Механическая игрушка

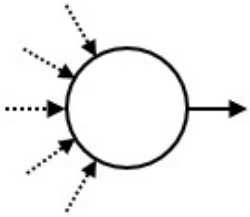
Механическая игрушка — это игрушка, которая автоматически повторяет определенную последовательность движений. В Японии механические игрушки популярны с древнейших времен.

Движения механической игрушки управляются **схемой**, которая состоит из **устройств**. Устройства соединены трубами. У каждого устройства есть любое количество (возможно, ноль) **входов** и один или два **выхода**. Каждая труба соединяет выход некоторого устройства со входом того же самого или другого устройства. К каждому входу подключена ровно одна труба, и к каждому выходу подключена ровно одна труба.

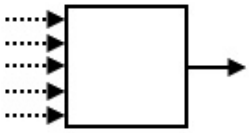
Чтобы описать, как работает схема, рассмотрим **шарик**, который в каждый момент находится в одном из устройств. Шарик перемещается между устройствами. На каждом шаге шарик покидает устройство, в котором он находится, через один из его выходов, перемещается по трубе и попадает в устройство на конце этой трубы через соответствующий вход.

Есть три типа устройств: **источник**, **триггер** и **переключатель**. Есть ровно один источник, M триггеров и S переключателей (S может быть равно нулю). Вам необходимо выбрать значение S. У каждого устройства есть уникальный серийный номер.

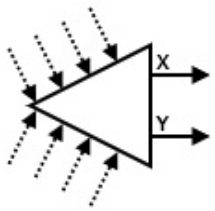
Источник — это устройство, в котором в начале находится шарик. У него ровно один выход. Его серийный номер равен 0.



Триггер — это устройство ровно с одним выходом. Серийные номера триггеров — целые числа от 1 до M.



У каждого переключателя два выхода, которые обозначаются 'X' и 'Y'. Каждый переключатель находится в одном из двух **состояний**: 'X' или 'Y'. После того, как шарик попадает в переключатель, он покидает его через выход, равный текущему состоянию переключателя. После этого переключатель меняет своё состояние на противоположное. В начале каждый переключатель находится в состоянии 'X'. Серийные номера переключателей — целые числа от -1 до $-S$.



Задано количество триггеров M. Также задана последовательность A длины N, каждый элемент которой — серийный номер триггера. Серийный номер триггера может встречаться в последовательности A несколько раз (возможно, ноль). Вам необходимо построить схему, которая удовлетворяет следующим требованиям:

- Начав движение в источнике, шарик возвращается в источник после нескольких перемещений.
- Когда шарик в первый раз после начала движения возвращается в источник, каждый переключатель находится в состоянии 'X'.
- Перед тем как шарик в первый раз после начала движения вернётся в источник, он побывает в триггерах ровно N раз. При этом он побывает в триггерах с серийными номерами $A_0, A_1, \ldots, A_{N-1}$ в этом порядке.
- Пусть P равно общему количеству изменений состояний переключателей в процессе перемещения шарика перед тем, как шарик впервые вернётся в источник. Значение P не должно превышать $20\,000\,000$.

При этом вы хотите использовать не слишком много переключателей.

Детали реализации

Вам необходимо реализовать следующую процедуру.

```
create_circuit(int M, int[] A)
```

- `M`: количество триггеров.
- `A`: массив длины N, который задает серийные номера триггеров в том порядке, в котором шарик должен в них побывать.
- Эта процедура будет вызвана ровно один раз.
- Обратите внимание, что число N — это длина массива `A`, которую можно узнать методом, описанным в памятке о деталях реализации.

Ваша программа должна вызвать следующую процедуру.

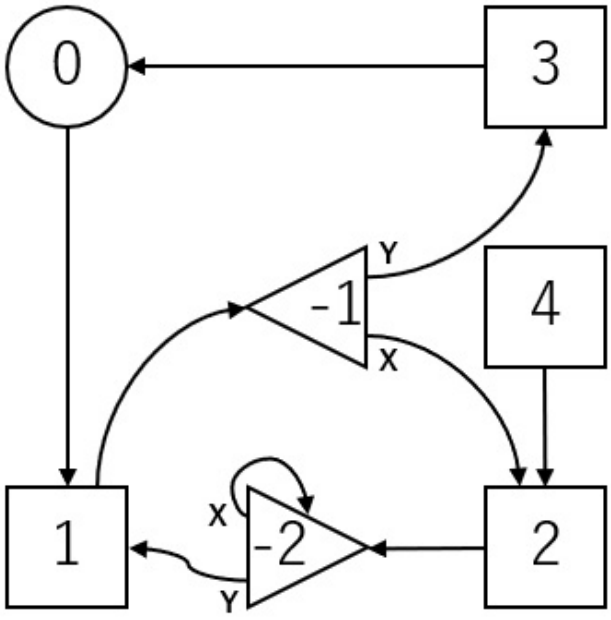
```
answer(int[] C, int[] X, int[] Y)
```

- C: массив длины $M+1$. Выход устройства с серийным номером i $(0 \leq i \leq M)$ соединён трубой с входом устройства с серийным номером C[i].
- X, Y: массивы одинаковой длины. Длина S каждого из этих массивов — количество переключателей. Для переключателя с серийным номером $-j$ $(1 \leq j \leq S)$ его выход 'X' соединён трубой с входом устройства с серийным номером X[j-1], а его выход 'Y' соединён трубой с входом устройства с серийным номером Y[j-1].
- Каждый элемент массивов C, X и Y должен быть целым числом в диапазоне от $-S$ до M, включительно.
- S должно быть не более $400\,000$.
- Процедура должна быть вызвана ровно один раз.
- Схема, описываемая массивами C, X и Y, должна удовлетворять требованиям, описанным в условии задачи.

Если одно из условий, перечисленных выше, не выполняется, ваша программа получает вердикт **Wrong Answer**. Иначе ваша программа получает вердикт **Accepted** и ваши баллы вычисляются по формуле в зависимости от значения S (смотрите раздел "Подзадачи").

Пример

Пусть $M = 4$, $N = 4$ и $A = [1, 2, 1, 3]$. Проверяющий модуль (grader) делает вызов `create_circuit(4, [1, 2, 1, 3])`.



Приведенный выше рисунок показывает схему, которая описывается вызовом `answer([1, -1, -2, 0, 2], [2, -2], [3, 1])`. Числа на рисунке — серийные номера устройств. Используется два переключателя, поэтому $S = 2$. В начале оба

переключателя -1 и -2 находятся в состоянии 'X'. Шарик перемещается следующим образом:

$$0 \longrightarrow 1 \longrightarrow -1 \xrightarrow{\text{X}} 2 \longrightarrow -2 \xrightarrow{\text{X}} -2 \xrightarrow{\text{Y}} 1 \longrightarrow -1 \xrightarrow{\text{Y}} 3 \longrightarrow 0$$

- Когда шарик впервые попадает в переключатель -1, тот находится в состоянии 'X'. Поэтому шарик перемещается в триггер 2. Затем состояние переключателя -1 изменяется на 'Y'.
- Когда шарик попадает в переключатель -1 во второй раз, тот находится в состоянии 'Y'. Поэтому шарик перемещается в триггер 3. Затем состояние переключателя -1 изменяется на 'X'.

Шарик впервые возвращается в источник, побывав в триггерах $1, 2, 1, 3$. Оба переключателя -1 и -2 находятся в состоянии 'X'. Значение P равно 4. Следовательно, эта схема удовлетворяет всем условиям.

Файл `sample-01-in.txt` в приложенном к задаче zip-архиве соответствует этому примеру. Другие примеры ввода также доступны в архиве.

Ограничения

- $1 \leq M \leq 100\,000$
- $1 \leq N \leq 200\,000$
- $1 \leq A_k \leq M$ $(0 \leq k \leq N-1)$

Подзадачи

1. (2 балла) Каждое целое число i $(1 \leq i \leq M)$ встречается в последовательности $A_0, A_1, \ldots, A_{N-1}$ не более одного раза.
2. (4 балла) Каждое целое число i $(1 \leq i \leq M)$ встречается в последовательности $A_0, A_1, \ldots, A_{N-1}$ не более двух раз.
3. (10 баллов) Каждое целое число i $(1 \leq i \leq M)$ встречается в последовательности $A_0, A_1, \ldots, A_{N-1}$ не более четырех раз.
4. (10 баллов) $N = 16$
5. (18 баллов) $M = 1$
6. (56 баллов) Нет дополнительных ограничений

Для каждого теста, если ваша программа получает на нём вердикт **`Accepted`**, ваши баллы за этот тест вычисляются по следующим правилам в зависимости от S:

- Если $S \leq N + \log_2 N$, вы получаете полный балл за этот тест.
- Для каждого теста в подзадачах 5 и 6, если $N + \log_2 N < S \leq 2N$, вы получаете частичные баллы. Баллы за тест равны значению $0.5 + 0.4 \times \left(\dfrac{2N - S}{N - \log_2 N}\right)^2$, умноженному на балл для этой подзадачи.

- В противном случае баллы равны 0.

Баллы за каждую подзадачу равны минимуму баллов за тест в этой подзадаче.

Пример проверяющего модуля

Пример проверяющего модуля читает входные данные из стандартного потока ввода в следующем формате:

- строка 1: M N
- строка 2: A_0 A_1 $\ldots$ A_{N-1}

В результате исполнения проверяющий модуль создаёт три файла.

Файл `out.txt` содержит вывод вашей программы в следующем формате.

- строка 1: S
- строка $2 + i$ ($0 \le i \le M$): `C[i]`
- строка $2 + M + j$ ($1 \le j \le S$): `X[j - 1] Y[j - 1]`

Далее проверяющий модуль симулирует перемещения шарика. Он выводит последовательность серийных номеров устройств, в которых побывал шарик, в файл `log.txt`.

Наконец, проверяющий модуль печатает на стандартный поток вывода результат оценки вашего решения.

- Если вердикт вашей программы **`Accepted`**, выводятся значения S и P в следующем формате `Accepted: S P`.

- Если вердикт вашей программы **`Wrong Answer`**, выводится сообщение `Wrong Answer: MSG`, где `MSG` — одно из:

  - `answered not exactly once`: Процедура `answer` вызывана более одного раза.
  - `wrong array length`: Длина `C` не равна $M + 1$ или длины `X` и `Y` различны.
  - `over 400000 switches`: S больше $400\,000$.
  - `wrong serial number`: Массив `C`, `X` или `Y` содержит одно или несколько значений, которые меньше $-S$ или больше M.
  - `over 20000000 inversions`: В процессе перемещений шарик не возвращается в источник после $20\,000\,000$ изменений состояний переключателей.
  - `state 'Y'`: Хотя бы один переключатель находится в состоянии 'Y' после первого возвращения шарика в источник.
  - `wrong motion`: Триггеры, посещенные шариком, не образуют последовательность A.

Если ваша программа получает вердикт `Wrong Answer`, файлы `out.txt` и/или `log.txt` могут быть не созданы.