Mechanical Doll

Un muñeco mecánico es un muñeco que repite automáticamente una secuencia de movimientos. En Japón, los muñecos mecánicos han sido creados desde tiempos antiguos.

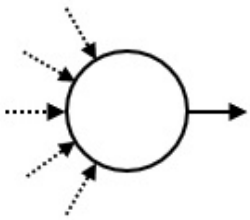
El movimiento de un muñeco mecánico es controlado por un **circuito** que está formado por varios **componentes**.

Los componentes están conectados por tubos. Cada componente tiene varias **entradas** (posiblemente cero) y una o dos **salidas**. Cada componente puede tener cualquier cantidad de entradas. Cada tubo conecta la salida de un componente con la entrada de otro componente (puede ser el mismo componente). Sólo se puede contectar un tubo a cada salida y a cada entrada.

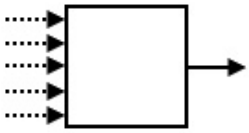
Para describir cómo un muñeco realiza sus movimientos, considera una **bola** que es colocada en alguno de los componentes. La bola se desplaza a través del circuito. En cada componente por el que pasa, deja el componente por una de sus salidas y se desplaza a través del tubo conectado a esa salida hasta entrar al componente que conecta el final del tubo.

Existen tres tipos de componentes, **origen**, **disparador** e **interruptor**. Existe exactamente un origen, M disparadores y S interruptores (S puede ser cero). Tú tienes que decidir el valor de S. Cada componente tiene un número de identificación distinto.

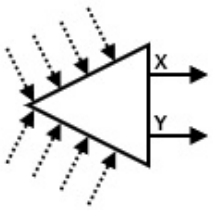
El origen es el componente donde la bola es colocada inicialmente. El origen sólo tiene una salida y su número de identificación es 0.



Un disparador es el componente que provoca que el muñeco realice un movimiento cuando la bola entra a ese componente. Los disparadores siempre tienen exactamente una salida. El número de identificación de los disparadores va de 1 hasta M inclusive.



Cada interruptor tiene dos salidas denotadas como 'X' y 'Y'. El **estado** del switch puede ser 'X' o 'Y'. Cuando la bola entra a un interruptor, deja el interruptor usando la salida que especifica el estado del interruptor en ese momento. El estado del interruptor se cambia automáticamente a su otro estado en el momento que la bola abandona el interruptor. Inicialmente, el estado de todos los interruptores es 'X'. El número de identificación de los interruptores va de -1 a $-S$ inclusive.



Conoces la cantidad de interruptores M. También conoces una secuentia A de longitud N, donde cada elemento es el número de identificación de un disparador. Cada disparador puede aparecer varias veces (incluso cero) en la secuencia A. Tu tarea es crear un circuito que satisfaga las siguientes condiciones:

- La bola debe regresar siempre al origen después de alguna cantidad de pasos.
- Cuando la bola regrese al origen, todos los interruptores deben estar en el estado 'X'.
- La bola debe de regresar al origen después de pasar por exactamente N disparadores. La bola debe pasar por los N disparadores que especifica A en el orden que se encuentran en A.
- Sea P el número de veces que la bola cambió de estado algún interruptor antes de regresar al origen. El valor de P no debe exceder $20\,000\,000$.

Adicionalmente, te interesa reducir la cantidad de interruptores que uses en el circuito.

Detalles de implementación

Debes implementar el siguiente procedimiento.

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

- `M`: el núemero de disparadores.
- `A`: un arreglo de longitud N, que especifica los números de identificación que la bola debe recorrer consecutivamente.
- Este procedimiento es llamado exactamente una vez.

- Nota que el valor de N es la longitud de `A` y puede ser obtenido como se indica en las notas de implementación.

Tu programa debe llamar el siguiente procedimiento para dar la respuesta.

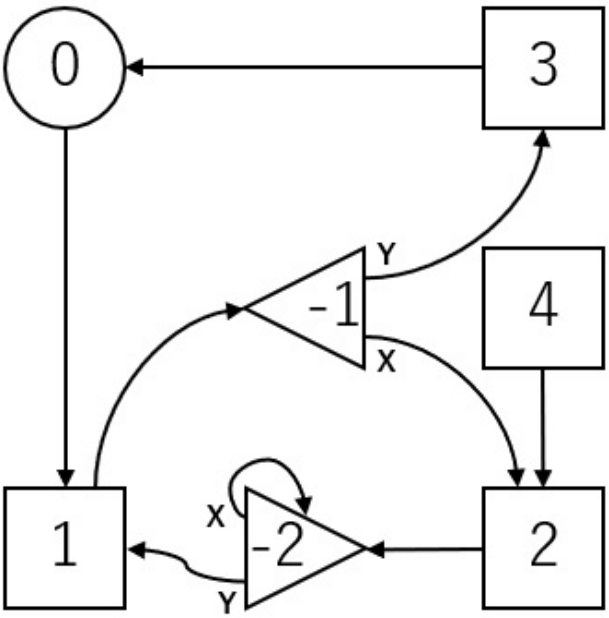
```
answer(int[] C, int[] X, int[] Y)
```

- `C`: un arreglo de longitud $M+1$. La salida del componente (origen o disparador) con número de identificación i ($0 \leq i \leq M$) está conectada al componente con número de identificación `C[i]`.
- `X`, `Y`: arreglos de la misma longitud. La longitud S de estos arreglos es la cantidad de interruptores. Para cada interruptor con número de identificación $-j$ ($1 \leq j \leq S$), su salida 'X' está conectada al componente`X[j - 1]` y su salida 'Y' está conectada al componente `Y[j - 1]`.
- Cada elemento `C`, `X`, y `Y` debe ser un entero entre $-S$ y M, inclusive.
- S debe ser a lo más $400\,000$.
- Este procedimiento debe ser llamado exactamente una vez
- El circuito representado por `C`, `X`, y `Y` debe satisfacer las condiciones que pide el problema.

Si alguna de las condiciones no se satisface, tu programa será evaluado como **`Wrong Answer`**. De lo contrario, tu programa será evaluado como **`Accepted`** y tu puntaje será calculado en función de S (ver subtareas).

Ejemplo

Sea $M = 4$, $N = 4$, y $A = [1, 2, 1, 3]$. El evaluador manda llamar `create_circuit(4, [1, 2, 1, 3])`.



La imagen de arriba muestra un circuito especificado por la llamada `answer([1, -1, -2, 0, 2], [2, -2], [3, 1])`. Los números en la imagen son los números de identificación de los componentes.

Dos interruptores son usados, es decir $S = 2$.

Inicialmente, los estados de los interruptores -1 y -2 se encuentran en 'X'.

La bola se desplaza de la siguiente manera:

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

- Cuando la bola entra al interruptor -1, su estado es 'X'. Por lo tanto, la bola se desplaza hacia el disparador 2. El estado del interruptor -1 se cambia a 'Y'.
- Cuando la bola entra al interruptor -1 por segunda vez, su estado es 'Y'. Por lo tanto, la bola se desplaza al disparador 3. El estado del interruptor -1 se cambia a 'X'.

La bola regresa al origen después de entrar en los disparadores $1, 2, 1, 3$. Los estados de los switches -1 y -2 se encuentran en 'X' cuando la bola regresa al origen. El valor de P es 4. Es decir, el circuito satisface las condiciones del problema.

El archivo `sample-01-in.txt` en el paquete zip adjuntado corresponde a este ejemplo. Otras entradas de ejemplo se encuentran en el paquete.

Restricciones

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

Subtareas

Los puntajes y las restricciones de cada caso de prueba son los siguientes:

1. (2 puntos) Para cada i $(1 \leq i \leq M)$, el entero i aparece a lo más una vez en la secuencia $A_0, A_1, \ldots, A_{N-1}$.
2. (4 puntos) Para cada i $(1 \leq i \leq M)$, el entero i aparece a lo más dos veces en la secuencia $A_0, A_1, \ldots, A_{N-1}$.
3. (10 puntos) Para cada i $(1 \leq i \leq M)$, el entero i aparece a lo más 4 veces en la secuencia $A_0, A_1, \ldots, A_{N-1}$.
4. (10 puntos) $N = 16$
5. (18 puntos) $M = 1$
6. (56 puntos) Sin restricciones adicionales.

Para cada caso de prueba, si tu programa es evaluado como **`Accepted`**, tu puntaje es

calculado en función del valor de S:

- Si $S \leq N + \log_2 N$, obtienes el puntaje completo para cada caso de prueba.
- Para cada caso de prueba de las Subtareas 5 y 6, si $N + \log_2 N < S \leq 2N$, obtienes un puntaje parcial. El puntaje para cada caso de prueba es $0.5 + 0.4 \times \left(\frac{2N - S}{N - \log_2 N} \right)^2$, multiplicado por el puntaje asignado a la subtarea.
- En cualquier otro caso, el puntaje es 0.

Nota que el puntaje para cada subtarea es el mínimo de los puntajes de cada caso de prueba en la subtarea.

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada del standard input en el siguiente formato:

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

El evaluador de ejemplo produce 3 salidas:

Primera, el evaluador de ejemplo imprime tu salida en un archivo llamado `out.txt` con el siguiente formato:

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

Segunda, el evaluador de ejemplo simula los movimiento de la bola e imprime el número de identificación de los componentes por los que la bola pasó en un archivo llamado `log.txt`.

Tercera, el evaluador de ejemplo imprime la evaluación de tu respuesta en el standard output.

- Si tu programa es evaluado como **Accepted**, el evaluador imprime S y P en el siguiente formato `Accepted: S P`.

- Si tu programa es evaluado como **Wrong Answer**, imprime `Wrong Answer: MSG`. Los significados de `MSG` son los siguientes:

  - `answered not exactly once`: El procedimiento `answer` no fue llamado exactamente una vez.
  - `wrong array length`: La longitud de `C` no es $M + 1$, o la longitud de `X` y `Y` son distintas.
  - `over 400000 switches`: S es mayor que $400\,000$.
  - `wrong serial number`: Hay un número de identificación `C`, `X`, o `Y` que es

menor que $-S$ o mayor que M.
- `over 20000000 inversions`: La bola no regresa al origen en $20\,000\,000$ o menos cambios de estado de interruptores.
- `state 'Y'`: Existe algún interruptor con estado 'Y' cuando la bola regresa al origen.
- `wrong motion`: Los interruptores que causan el movimiento fueron visitados por la bola de forma distinta a como es especificado en A.

Nota que el evaluador de ejemplo podría no crear los archivos `out.txt` y/o `log.txt` cuando tu programa es evaluado como `Wrong Answer`.