Muñeca Mecánica

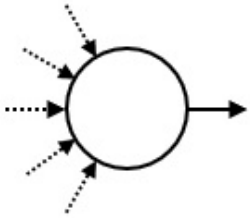
Una muñeca mecánica es una muñeca automática que repite una secuencia específica de movimientos. En Japón, muchas muñecas mecánicas fueron creadas desde tiempos antiguos.

Los movimientos de una muñeca mecánica son controlados por un **circuito** que consiste de **dispositivos**. Los dispositivos están conectados con tubos. Cada dispositivo tiene algunas (posiblemente cero) **entradas**, y una o dos **salidas**. Cada dispositivo puede tener arbitrariamente muchas entradas. Cada tubo conecta una salida de un dispositivo a una entrada del mismo u otro dispositivo. Un tubo está conectado exactamente a cada entrada y cada salida.

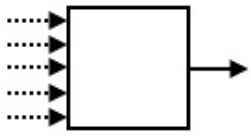
Para describir como la muñeca realiza los movimientos, imagina una **pelota** que es colocada en uno de los dispositivos. La pelota viaja a travez del circuito. En cada paso del viaje, esta deja el dispositivo usando una de sus salidas, viaja a travez del tubo conectada a la salida y entra al dispositivo en el otro extremo del tubo. La pelota viaja indefinidamente.

Hay tres tipos de dispositivos: **origen**, **lanzador**, y **cambio**. Hay exactamente un origen, M lanzadores, y S cambios (S puede ser cero). Tú debes decidir el valor de S. Cada dispositivo tienene un número identificador único.

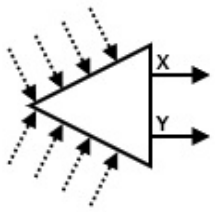
El origen es el dispositivo donde la bola es puesta inicialmente. Este tiene una salida. Su número identificador es 0.



Un lanzador hace que la muñeca realice un tipo de movimiento en cuanto la pelota entre a este. Cada lanzador tiene una salida. Los números identificadores de los lanzadores son de 1 a M.



Cada cambiador tiene dos salidas, las cuales son llamadas 'X' e 'Y'. El **estado** de un cambiador es entonces 'X' o 'Y'. Después que la pelota entra al cambiador, esta deja el cambiador usando la salida dada por el estado actual del cambiador. Después de eso, el cambiador cambia su estado al estado opuesto. Inicialmente, el estado de cada cambiador es 'X'. Los números identificadores de los cambiadores son de -1 a $-S$.



Tú tienes el número de lanzadores M. También tienes la secuencia A de tamaño N, cuyos elementos son los números identificadores de los lanzadores. Cada lanzador podría aparecer algunas (posiblemente cero) veces en A. Tu tarea es crear un circuito que cumpla las siguientes condiciones:

- La pelota retorna al origen después de algunos pasos.
- Cuando la pelota retorna por primera vez al origen, el estado de cada cambiador es 'X'.
- La pelota retorna por primera vez al origen después de entrar a los lanzadores exactamente N veces. Los números identificadores *consetutivos* de esos lanzadores son $A_0, A_1, \ldots, A_{N-1}$.
- Sea P el número total de cambios de estado de todos los cambiadores causados por la pelota antes que esta retorne al origen. El valor de P no excede $20\,000\,000$.

Al mismo tiempo, tú no quieres usar muchos cambiadores.

Detalles de implementación

Debes implementar el siguiente procedimiento.

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

- `M`: el número de lanzadores.
- `A`: un arreglo de tamaño N, dado los números identificadores consecutivos de los lanzadores a los cuales la pelota tiene que entrar.
- Este procedimiento es llamado exactamente una vez.
- Note que el valor de N es el tamaño del arreglo `A`, y puede ser obtenido como se indica en la nota de implementación.

Tu programa debería llamar al siguiente procedimiento para responder.

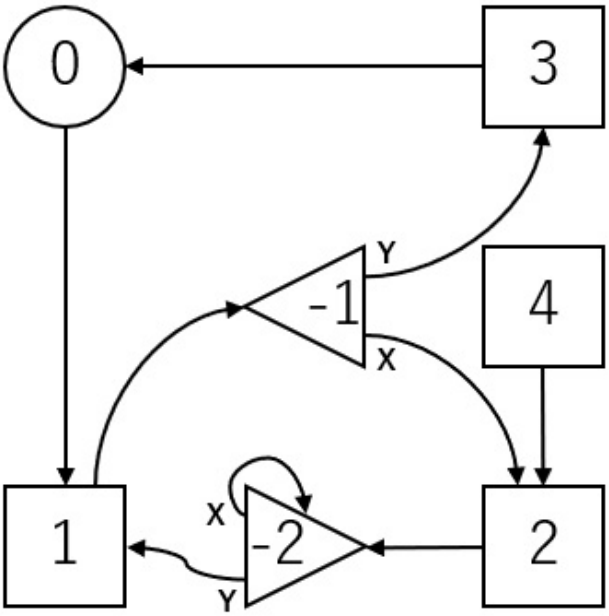
```
answer(int[] C, int[] X, int[] Y)
```

- `C`: un arreglo de tamaño $M + 1$. La salida del dispositivo i ($0 \leq i \leq M$) está conectada al dispositivo `C[i]`.
- `X`, `Y`: arreglos del mismo tamaño. El tamaño S de esos arreglos es el número de los cambiadores. Para el cambiador $-j$ ($1 \leq j \leq S$), su salida 'X' está conectada al dispositivo `X[j - 1]` y su salida 'Y' está conectada al dispositivo `Y[j - 1]`.
- Cada elemento de `C`, `X`, y `Y` debe ser un entero entre $-S$ and M, inclusive.
- S debe ser a lo mucho $400\,000$.
- Este procedimiento debe ser llamado exactaente una vez.
- El circuito representado por `C`, `X`, y `Y` debe satisfacere las condiciones del enunciado del problema.

Si algunas de las anteriores condiciones no son cumplidas, tu programa es juzgado como **`Wrong Answer`**. De otra manera, tu programa es juzgado como **`Accepted`** y tu puntaje es calculado por S (ver Subtareas).

Ejemplo

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



La anterior figura muestra un circuito, el cual está descrito por una llamada `answer([1, -1, -2, 0, 2], [2, -2], [3, 1])`. Los números en la figura son los números identificadores de los dispositivos.

Dos cambiadores son usados. Entonces $S = 2$.

Inicialmente, los estados de los cambiadores -1 y -2 ambos son 'X'.

La pelota viaja de la siguiente manera:

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

- Cuando la pelota entra por primera vez al cambiador -1, su estado es 'X'. Por lo tanto, la pelota viaja a el lanzador 2. Luego el estado del cambiador -1 es cambiado a 'Y'.
- Cuando la pelota entra al cambiador -1 por segunda vez, su estado es 'Y'. Por lo tanto, la pelota viaja al lanzador 3. Luego el estado del cambiador -1 es cambiado a 'X'.

La pelota retorna por primera vez al origen, habiendo entrado a los lanzadores $1, 2, 1, 3$. Los estados de los cambiadores -1 y -2 son ambos 'X'. El valor de P es 4. Por lo tanto, este circuito satisface las condiciones.

El archivo `sample-01-in.txt` en el paquete comprimido adjunto corresponde a este ejemplo. Otros ejemplos de entrada están disponibles en el paquete.

Restricciones

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

Subtareas

El puntaje y las restricciones de cada caso de prueba son los siguientes:

1. (2 puntos) Por cada i $(1 \le i \le M)$, el entero i aparece a lo mucho una vez en la secuencia $A_0, A_1, \ldots, A_{N-1}$.
2. (4 puntos) Por cada i $(1 \le i \le M)$, el entero i aparece a lo mucho dos veces en la secuencia $A_0, A_1, \ldots, A_{N-1}$.
3. (10 puntos) Por cada i $(1 \le i \le M)$, el entero i aparece a lo mucho 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 restriccioes adicionales

Para cada caso de prueba, si tu programa es juzgado como **`Accepted`**, tu puntaje es calculado de acuerdo al valor de S:

- Si $S \le N + \log_2 N$, tú ganas el puntaje total para el caso de prueba.
- Por cada caso de prueba en las subtareas 5 y 6, si $N + \log_2 N < S \le 2N$, tu ganas un puntaje parcial. El puntaje para el caso de prueba es $0.5 + 0.4 \times \left(\dfrac{2N - S}{N - \log_2 N} \right)^2$, multiplicado por el puntaje asignado a la subtarea.
- De otra forma, el puntaje es 0.

Note que su puntaje para cada subtarea es el mínimo de los puntajes de los casos de ejemplo en la subtarea.

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada desde la entrada estandar con el siguiente formato.

- línea 1: M N
- línea 2: A_0 A_1 ... A_{N-1}

El evaluador de ejemplo produce tres salidas.

Primero, el evaluador de ejemplo coloca tu respuesta a un archivo llamado `out.txt` en el siguiente formato.

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

Segundo, el evaluador de ejmplo simula las movidas de la pelota. Este imprime los números identificadores de los dispositivos a los cuales la pelota entro en orden aun archivo llamado `log.txt`.

Tercero, el evaluador de ejemplo imprime la evaluación de su salida a la salida standard.

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

- Si tu programa es juzgado como **`Wrong Answer`**, este imprime `Wrong Answer: MSG`. El significado de `MSG` es como sigue:

    - `answered not exactly once`: El procedimiento `answer` no es llamado exactamente una vez.
    - `wrong array length`: El tamaño de `C` no es $M + 1$, o los tamaños de `X` e `Y` son diferentes.
    - `over 400000 switches`: S es más grande que $400\,000$.
    - `wrong serial number`: Hay un elemento de `C`, `X`, o `Y` el cual es más pequeño que $-S$ o más grande que M.
    - `over 20000000 inversions`: La pelota no retorna al origen dentro de $20\,000\,000$ cambios de estado de los cambiadores.
    - `state 'Y'`: Hay un cambiador cuyo estado es 'Y' cuando la pelota retorna por primera vez al origen.
    - `wrong motion`: Los lanzadores que provocan movimientos son diferentes que que los de la secuencia A.

Note que el mismo evaluador no podría crear `out.txt` y/o `log.txt` cuando tu programa es juzgado como `Wrong Answer`.