Программирование: Пермутации RAM

Предыстория

В свободное от работы в Lemon AI время Янис торгует модулями оперативной памяти, чтобы заработать дополнительный доход.

Однажды Янис нашел невероятную сделку на набор высокопроизводительных модулей оперативной памяти и решил модернизировать сервер компании. Каждый модуль имеет уникальный серийный номер (от 1 до n).

Однако есть одна загвоздка: сервер имеет особую систему конфигурации BIOS. Для оптимальной производительности модули оперативной памяти должны быть установлены в определенном порядке, определяемом контроллером памяти материнской платы, но инструкция пропала!

К счастью, BIOS имеет диагностический режим: Янис может протестировать расположение модулей памяти, и система выдаст ему оценку, указывающую, в скольких слотах установлены правильные модули.

Будучи опытным программистом, Янис понимает, что это задача **комбинаторной оптимизации** — поиск правильного расположения в дискретном пространстве.

> ***Комбинаторная оптимизация*** *— область, посвященная поиску оптимальных решений в дискретных пространствах (например, перестановки, комбинации, графы). Такие задачи чрезвычайно сложны, поскольку количество возможных комбинаций растет экспоненциально с увеличением количества элементов. Их часто решают с помощью с помощью нейросетей или обучения с подкреплением, хотя умные алгоритмы могут эффективно решать конкретные случаи.*

Перестановка — это определенный порядок заданных элементов. Например, возможные перестановки чисел 1, 2, 3:

```
- 1 - 2 - 3
- 1 - 3 - 2
- 2 - 1 - 3
- 2 - 3 - 1
- 3 - 1 - 2
- 3 - 2 - 1
```

Ваша задача — помочь Янису эффективно определить правильную конфигурацию оперативной памяти!

Задача

Это интерактивная задача.

BIOS сервера имеет секретную оптимальную конфигурацию C из n модулей памяти ($1 \leq n \leq 31$).

Каждый модуль обозначен уникальным серийным номером ($1 \leq \mathrm{id} \leq n$).

Цель Яниса — найти правильную конфигурацию с помощью диагностического режима BIOS.

В каждом запросе Янис тестирует комбинацию P и получает оценку S.

Оценка S рассчитывается следующим образом:

- Для каждой позиции i ($1 \le i \le n$) мы добавляем 2^i к S, если планка в позиции i в P совпадает с планкой в позиции i в C.

Например, если:

- Оптимальная конфигурация BIOS $C = [2, 1, 3, 4]$
- Тестовая комбинация Яниса $P = [4, 1, 3, 2]$

Тогда оценка $S = 2^2 + 2^3 = 12$, потому что палочки в позициях 2 и 3 совпадают (у обеих палочка 1 находится в позиции 2, а палочка 3 — в позиции 3).

Ваша задача — помочь Янису найти конфигурацию C за не более чем n запросов.

Входные данные

Первая строка содержит целое число n ($1 \le n \le 31$).

Взаимодействие

Вы можете протестировать расположение с помощью диагностики BIOS, запрашивая `? p_1 p_2 ... p_n` через данную функцию `query`, где:

- `p_1, p_2, ..., p_n`: предлагаемое расположение P из n палочек

BIOS ответит целым числом S.

Вывод

После того, как вы определили оптимальную конфигурацию, запросите `! c_1 c_2 ... c_n`, где $c_1, c_2, \ldots, c_n$ — конфигурация C.

Ограничения

Ограничения на входные данные:

- количество тестовых случаев $\le 2 \times 10^4$
- сумма n по всем тестовым случаям $\le 1 \times 10^6$

Ограничение по времени выполнения — 20 секунд. Ограничение по памяти — 512 МБ.

Пример

Возьмем $n = 3$, $C = [2, 1, 3]$.
Мы зададим два запроса, чтобы определить C, а затем выведем результат.

Входные данные:

```
3
```

Взаимодействие:

```
? 1 2 3 8 ? 2 3 1 2 ! 2 1 3
```

Оценивание

Ваше решение будет проверено на нескольких скрытых тестах.

Оценка = пройденные тесты / общее количество тестов