 2 ? 0 0 0 1 0 ? 0 1 0 0 1 ! 4 2 1 5 3 4 4 1

Замечание

Дерево-шифр из первого теста выглядит так:

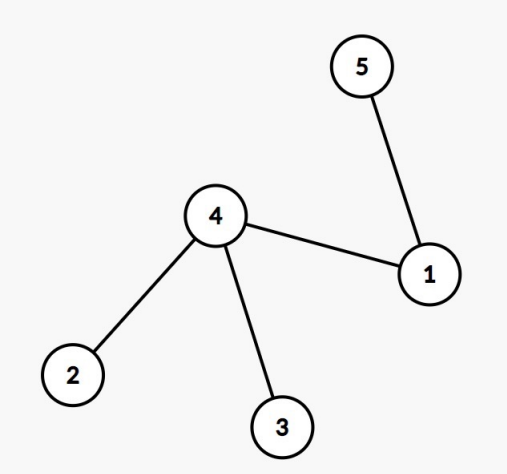


Таблица попарных расстояний между вершинами выглядит так:

	1	2	3	4	5
1	0	2	2	1	1
2	2	0	2	1	3
3	2	2	0	1	3
4	1	1	1	0	2
5	1	3	3	2	0

- Если сделать операцию, в которой $d = [0, 0, 0, 0, 0]$, то ни одна лампочка не загорится, потому что $dist(i, j) > 0$ при всех $i \neq j$.
- Если сделать операцию, в которой $d = [1, 1, 2, 0, 2]$, то загорятся все лампочки, кроме лампочки, соответсвующей вершине дерева-шифра с номером 3. Например, лампочка, соответсвующая вершине с номером 1 загорится, потому что $dist(1, 5) = 1 \leqslant 2 = d_5$.
- Если сделать операцию, в которой $d = [0, 0, 0, 1, 0]$, то загорятся все лампочки, кроме лампочек, соответсвующих вершинам дерева-шифра с номерами 4 и 5.
- Если сделать операцию, в которой $d = [0, 1, 0, 0, 1]$, то загорятся только лампочки, соответсвующие вершинам дерева-шифра с номерами 1 и 4.

Задача C. Сортировка стеком

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Предположим, дан массив a, стек s (изначально пустой) и массив b (также пустой).
Можно производить следующие операции, пока a или s непусты:

- Взять первый элемент a, положить его на верхушку s и удалить из a (если a не пустой);
- Взять элемент с верхушки s, добавить его в конец массива b и удалить из s (если s не пустой).

Операции можно осуществлять в произвольном порядке.

Если существует способ проделать операции в таком порядке, что массив b отсортирован в неубывающем порядке в конце процесса, то назовем массив a *стек-сортируемым*.

Например, $[3, 1, 2]$ — *стек-сортируемый*, потому что b будет отсортированным после следующих операций:

1. Удалить 3 из a и положить на верхушку s;
2. Удалить 1 из a и положить на верхушку s;
3. Удалить 1 из s и добавить в конец b;
4. Удалить 2 из a и положить на верхушку s;
5. Удалить 2 из s и добавить в конец b;
6. Удалить 3 из s и добавить в конец b.

После всех операций $b = [1, 2, 3]$, поэтому $[3, 1, 2]$ — *стек-сортируемый*. $[2, 3, 1]$ — не *стек-сортируемый*.

Заданы первые k элементов некоторой перестановки p размера n (напоминаем, что перестановка размера n — это такой массив размера n, где каждое число от 1 до n встречается ровно один раз). Необходимо восстановить оставшиеся $n - k$ элементов этой перестановки, так чтобы она стала *стек-сортируемой*. Если существует несколько ответов, то выберите такой, что p лексикографически максимальна (массив q лексикографически больше массива p тогда и только тогда, когда существует такое целое число k, что для всех $i < k$ $q_i = p_i$, and $q_k > p_k$). **Запрещено как-то переставлять или менять первые k элементов**.

Выведите лексикографически максимальную перестановку p, которую можно получить.

Если ответа не существует, то выведите `-1`.

Формат входных данных

В первой строке записаны два целых числа n и k ($2 \leqslant n \leqslant 200000$, $1 \leqslant k < n$) — размер желаемой перестановки и количество элементов, которые уже заданы.

Во второй строке записаны k чисел $p_1, p_2, \ldots, p_k$ ($1 \leqslant p_i \leqslant n$) — первые k элементов перестановки p. Эти числа попарно различны.

Формат выходных данных

Если возможно восстановить *стек-сортируемую* перестановку p размера n такую, что первые k элементов перестановки p совпадают с заданными, выведите максимальную лексикографически из таковых.

В противном случае выведите `-1`.

Примеры

стандартный ввод	стандартный вывод
5 3 3 2 1	3 2 1 5 4
5 3 2 3 1	-1
5 1 3	3 2 1 5 4
5 2 3 4	-1

Задача D. Железнодорожная система

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Это интерактивная задача.

Под непосредственным наблюдением Kanako и Moriya железнодорожная система Gensokyo наконец-то завершена. GSKR (Gensokyo Railways) состоит из n станций с m двунаправленными дорожками, соединяющими их. i-я дорожка имеет длину l_i ($1 \leqslant l_i \leqslant 10^6$). Из-за бюджетных ограничений железнодорожная система **может быть не связной**, хотя между двумя станциями может быть более одного пути.

Значение железнодорожной системы определяется как общая длина всех ее дорожек. *Максимальная (или минимальная) мощность* железнодорожной системы определяется как максимальное (или минимальное) значение среди всех полных остовных лесов.

Полный остовный лес графа — это остовный лес с той же связностью, что и данный граф.

У Kanako есть симулятор, способный обрабатывать не более $2m$ запросов. На вход симулятора подается строка s длины m, состоящая из символов `0` и/или `1`. Симулятор оставит i-ю дорожку, если $s_i = $ `1`. Затем устройство сообщит Kanako максимальную мощность системы в смоделированном состоянии.

Kanako хочет узнать минимальную мощность системы со всеми дорожками с помощью симулятора.

Структура железнодорожной системы определяется заранее. Другими словами, интерактор не адаптивен.

Формат входных данных

Первая и единственная строка ввода содержит два целых числа n, m ($2 \leqslant n \leqslant 200$, $1 \leqslant m \leqslant 500$) — количество станций и путей.

Протокол взаимодействия

Начните взаимодействие, прочитав n, m.

Чтобы сделать запрос, выведите «`?` s» (без кавычек, s — это строка длины m, состоящая из символов `0` и/или `1`). Затем вы должны прочитать наш ответ из стандартного ввода — максимальная мощность системы в смоделированном состоянии.

Если ваша программа сделала недопустимый запрос или закончились попытки, интерактор немедленно завершится, и ваша программа получит вердикт `Неверный ответ`.

Чтобы дать окончательный ответ, выведите «`!` L» (без кавычек, L — это минимальная мощность системы со всеми дорожками). Обратите внимание, что предоставление этого ответа не считается как один из $2m$ запросов.

После печати запроса не забудьте вывести конец строки и очистить вывод. В противном случае вы получите `превышение лимита бездействия`. Для этого используйте:

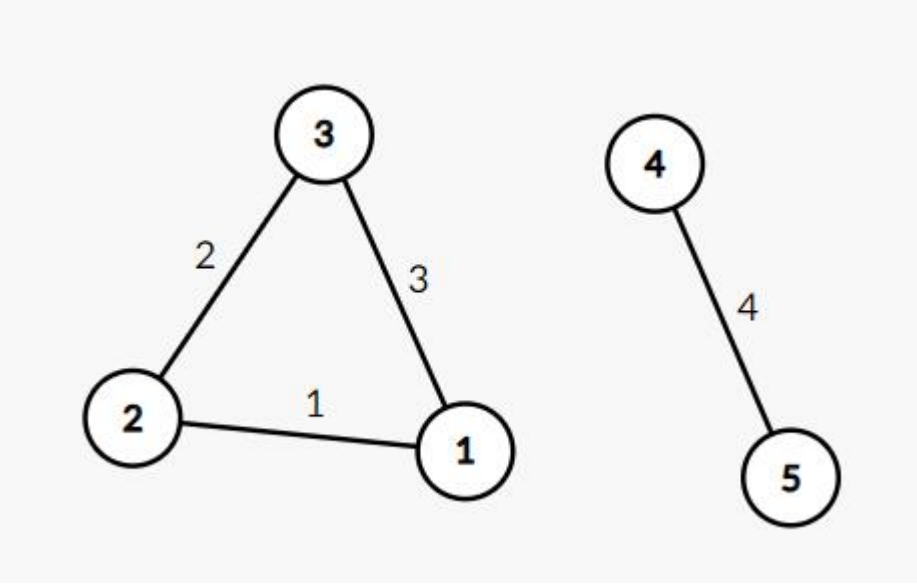
- `fflush(stdout)` или `cout.flush()` в C++;
- `System.out.flush()` в Java;
- `flush(output)` в Паскале;
- `stdout.flush()` в Python;
- см. документацию для других языков.

Пример

стандартный ввод	стандартный вывод
5 4 0 5 9 7	? 0000 ? 1110 ? 1111 ? 1101 ! 7

Замечание

Пример из условия, в котором $l_i = i$:



Задача F. Разделение рёбер

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Вам дан связный, неориентированный и невзвешенный граф с n вершинами и m рёбрами. Обратите внимание на **ограничение на количество рёбер**: $m \leqslant n + 2$.

Давайте скажем, что мы красим некоторые рёбра в красный и оставшиеся рёбра в синий. Теперь рассмотрим только красные рёбра и посчитаем количество компонент связности в графе. Пусть это значение равно c_1. Аналогично, рассмотрим только синие рёбра и посчитаем количество компонент связности в графе. Пусть это значение равно c_2.

Найдите распределение цветов по рёбрам такое, что величина $c_1 + c_2$ **минимальна**.

Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. Первая строка содержит единственное целое число t ($1 \leqslant t \leqslant 10^5$) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа n и m ($2 \leqslant n \leqslant 2 \cdot 10^5$; $n - 1 \leqslant m \leqslant \min\left(n + 2, \frac{n \cdot (n-1)}{2}\right)$) — количество вершин и рёбер в графе соответственно.

Затем следуют m строк. i-я строка содержит два целых числа u_i и v_i ($1 \leqslant u_i, v_i \leqslant n$, $u_i \neq v_i$) обозначающая, что i-е ребро соединяет вершины u_i и v_i. Гарантируется, что в графе нет петель и кратных рёбер. Также гарантируется, что граф связен.

Гарантируется, что сумма n по всем наборам входных данных не превосходит 10^6. Гарантируется, что сумма m по всем наборам входных данных не превосходит $2 \cdot 10^6$.

Формат выходных данных

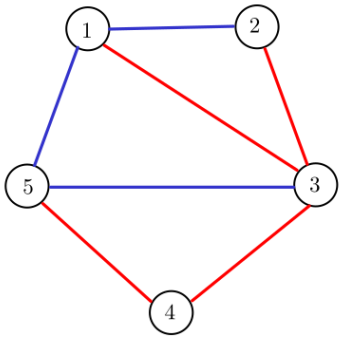
Для каждого набора входных данных выведите бинарную строку длины m. i-й символ строки должен быть `1`, если i-е ребро должно иметь красный цвет, и `0`, если оно должно иметь синий цвет. Если существует несколько способов распределить цвета, чтобы получить минимальную величину, вы можете вывести любой.

Пример

стандартный ввод	стандартный вывод
4 5 7 1 2 2 3 3 4 4 5 5 1 1 3 3 5 4 4 1 2 2 3 1 4 3 4 6 7 1 2 1 3 3 4 4 5 1 4 5 6 6 2 2 1 1 2	1110001 1101 1011011 1

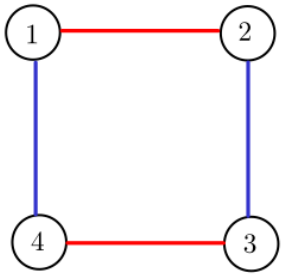
Замечание

- Граф в первом наборе входных данных следующий:



$c_1 + c_2 = 1 + 2 = 3$

- Граф во втором наборе входных данных следующий:



$c_1 + c_2 = 2 + 2 = 4$

Задача G. AND-MEX-прогулка

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	3 секунды
Ограничение по памяти:	256 мегабайт

Вам дан неориентированный связный граф из n вершин и m взвешенных ребер. *Прогулкой* от вершины u до вершины v называется последовательность вершин $p_1, p_2, \ldots, p_k$ (не обязательно различных), начинающаяся в вершине u и заканчивающаяся в вершине v, такая что p_i и p_{i+1} соединены ребром для всех $1 \leqslant i < k$.

Определим *длину* прогулки следующим образом: выпишем веса ребер в порядке следования в массив. Затем выпишем побитовое И каждого непустого префикса этого массива. *Длиной* прогулки называется MEX этих значений.

Более формально, пусть веса ребер это $[w_1, w_2, \ldots, w_{k-1}]$, где w_i — вес ребра между p_i и p_{i+1}. Тогда *длина* прогулки равна $\mathrm{MEX}(\{w_1,\ w_1 \& w_2,\ \ldots,\ w_1 \& w_2 \& \ldots \& w_{k-1}\})$, где & обозначает операцию побитового И.

Вы должны обработать q запросов вида `u v`. Для каждого запроса найдите **минимально** возможную длину пути из u в v.

MEX (минимальное отсутствующее) множества чисел — наименьшее неотрицательное число, отсутствующее в этом множестве. Например:

- MEX множества $\{2, 1\}$ равен 0, т. к. 0 отсутствует в множестве.
- MEX множества $\{3, 1, 0\}$ равен 2, т. к. 0 и 1 присутствуют в множестве, а 2 — нет.
- MEX множества $\{0, 3, 1, 2\}$ равен 4, т. к. 0, 1, 2 и 3 присутствуют в множестве, а 4 отсутствует.

Формат входных данных

Первая строка содержит два целых числа n и m ($2 \leqslant n \leqslant 10^5$; $n - 1 \leqslant m \leqslant \min\left(\frac{n(n-1)}{2}, 10^5\right)$).

Каждая из следующих m строк содержит три целых числа a, b и w ($1 \leqslant a, b \leqslant n$, $a \neq b$; $0 \leqslant w < 2^{30}$), обозначающих неориентированное ребро между вершинами a и b веса w. Гарантируется, что не существует петель и кратных ребер, а также что граф связный.

Следующая строка содержит одно целое число q ($1 \leqslant q \leqslant 10^5$).

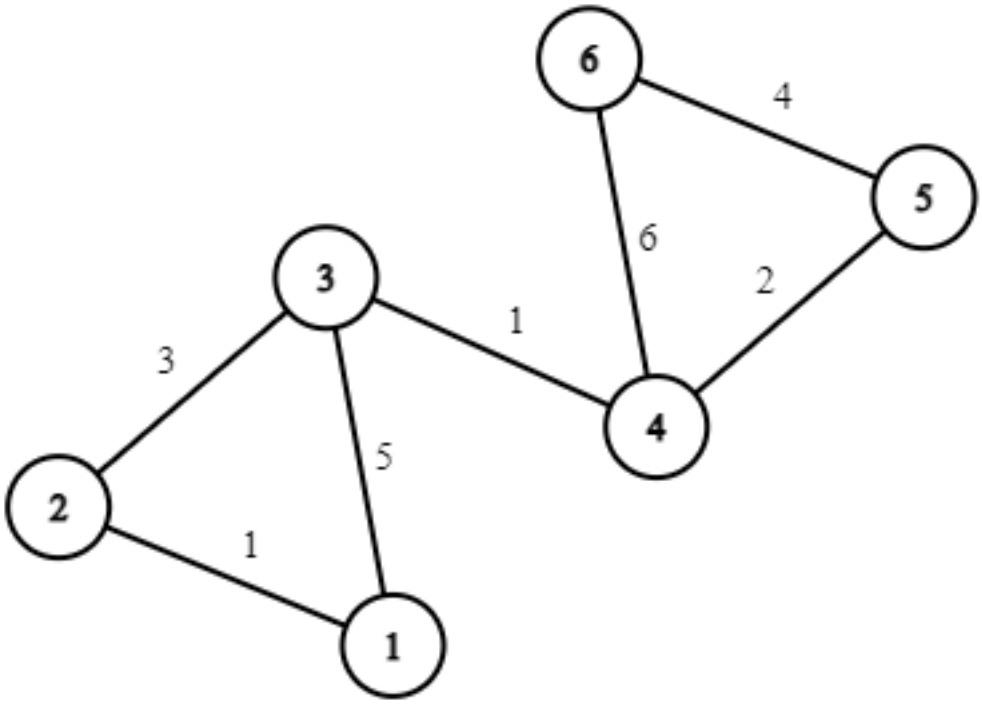
Каждая из следующих q строк содержит два целых числа u и v ($1 \leqslant u, v \leqslant n$, $u \neq v$) — описание запроса.

Формат выходных данных

Для каждого запроса выведите одно целое число — ответ на запрос.

Замечание

Ниже находится пояснение к первому примеру.



Граф из первого примера.

Идна из возможных прогулок в первом запросе:

$$1 \xrightarrow{5} 3 \xrightarrow{3} 2 \xrightarrow{1} 1 \xrightarrow{5} 3 \xrightarrow{1} 4 \xrightarrow{2} 5.$$

Массив весов равен $w = [5, 3, 1, 5, 1, 2]$. Если мы возьмем побитовое И для всех префиксов, мы получим множество $\{5, 1, 0\}$. MEX этого множества равен 2. Нельзя получить прогулку меньшей длины.

Задача H. Развороты

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Вам даны два целых положительных числа x и y. Вы можете совершать с числом x следующую операцию: записать его в двоичном виде без ведущих нулей, приписать справа 0 или 1, затем развернуть двоичную запись и превратить ее в десятичное число, которое окажется новым значением x.

Например:

- 34 можно одной операцией превратить в 81: двоичная запись 34 — это 100010, если приписать справа 1, развернуть и убрать ведущие нули, можно получить 1010001, что является двоичной записью числа 81;
- 34 можно одной операцией превратить в 17: двоичная запись 34 — это 100010, если приписать справа 0, развернуть и убрать ведущие нули, можно получить 10001, что является двоичной записью числа 17;
- 81 можно одной операцией превратить в 69: двоичная запись 81 — это 1010001, если приписать справа 0, развернуть и убрать ведущие нули, можно получить 1000101, что является двоичной записью числа 69;
- 34 можно превратить в 69 за две операции: сначала превратить 34 в 81, а потом превратить 81 в 69.

Ваша задача — выяснить, можно ли превратить x в y за какое-то (возможно, нулевое) число операций.

Формат входных данных

Единственная строка входных данных содержит два целых числа x и y ($1 \leqslant x, y \leqslant 10^{18}$).

Формат выходных данных

Выведите `YES`, если вы можете сделать x равным y, или `NO`, если это невозможно.

Примеры

стандартный ввод	стандартный вывод
3 3	YES
7 4	NO
2 8	NO
34 69	YES
8935891487501725 71487131900013807	YES

Замечание

В первом примере не надо совершать никаких действий.

Четвертый пример разобран в условии задачи.

Задача I. Xoractive

Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Айдос загадал последовательность a из n попарно различных неотрицательных целых чисел $a_1, a_2, \ldots, a_n$.

Ваша задача — восстановить всю загаданную последовательность.

Для этого разрешается задавать запросы двух типов:

- узнать значение одного элемента последовательности;
- для заданного набора индексов получить отсортированный список всех попарных значений `xor` элементов на этих позициях.

Для индексов $i_1, i_2, \ldots, i_k$ список попарных значений `xor` состоит из чисел $a_{i_x} \oplus a_{i_y}$, где $1 \leqslant x \leqslant k$ и $1 \leqslant y \leqslant k$.

Обратите внимание, что рассматриваются упорядоченные пары. Поэтому ответ содержит ровно k^2 чисел.

В частности, для каждого выбранного индекса в ответе присутствует ноль, так как $a_{i_x} \oplus a_{i_x} = 0$.

Например, если загадана последовательность $[1, 5, 6, 3]$, то для индексов 3 и 4 будет получен список $[0, 0, 5, 5]$.

Действительно, $a_3 \oplus a_3 = 0$, $a_4 \oplus a_4 = 0$, $a_3 \oplus a_4 = 5$ и $a_4 \oplus a_3 = 5$.

Последовательность a фиксируется до начала взаимодействия и не изменяется в зависимости от запросов решения.

Формат входных данных

В начале взаимодействия интерактор выводит одно целое число n — длину загаданной последовательности, где $2 \leqslant n \leqslant 100$.

Загаданная последовательность состоит из n попарно различных целых чисел $a_1, a_2, \ldots, a_n$, причём $0 \leqslant a_i \leqslant 10^9$.

Значения элементов последовательности решению не сообщаются.

Формат выходных данных

Когда вы восстановили загаданную последовательность, выведите строку

`3` a_1 a_2 $\ldots$ a_n

После вывода финального ответа программа должна немедленно завершиться.

Финальный ответ не считается запросом.

Протокол взаимодействия

Для взаимодействия с интерактором можно использовать запросы двух типов.

Чтобы узнать значение элемента a_i, выведите строку

`1` i

где $1 \leqslant i \leqslant n$.

В ответ интерактор выведет одно целое число — значение a_i.

Чтобы получить попарные значения `xor`, выведите строку

`2` k i_1 i_2 $\ldots$ i_k

где $1 \leqslant k \leqslant n$, все индексы $i_1, i_2, \ldots, i_k$ попарно различны и для каждого из них выполняется $1 \leqslant i_j \leqslant n$.

В ответ интерактор выведет k^2 целых чисел в неубывающем порядке. Эти числа представляют значения $a_{i_x} \oplus a_{i_y}$ для всех $1 \leqslant x \leqslant k$ и $1 \leqslant y \leqslant k$.

Суммарное количество запросов типов 1 и 2 не должно превышать 15.

После каждого запроса необходимо вывести перевод строки и очистить буфер вывода. Например, в C++ можно использовать `endl` или вызов `cout.flush()`.

Если запрос имеет неверный формат, содержит недопустимые или повторяющиеся индексы либо превышен лимит запросов, решение получит вердикт **Wrong Answer**.

Замечание

Под операцией `xor` подразумевается побитовое исключающее ИЛИ.

Пусть загадана последовательность $a = [1, 5, 6, 3]$. Пример возможного взаимодействия:

Интерактор выводит:

```
4
```

Решение выводит запрос:

```
1 2
```

Интерактор отвечает:

```
5
```

Решение выводит запрос:

```
2 3 1 2 3
```

Интерактор отвечает:

```
0 0 0 3 3 4 4 7 7
```

Решение выводит запрос:

```
1 3
```

Интерактор отвечает:

```
6
```

Решение выводит запрос:

```
2 2 4 2
```

Интерактор отвечает:

```
0 0 6 6
```

Решение выводит запрос:

```
2 1 2
```

Интерактор отвечает:

```
0
```

После восстановления последовательности решение выводит финальный ответ:

```
3 1 5 6 3
```

Задача J. Сортировка кучей

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Хорошо известный алгоритм, называемый пирамидальной сортировкой (heapsort), представляет собой детерминированный алгоритм сортировки, работающий за время $O(n \log n)$ и использующий $O(1)$ дополнительной памяти. Опишем сортировку по возрастанию массива различных целых чисел.

Алгоритм состоит из двух фаз. На первой фазе, называемой построением пирамиды (heapification), массив целых чисел, подлежащий сортировке, преобразуется в пирамиду (heap). Массив $a[1 \dots n]$ целых чисел называется пирамидой, если для всех $1 \leqslant i \leqslant n$ выполняются следующие условия пирамиды:

- если $2i \leqslant n$, то $a[i] > a[2i]$;
- если $2i + 1 \leqslant n$, то $a[i] > a[2i + 1]$.

Мы можем интерпретировать массив как бинарное дерево, считая детьми элемента $a[i]$ элементы $a[2i]$ и $a[2i + 1]$. В этом случае родителем $a[i]$ является $a[i \text{ div } 2]$, где $i \text{ div } 2 = \lfloor i/2 \rfloor$. В терминах деревьев свойство быть пирамидой означает, что для каждого узла его значение больше значений его детей.

Во второй фазе пирамида превращается в отсортированный массив. Из-за условия пирамиды наибольший элемент в массиве, преобразованном в пирамиду, находится в $a[1]$. Поменяем его местами с $a[n]$, теперь наибольший элемент массива находится на своей правильной позиции в отсортированном массиве. Это называется извлечением максимума (extract-max).

Теперь рассмотрим часть массива $a[1 \dots n - 1]$. Она может не быть пирамидой, потому что условие пирамиды может нарушаться для $i = 1$. Если это так (то есть либо $a[2]$, либо $a[3]$, либо оба больше, чем $a[1]$), поменяем местами $a[1]$ и его наибольшего ребенка, восстанавливая условие пирамиды для $i = 1$. Теперь возможно, что условие пирамиды нарушается для позиции, которая теперь содержит бывшее значение $a[1]$. Применим ту же процедуру к ней, меняя её с её наибольшим ребенком. Продолжая так, мы преобразуем весь массив $a[1 \dots n - 1]$ в пирамиду. Эта процедура называется просеиванием вниз (sifting down). После преобразования части $a[1 \dots n - 1]$ в пирамиду с помощью просеивания мы снова применяем извлечение максимума, помещая второй по величине элемент массива в $a[n - 1]$, и так далее.

Например, посмотрим, как пирамида $a = (5, 4, 2, 1, 3)$ преобразуется в отсортированный массив. Выполним первое извлечение максимума. После этого массив превращается в $(3, 4, 2, 1, 5)$. Условие пирамиды нарушается для $a[1] = 3$, так как его ребенок $a[2] = 4$ больше него. Выполним просеивание вниз, обменяв $a[1]$ и $a[2]$. Теперь массив имеет вид $(4, 3, 2, 1, 5)$. Условие пирамиды выполнено для всех элементов, поэтому просеивание закончено. Выполним извлечение максимума снова. Теперь массив превращается в $(1, 3, 2, 4, 5)$. Опять условие пирамиды нарушается для $a[1]$; обменяв его с наибольшим ребенком, получаем массив $(3, 1, 2, 4, 5)$, который является правильной пирамидой. Итак, выполняем извлечение максимума и получаем $(2, 1, 3, 4, 5)$. На этот раз условие пирамиды выполнено для всех элементов, поэтому выполняем извлечение максимума, получая $(1, 2, 3, 4, 5)$. Начальная часть массива является пирамидой, и последнее извлечение максимума окончательно дает $(1, 2, 3, 4, 5)$.

Известно, что построение пирамиды может быть выполнено за $O(n)$ времени. Следовательно, самой трудоемкой операцией в алгоритме пирамидальной сортировки является просеивание, которое занимает $O(n \log n)$ времени.

В этой задаче вам нужно найти массив, преобразованный в пирамиду, содержащий различные числа от 1 до n, такой, что при его преобразовании в отсортированный массив общее количество обменов во всех операциях просеивания было максимально возможным. В приведенном выше примере количество обменов равно $1 + 1 + 0 + 0 + 0 = 2$, что не является максимальным. Массив $(5, 4, 3, 2, 1)$ дает максимальное количество обменов, равное 4, для $n = 5$.

Формат входных данных

В единственной строке содержится целое число n ($1 \leqslant n \leqslant 50000$).

Формат выходных данных

Выведите массив из n различных целых чисел от 1 до n, являющийся кучей, такой, что при преобразовании его в отсортированный массив общее число обменов при операциях просеивания максимально возможное. Числа разделяйте пробелами.

Пример

стандартный ввод	стандартный вывод
6	6 5 3 2 4 1