Mechanical Doll

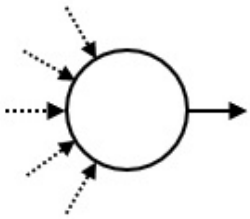
Une poupée mécanique est une poupée qui répète automatiquement une séquence spécifique de mouvements. Au japon, de nombreuses poupées mécaniques ont été créées et ce depuis des siècles.

Les mouvements d'une poupée mécanique sont contrôlés par un **circuit** constitué de **composants**. Les composants sont connectés par des tubes. Chaque composant possède une ou deux **sorties** et peut avoir zéro ou plusieurs entrées. Chaque tube relie une sortie d'un composant à une entrée du même composant ou d'un autre. Exactement un tube est connecté à chaque entrée et exactement un tube est connecté à chaque sortie.

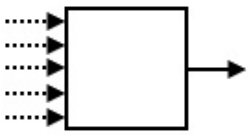
Pour décrire les mouvements de la poupée, pensez à une **balle** qui est placée dans l'un des composants. La balle se déplace à travers le circuit. A chaque étape de son déplacement, la balle quitte le composant en utilisant une des sorties, traverse le tube connecté à la sortie et entre dans le composant à l'autre extrémité du tube.

Il existe trois types de composants : **origine**, **déclencheur**, et **commutateur**. Il y a exactement une origine, M déclencheurs, et S commutateurs (S peut être égal à zéro). Vous devez décider de la valeur de S. Chaque composant a un numéro de série unique.

L'origine est le composant où la balle est initialement placée. Elle a une sortie. Son numéro de série est 0.

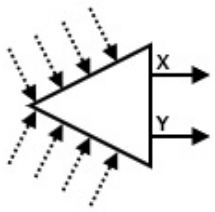


Un déclencheur provoque un mouvement spécifique à la poupée à chaque fois que la balle y entre. Chaque déclencheur possède une sortie. Les numéros de série des déclencheurs vont de 1 à M.



Chaque commutateur possède deux sorties 'X' et 'Y'. L'**état** d'un commutateur est soit

'X' soit 'Y'. Quand une balle entre dans un commutateur, elle sort en utilisant la sortie donnée par l'état actuel de ce commutateur. Ensuite, le commutateur change à l'état opposé. Initialement, l'état de chaque commutateur est 'X'. Les numéros de série des commutateurs vont de -1 à $-S$.



On vous donne M, le nombre de déclencheurs. On vous donne aussi une séquence A de longueur N composée de numéros de série de déclencheurs. Chaque déclencheur peut apparaitre zéro ou plusieurs fois dans la séquence A. Votre objectif est de créer un circuit qui satisfait les conditions suivantes :

- La balle revient à l'origine après un certain nombre d'étapes.
- Quand la balle revient pour la première fois à l'origine, l'état de chaque commutateur est 'X'.
- La balle revient pour la première fois à l'origine après être entrée dans les déclencheurs exactement N fois. L'ordre dans lequel la balle entre dans les déclencheurs est $A_0, A_1, \ldots, A_{N-1}$.
- Soit P le nombre total de changements d'état de tous les commutateurs causés par la balle avant son premier retour à l'origine. La valeur de P ne dépasse pas $20\,000\,000$.

En même temps, vous ne souhaitez pas utiliser trop de commutateurs.

Détails d'implémentation

Vous devez implémenter la procédure suivante :

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

- `M` : le nombre de déclencheurs
- `A` : un tableau de taille N fournissant des numéros de série de déclencheurs dans l'ordre dans lequel la balle doit entrer.
- Cette procédure est appelée exactement une fois.
- Notez que la valeur de N est la longueur du tableau `A` et peut être obtenue comme indiqué dans la notice d'implémentation.

Votre programme doit appeler la procédure suivante pour fournir une réponse.

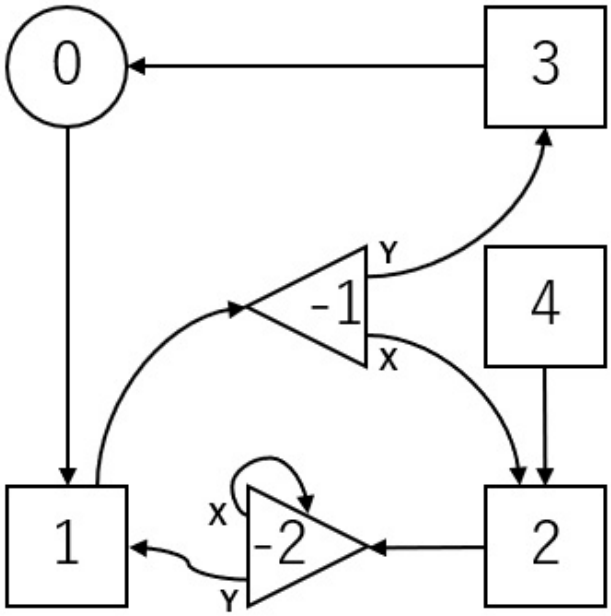
```
answer(int[] C, int[] X, int[] Y)
```

- `C` : un tableau de taille $M+1$. La sortie du composant i ($0 \le i \le M$) est connectée au composant `C[i]`.
- `X`, `Y` : tableaux de même taille. La taille S de ces tableaux correspond au nombre de commutateurs. Pour le commutateur $-j$ ($1 \le j \le S$), sa sortie 'X' est connectée au composant `X[j - 1]` et sa sortie 'Y' est connectée au composant `Y[j - 1]`.
- Chaque élément de `C`, `X`, et `Y` doit être un entier compris entre $-S$ et M, inclus.
- S doit être au plus 400 000.
- Cette procédure doit être appelée exactement une fois.
- Le circuit représenté par `C`, `X`, et `Y` doit satisfaire les conditions décrites dans l'énoncé de ce problème.

Si l'une des conditions est non satisfaite, votre programme recevra **Wrong Answer**. Sinon, votre programme recevra **Accepted** et votre score sera calculé en fonction de S (voir sous-tâches)

Exemple

Pour $M = 4$, $N = 4$, et $A = [1, 2, 1, 3]$. L'évaluateur (grader) appelle `create_circuit(4, [1, 2, 1, 3])`.



La figure ci-dessus montre un circuit décrit par l'appel de `answer([1, -1, -2, 0, 2], [2, -2], [3, 1])`. Les nombres dans la figure sont les numéros de série des composants.

Deux commutateurs sont utilisés. Ainsi $S = 2$.

Initialement, les états des commutateurs -1 et -2 sont à 'X'.

La balle se déplace comme suit :

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

- Quand la balle entre pour la première fois dans le commutateur -1, son état est 'X'. Ainsi, la balle se déplace vers le déclencheur 2. Ensuite, l'état du commutateur -1 passe à 'Y'.
- Quand la balle entre dans le commutateur -1 pour la seconde fois, son état est 'Y'. Ainsi, la balle se déplace vers le déclencheur 3. Ensuite, l'état du commutateur -1 passe à 'X'.

La balle retourne la première fois à l'origine après être passée par les déclencheurs $1, 2, 1, 3$. Les états des commutateurs -1 et -2 sont à 'X'. La valeur de P est 4. Donc, ce circuit satisfait les conditions.

Le fichier `sample-01-in.txt` dans l'archive du problème correspond à cet exemple.

D'autres exemples sont également disponibles dans l'archive.

Contraintes

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

Sous-tâches

Le score et les contraintes pour chaque cas de test sont comme suit:

1. (2 points) pour chaque i $(1 \leq i \leq M)$, l'entier i apparait au plus une fois dans la séquence $A_0, A_1, \ldots, A_{N-1}$.
2. (4 points) pour chaque i $(1 \leq i \leq M)$, l'entier i apparait au plus deux fois dans la séquence $A_0, A_1, \ldots, A_{N-1}$.
3. (10 points) pour chaque i $(1 \leq i \leq M)$, l'entier i apparait au plus 4 fois dans la séquence $A_0, A_1, \ldots, A_{N-1}$.
4. (10 points) $N = 16$
5. (18 points) $M = 1$
6. (56 points) Pas de contraintes additionnelles

Pour chaque cas de test, si votre programme est jugé comme **`Accepted`**, votre score est calculé en fonction de la valeur de S :

- Si $S \leq N + \log_2 N$, vous obtenez le score total pour le cas de test.
- Pour chaque cas de test des sous-tâches 5 et 6, si $N + \log_2 N < S \leq 2N$, vous obtenez un score partiel. Le score pour le cas de test est $0.5 + 0.4 \times \left(\dfrac{2N - S}{N - \log_2 N}\right)^2$, multiplié par le score de la sous-tâche.

- Sinon, le score est 0.

Notez que votre score pour chaque sous-tâche est le minimum des scores des cas de test dans la sous-tâche.

Evaluateur d'exemple (sample grader)

L'évaluateur d'exemple (sample grader) lit l'input à partir de l'entrée standard comme suit:

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

L'évaluateur d'exemples produit trois sorties.

En premier lieu, l'évaluateur d'exemple écrit votre réponse sur un fichier nommé `out.txt` dans le format suivant.

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

En second lieu, l'évaluateur d'exemple simule les mouvements de la balle. Il écrit les numéros de série des composants où la balle est entrée, dans l'ordre, dans un fichier nommé `log.txt`.

En troisième lieu, l'évaluateur d'exemple écrit l'évaluation de vos réponses vers la sortie standard .

- Si votre programme est jugé **Accepted**, l'évaluateur d'exemple affiche S et P comme suit `Accepted: S P`.
- Si votre programme est jugé **Wrong Answer**, il affiche `Wrong Answer: MSG`. La signification de `MSG` est comme suit:
  - `answered not exactly once` : La procédure `answer` n'a pas été appelée exactement une fois.
  - `wrong array length` : La longueur de `C` n'est pas $M+1$, ou les longueurs de `X` et `Y` sont différentes.
  - `over 400000 switches` : S est plus grand que $400\,000$.
  - `wrong serial number` : Il y a un élément de `C`, `X`, ou `Y` qui est plus petit que $-S$ ou plus grand que M.
  - `over 20000000 inversions` : la balle n'est pas revenue à l'origine en $20\,000\,000$ changements d'état des commutateurs.
  - `state 'Y'` : Il y a un commutateur dont l'état est 'Y' quand la balle est revenue la première fois à l'origine.
  - `wrong motion` : Les déclencheurs qui engendrent le mouvement sont différents de la séquence A.

Notez que l'évaluateur d'exemple peut ne pas créer `out.txt` et/ou `log.txt` quand votre programme est jugé `Wrong Answer`.