Bambola meccanica

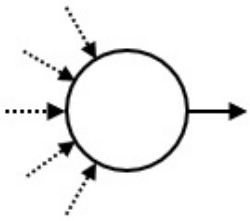
In Giappone, sin da tempi molto antichi, la *bambola meccanica* (bambola che esegue ripetutamente una specifica serie di movimenti) è considerata un pregevole manufatto.

I movimenti di una bambola meccanica sono controllati da un **circuito** che a sua volta consiste di **componenti**, connessi tra loro mediante tubi. Ogni componente ha un numero arbitrario di **ingressi** (possibilmente nessuno) ed esattamente *una* o *due* **uscite**. Ogni tubo connette un'uscita con un ingresso (possibilmente dello stesso componente), e ogni ingresso o uscita è sempre connesso a un qualche tubo.

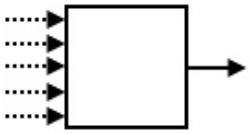
Per capire come una bambola esegue i movimenti, immagina che una **pallina** sia posizionata in uno dei componenti, e da lì inizi ad attraversare il circuito indefinitamente (senza mai fermarsi). Ad ogni passo del suo viaggio, la pallina lascia un componente da una delle sue uscite, viaggia lungo il tubo connesso all'uscita ed entra nel componente connesso all'altro capo del tubo.

Ci sono tre dipi di dispositivi: **sorgente**, **innesco**, e **scambio**. In ogni bambola c'è esattamente una sorgente, M inneschi ed S scambi (eventualmente nessuno); e ciascuno di questi dispositivi ha un *numero di serie* univoco.

La sorgente è il componente in cui la pallina è posizionata inizialmente, ed ha sempre un unica uscita e numero di serie 0.

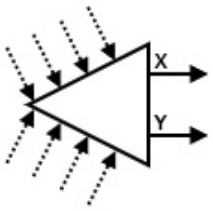


Un innesco fa sì che la bambola esegua uno specifico movimento ogni volta che la pallina lo attraversa, ed ha sempre un unica uscita. I numeri di serie degli inneschi sono compresi tra 1 ed M.



Ogni scambio ha due uscite, chiamate 'X' ed 'Y', ed ha anche uno **stato** che può parimenti essere 'X' o 'Y'. Inizialmente, lo stato di ogni scambio è 'X'. Quando la pallina

entra nel componente, lo lascia tramite l'uscita designata dallo stato corrente dello scambio; dopo che la pallina è uscita, lo stato dello scambio si inverte. I numeri di serie degli scambi sono compresi tra -1 e $-S$.



Dato il numero di inneschi M necessario, e una sequenza A composta da N numeri di serie di inneschi, devi progettare una bambola in grado di realizzare la sequenza di movimenti richiesta. Più precisamente:

- Ogni innesco può comparire un numero arbitrario di volte in A (possibilmente nessuna).
- Puoi decidere di quanti scambi S avvalerti (ma senza che siano troppi!).
- La pallina deve tornare alla sorgente per la prima volta dopo aver attraversato esattamente N inneschi, con numeri di serie $A_0, A_1, \ldots, A_{N-1}$ *in ordine di attraversamento*.
- Quando la pallina torna alla sorgente, lo stato di ogni scambio deve essere 'X'.
- Sia P il numero totale di inversioni di stato degli scambi causato dalla pallina prima che ritorni per la prima volta alla sorgente: il valore di P non deve superare $20\,000\,000$.

Dettagli di implementazione

Devi implementare la seguente funzione.

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

- `M`: numero di inneschi.
- `A`: array di lunghezza N, con i numeri di serie degli inneschi che la pallina deve attraversare.
- Questa funzione è chiamata esattamente una volta.
- Nota che la lunghezza N non viene fornita come parametro perché si può ottenere (come indicato nelle "Note di implementazione") dall'array stesso.

Il tuo programma deve chiamare la seguente funzione per riportare la risposta.

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

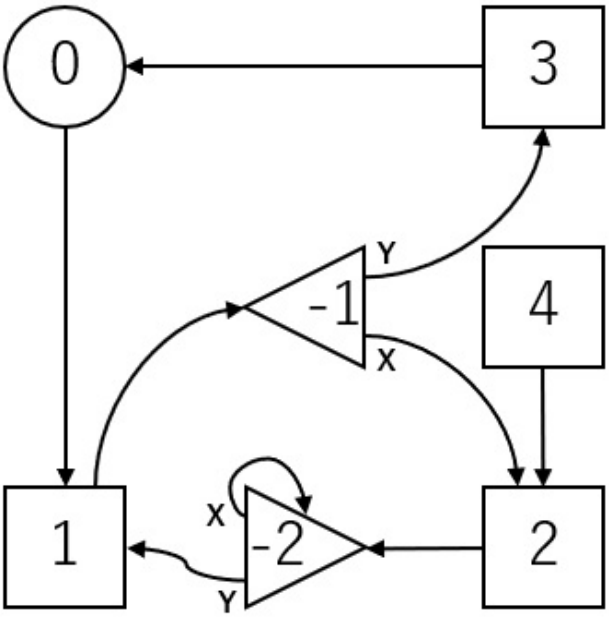
- `C`: array di lunghezza $M+1$, per cui l'uscita del componente i ($0 \leq i \leq M$) è collegata al componente `C[i]`.

- `X`, `Y`: array di uguale lunghezza S, contenenti la descrizione degli scambi. Per lo scambio $-j$ ($1 \le j \le S$), l'uscita 'X' è collegata al dispositivo `X[j - 1]` mentre l'uscita 'Y' è collegata al dispositivo `Y[j - 1]`.
- Ogni elemento di `C`, `X`, e `Y` deve essere un intero tra $-S$ ed M inclusi.
- S deve essere al massimo $400\,000$.
- Questa funzione deve essere chiamata esattamente una volta.
- Il circuito determinato da `C`, `X`, ed `Y` deve soddisfare le condizioni nel testo del problema.

Se una qualunque delle condizioni sopra non è verificata, il tuo programma è valutato come **`Wrong Answer`**. Altrimenti, il tuo programma è valutato come **`Accepted`** con un punteggio calcolato sulla base di S (vedi sezione Subtask).

Esempio

Siano $M = 4$, $N = 4$, $A = [1, 2, 1, 3]$.
Il grader chiama `create_circuit(4, [1, 2, 1, 3])`.



La figura sopra mostra un circuito, determinato dalla chiamata ad `answer([1, -1, -2, 0, 2], [2, -2], [3, 1])`. I numeri in figura sono i numeri di serie dei componenti.

In questo caso, $S = 2$ e quindi due scambi sono utilizzati. Inizialmente, lo stato degli scambi -1 e -2 è 'X'. La pallina percorre questo circuito come di seguito:

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

- Quando la pallina entra per la prima volta nello scambio -1, il suo stato è 'X' e quindi procede verso l'innesco 2, cambiando contemporaneamente lo stato dello scambio -1 in 'Y'.

- Quando la pallina entra per la seconda volta nello scambio -1, il suo stato è 'Y' e quindi procede verso l'innesco 3, cambiando contemporaneamente lo stato dello scambio -1 in 'X'.

La pallina ritorna quindi alla sorgente per la prima volta dopo aver attraversato gli scambi $1, 2, 1, 3$. A questo punto, lo stato degli scambi -1 e -2 è 'X' e il valore di P è 4, per cui il circuito soddisfa tutte le condizioni richieste.

Il file `sample-01-in.txt` nell'archivio compresso in allegato corrisponde a questo esempio. Altri input di esempio sono inoltre disponibili in questo archivio.

Assunzioni

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

Subtask

I punteggi e i vincoli per ogni caso di test sono i seguenti:

1. (2 punti) Per ogni i $(1 \leq i \leq M)$, l'intero i appare al più 1 volta nella sequenza $A_0, A_1, \ldots, A_{N-1}$
2. (4 punti) Per ogni i $(1 \leq i \leq M)$, l'intero i appare al più 2 volte nella sequenza $A_0, A_1, \ldots, A_{N-1}$
3. (10 punti) Per ogni i $(1 \leq i \leq M)$, l'intero i appare al più 4 volte nella sequenza $A_0, A_1, \ldots, A_{N-1}$
4. (10 punti) $N = 16$
5. (18 punti) $M = 1$
6. (56 punti) Nessuna limitazione aggiuntiva

Per ogni caso di test, se il tuo programma viene giudicato come **`Accepted`**, il tuo punteggio viene calcolato a seconda del valore di S:

- Se $S \leq N + \log_2 N$, prenderai il punteggio pieno per il caso di test.
- Per ogni caso di test nei Subtask 5 e 6, se $N + \log_2 N < S \leq 2N$, prenderai un punteggio parziale pari a $0.5 + 0.4 \times \left(\dfrac{2N - S}{N - \log_2 N}\right)^2$, moltiplicato per il punteggio assegnato al subtask.
- Altrimenti, il punteggio è 0.

Nota che il tuo punteggio per ogni subtask è il minimo dei punteggi dei casi di test del subtask.

Grader di esempio

Il grader di esempio legge l'input dallo standard input nel seguente formato:

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

Il grader di esempio produce tre output.

Innanzitutto, il grader di esempio stampa su un file chiamato `out.txt` nel seguente formato:

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

Poi, il grader di esempio simula il movimento della pallina e stampa i numeri di serie delle componenti che attraversa (in ordine) in un file chiamato `log.txt`.

Infine, il grader di esempio stampa la valutazione della tua risposta sullo standard output.

- Se il tuo programma viene giudicato come **`Accepted`**, il grader di esempio stampa S e P nel seguente formato: `Accepted: S P`.

- Se il tuo programma viene giudicato come **`Wrong Answer`**, stampa `Wrong Answer: MSG`. Il significato di `MSG` è il seguente:

    - `answered not exactly once`: La procedura `answer` non è stata chiamata esattamente una volta.
    - `wrong array length`: La lunghezza di `C` non è $M + 1$, o le lunghezze di `X` e `Y` sono diverse.
    - `over 400000 switches`: S è più grande di $400\,000$.
    - `wrong serial number`: C'è un elemento di `C`, `X` o `Y` che è più piccolo di $-S$ o più grande di M.
    - `over 20000000 inversions`: La pallina non ritorna nella sorgente entro $20\,000\,000$ cambi di stato degli scambi.
    - `state 'Y'`: C'è uno scambio con stato `Y` quando la pallina ritorna alla sorgente.
    - `wrong motion`: Gli inneschi che vengono attraversati dalla pallina non sono quelli della sequenza A.

Nota che il grader di esempio potrebbe non creare `out.txt` e/o `log.txt` quando il tuo programma viene giudicato come `Wrong Answer`.