Mechanical Doll

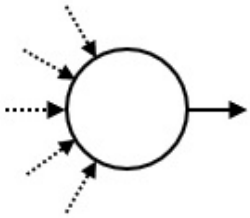
O jucărie mecanică este o jucărie care repetă automat o secvență de mișcări. În Japonia multe jucării mecanice au fost create încă din vremuri antice.

Mișcările unei jucării mecanice sunt controlate de un **circuit**, constituit din **dispozitive**. Dispozitivele sunt conectate prin tuburi. Fiecare dispozitiv are una sau două **ieșiri** și un număr arbitrar (posibil zero) de **intrări**. Fiecare tub conectează ieșirea unui dispozitiv cu intrarea aceluiași sau altui dispozitiv. Exact un tub este conectat cu fiecare intrare și exact un tub este conectat cu fiecare ieșire.

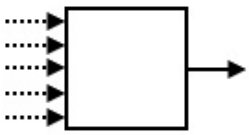
Pentru a descrie mișcările produse de jucărie, considerați o **bilă** plasată într-unul din dispozitive. Bila se mișcă prin circuit. La fiecare moment al traseului acesteia, bila, părăsește dispozitivul folosind una din ieșiri, traversează tubul conectat de ieșire și intră în dispozitivul de la capătul celălalt al tubului.

Există trei tipuri de dispozitive: **origine**, **întrerupător** și **comutator**. Există exact o origine, M întrerupătoare și S comutatoare (S poate fi zero). Voi trebuie să decideți valoarea lui S. Fiecare dispozitiv are un cod unic de înregistrare.

Originea este dispozitivul unde bila se află inițial. Are exact o ieșire. Codul ei unic de înregistrare este 0.

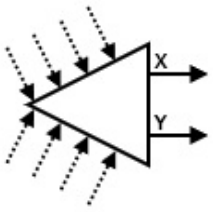


Un întrerupător face bila să execute o mișcare specială de fiecare dată când bila intră în acesta. Orice întrerupător are exact o ieșire. Codurile unice de înregistrare ale întrerupătoarelor sunt de la 1 la M.



Fiecare comutator are două ieșiri, denumite 'X' și 'Y'. **Starea** unui comutator este fie 'X', fie 'Y'. După ce bila intră într-un comutator îl părăsește prin ieșirea dată de starea curentă a acestuia. După aceea, starea comutatorului se schimbă. Inițial starea fiecărui

comutator este 'X'. Codurile unice de înregistrare ale comutatoarelor sunt de la -1 la $-S$.



Vi se dă numărul de întrerupătoare M. Vi se dă de asemenea o secvență A de lungime N, ale cărei elemente reprezintă coduri unice de înregistrare ale întrerupătoarelor. Fiecare întrerupător poate apărea de câteva (posibil zero) ori în secvența A. Sarcina voastră este să creați un circuit care satisface următoarele condiții:

- Bila se întoarce în origine după niște pași.
- Când bila se întoarce prima dată în origine, starea fiecărui comutator este 'X'.
- Bila se întoarce prima dată în origine după ce a intrat în întrerupătoare de exact N ori. Codurile unice de înregistrare ale acestor întrerupătoare, în ordinea în care trebuie să intre bila în ele, sunt $A_0, A_1, \ldots, A_{N-1}$.
- Fie P numărul total de schimbări de stare ale comutatoarelor cauzate de bilă înainte ca aceasta să se întoarcă la origine prima dată. Valoarea lui P nu poate depăși $20\,000\,000$.

De asemenea, nu vreți să folosiți prea multe comutatoare.

Detalii de implementare

Trebuie să implementați următoarea procedură.

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

- `M`: numărul de întrerupătoare
- `A`: un tablou unidimensional cu N elemente, reprezentând codurile unice de înregistrare ale întrerupatoarelor prin care trebuie să intre bila, în ordinea în care trebuie să intre bila în ele.
- Această procedură este apelată exact o dată.
- Luați la cunoștință că valoarea lui N este dimensiunea tabloului `A`, și poate fi obținut după cum este indicat în Observațiile de implementare.

Programul vostru trebuie să apeleze următoarea procedură pentru a întoarce răspunsul.

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

- `C`: un tablou unidimensional cu $M+1$ elemente. Ieșirea dispozitivului i (

$0 \le i \le M$) este conectată cu dispozitivul `C[i]`.
- `X`, `Y`: tablouri unidimensionale cu număr egal de elemente. Dimensiunea S a acestor tablouri reprezintă numărul de comutatoare. Pentru comutatorul $-j$ ($1 \le j \le S$) ieșirea 'X' a acestuia este conectată cu dispozitivul `X[j - 1]` și ieșirea 'Y' cu dispozitivul `Y[j - 1]`.
- Orice element a lui 'C', 'X' și 'Y' trebuie să fie un număr întreg între $-S$ și M inclusiv.
- S trebuie să fie cel mult $400\,000$.
- Această procedură trebuie apelată exact o dată.
- Circuitul reprezentat de 'C', 'X' și 'Y' trebuie să satisfacă condițiile din enunțul problemei.

Dacă vreuna din condițiile de mai sus nu este satisfăcută programul vostru va fi evaluat ca **`Wrong answer`**. Altfel, programul vostru va fi evaluat ca **`Accepted`**, iar punctajul vostru va fi calculat pe baza lui S (vedeți Subtask-uri).

Exemplu

Fie $M = 4$, $N = 4$ și $A = [1, 2, 1, 3]$. Grader-ul apelează `create_circuit(4, [1, 2, 1, 3])`.

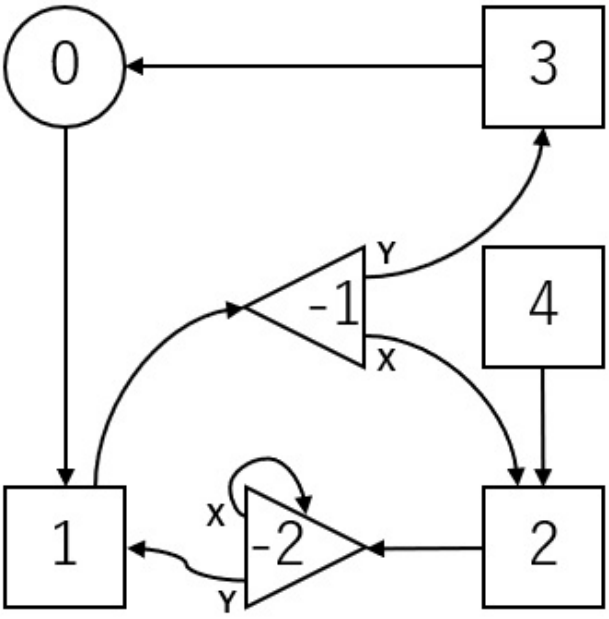


Figura de mai sus arată un circuit care este descris de apelul `answer([1, -1, -2, 0, 2], [2, -2], [3, 1])`. Numerele din figură reprezintă codurile unice de înregistrare ale dispozitivelor.

Două comutatoare sunt folosite. Deci $S = 2$.

Inițial, starea comutatoarelor -1 și -2 este 'X'.

Bila are următorul traseu:

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

- Când bila intră prima oară în comutatorul -1 starea acestuia este 'X'. Deci, bila va traversa către întrerupătorul 2. Apoi, starea comutatorului -1 devine 'Y'.
- Când bila intră în comutatorul -1 pentru a doua oară starea acestuia este 'Y'. Deci bila va traversa către întrerupătorul 3. Apoi, starea comutatorului -1 devine 'X'.

Când bila revine prima oară în origine aceasta a trecut prin întrerupătoarele $1, 2, 1, 3$. Starea comutatoarelor -1 și -2 este 'X'. Valoarea lui P este 4. Deci circuitul satisface condițiile.

Fișierul `sample-01-in.txt` din pachetul arhivat atașat corespunde acestui exemplu. Alte exemple se găsesc în acest pachet.

Restricții

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

Subtask-uri

Punctajele și restricțiile pentru fiecare test sunt după cum urmează:

1. (2 puncte) Pentru fiecare i $(1 \leq i \leq M)$ numărul i apare cel mult o dată în secvența $A_0, A_1, \ldots, A_{N-1}$.
2. (4 puncte) Pentru fiecare i $(1 \leq i \leq M)$ numărul i apare de cel mult de două ori în secvența $A_0, A_1, \ldots, A_{N-1}$.
3. (10 puncte) Pentru fiecare i $(1 \leq i \leq M)$ numărul i apare de cel mult patru ori în secvența $A_0, A_1, \ldots, A_{N-1}$.
4. (10 puncte) $N = 16$
5. (18 puncte) $M = 1$
6. (56 puncte) Fără constrângeri adiționale.

Pentru fiecare test, dacă programul vostru este evaluat ca **`Accepted`**, punctajul vostru este calculat pe baza lui S:

- Dacă $S \leq N + \log_2 N$ veți primi punctajul complet pentru acel test.
- Pentru fiecare test din subtask-urile 5 și 6, dacă $N + \log_2 N < S \leq 2N$ veți primi un punctaj parțial. Punctajul pentru acest test este $0.5 + 0.4 \times \left(\dfrac{2N - S}{N - \log_2 N}\right)^2$ multiplicat cu punctajul asociat subtask-ului.
- Altfel punctajul este 0.

Luați la cunoștință că punctajul pentru fiecare subtask este minimul punctajelor pentru fiecare test din acel subtask.

Grader local

Grader-ul local citește datele de intrare de la intrarea standard în următoarea formă.

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

Grader-ul local produce trei ieșiri.

Mai întâi grader-ul local va afișa răspunsul vostru într-un fișier numit `out.txt` în următoarea formă.

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

Apoi, grader-ul local va simula mișcarea bilei. Va afișa codurile unice de înregistrare ale dispozitivelor prin care trece bila în fișierul `log.txt`.

În sfârșit, grader-ul local va afișa evaluarea răspunsului vostru la ieșirea standard.

- Dacă programul vostru este evaluat ca **`Accepted`** grader-ul local va afișa S și P în următoarea formă `Accepted: S P`.
- Dacă programul vostru este evaluat ca **`Wrong Answer`**, grader-ul local va afișa `Wrong answer: MSG`. Semnificația lui `MSG` este după cum urmează:
  - `answered not exactly once`: Procedura `answer` nu a fost apelată exact o dată.
  - `wrong array length`: Dimensiunea tabloului `C` nu este $M + 1$, sau dimensiunile lui `X` și `Y` sunt diferite.
  - `over 400000 switches`: S este mai mare decât $400\,000$.
  - `wrong serial number`: Există cel puțin un element în 'C', 'X' sau 'Y' care este mai mic decât $-S$ sau mai mare decât M.
  - `over 20000000 inversions`: Bila nu a revenit la origine în $20\,000\,000$ de schimbări de stare ale comutatoarelor.
  - `state 'Y'`: Există un comutator al cărui stare este 'Y' când bila revine prima oară în origine.
  - `wrong motion`: Întrerupătoarele care determină mișcarea bilei sunt diferite de cele din secvența A.

Luați la cunoștință că grader-ul local nu creează `out.txt` și/sau `log.txt` atunci când programul vostru este evaluat ca `Wrong Answer`.