Mechanical Doll

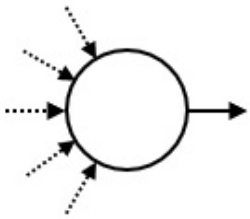
Uma boneca mecânica é uma boneca que repete automaticamente uma sequência específica de movimentos. No Japão, desde a antiguidade, foram criadas muitas bonecas mecânicas.

Os movimentos de uma boneca mecânica são controlados por um **circuito** composto de **componentes**, conectados por tubos. Cada componente possui uma ou duas **saídas**, e pode ter um número arbitrário (possivelmente zero) de **entradas**. Cada componente pode ter um número arbitrário de entradas. Cada tubo conecta a saída de um componente à entrada do mesmo ou de outro componente. Exatamente um tubo está conectado a cada entrada e exatamente um tubo está conectado a cada saída.

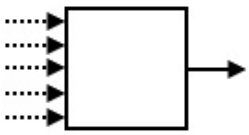
Para entender como a boneca se mexe, considere que uma **bola** é colocada em um dos componentes. A bola anda pelo circuito. A cada passo de seu movimento, ela deixa o componente por uma de suas saídas, atravessa o tubo conectado a essa saída, e entra no componente na outra ponta do tubo.

Há três tipos de componentes: **origem**, **gatilho**, e **interruptor**. Há exatamente uma origem, M gatilhos e S interruptores (S pode ser zero). Sua tarefa é determinar o valor de S. Cada componente tem um número de série distinto.

A origem é o componente onde a bola inicia seu trajeto. Ele tem uma saída e seu número de série é 0.

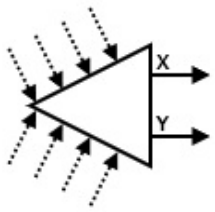


O gatilho faz a boneca executar um movimento específico sempre que a bola entra no componente. Todo gatilho tem uma saída; os números de série dos gatilhos variam de 1 a M.



Um interruptor tem duas saídas, chamadas de 'X' e 'Y'. O **estado** de um interruptor é

'X' ou 'Y'. Após a bola entrar em um interuptor, ela sai do componente através da saída indicada pelo estado atual do interruptor. Em seguida, interruptor troca seu estado para o estado oposto. Inicialmente, o estado de cada interruptor é 'X'. Os números de série dos interruptores variam de -1 a $-S$.



Você receberá o número de gatilhos M, assim como uma sequência A de comprimento N, cujos elementos são números de série de gatilhos; cada gatilho pode figurar várias vezes (ou nenhuma vez) em A. Sua tarefa é criar um circuito que satisfaça as seguintes condições:

- A bola retorna à origem após algum número de passos.
- Quando a bola returna à origem pela primeira vez, o estado de cada interruptor é 'X'.
- A bola returna à origem pela primeira vez após ativar gatilhos exatamente N vezes. Os números de série destes gatilhos, *em ordem*, são $A_0, A_1, \ldots, A_{N-1}$.
- Seja P o número de trocas de estados em gatilhos causados pela bola antes de seu retorno à origem; o valor de P não deve exceder $20\,000\,000$.

Ao mesmo tempo, você não quer usar muitos interuptores.

Detalhes de implementação

Você deve implementar o procedimento seguinte:

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

- `M`: o número de gatilhos.
- `A`: um array de comprimento N, indicando, em ordem, os números de série dos gatilhos que a bola precisa entrar.
- Este procedimento é chamado exatamente uma vez.
- Note que o valor de N é o comprimento do array `A`, e pode ser obtido conforme as notas de implementação.

Seu programa deve chamar o seguinte procedimento para responder:

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

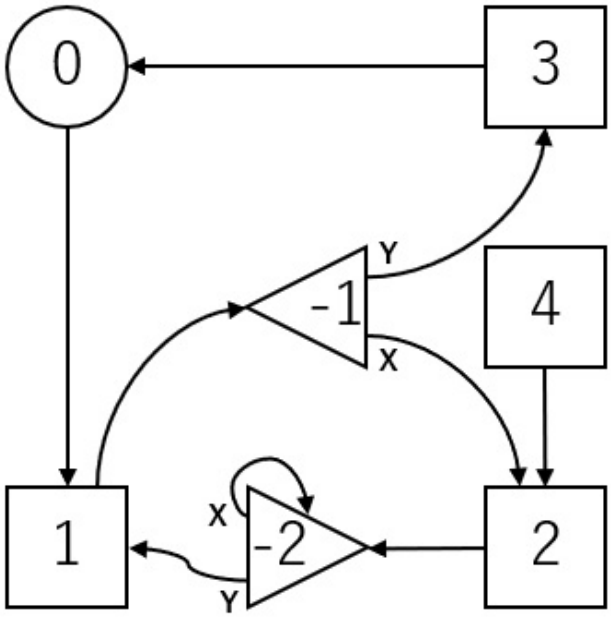
- `C`: um array de comprimento $M+1$. A saída do componente i ($0 \leq i \leq M$) é conectada ao componente `C[i]`.

- `X`, `Y`: arrays do mesmo comprimento. O comprimento S desses arrays é o número de interruptores. O interruptor $-j$ ($1 \leq j \leq S$), tem a sua saída 'X' conectada ao dispositivo `X[j - 1]`, e a sua saída 'Y' conectada ao dispositivo `Y[j - 1]`.
- Todo elemento de `C`, `X`, e `Y` deve ser um inteiro entre $-S$ e M, inclusive.
- S deve ser no máximo $400\,000$.
- Este procedimento deve ser chamado exatamente uma vez.
- O circuito representado por `C`, `X`, e `Y` deve satisfazer as condições do enunciado.

Se alguma das condições acima não for satisfeita, seu programa será julgado **`Wrong Answer`**. Caso contrário, seu programa será julgado **`Accepted`** e sua pontuação será calculada a partir de S (veja a seção Subtarefas).

Example

Seja $M = 4$, $N = 4$, e $A = [1, 2, 1, 3]$. O corretor chama `create_circuit(4, [1, 2, 1, 3])`.



A figura acima mostra o circuito indicado pela chamada `answer([1, -1, -2, 0, 2], [2, -2], [3, 1])`. Os números na figura são os números de série dos componentes.

Dois interruptores são usados e, portanto, $S = 2$.

Inicialmente, os estados dos interruptores -1 e -2 são ambos 'X'.

A bola percorre o circuito da seguinte maneira:

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

- Quando a bola entra o interruptor -1 pela primeira vez, seu estado é 'X'. Logo, a bola segue para o gatilho 2; em seguida, o estado do interruptor -1 muda para 'Y'.

- Quando a bola entra o interruptor -1 pela segunda vez, seu estado é 'Y'. Logo, a bola segue para o gatilho 3; em seguinda, o estado do interruptor -1 muda para 'X'.

Finalmente, a bola retorna à origem após ter percorrido os gatilhos $1, 2, 1, 3$. Os estados dos gatilhos -1 e -2 são ambos 'X'. O valor de P é 4. Portanto, o circuito satisfaz as condições do enunciado.

O arquivo `sample-01-in.txt` no pacote anexo compactado corresponde a esse exemplo; outros exemplos de entrada também estão disponíveis no pacote.

Restrições

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

Subtarefas

A pontução e as restrições para cada caso de teste são:

1. (2 pontos) Para cada i $(1 \le i \le M)$, o inteiro i aparece no máximo uma vez na sequência $A_0, A_1, \ldots, A_{N-1}$.
2. (4 pontos) Para cada i $(1 \le i \le M)$, o inteiro i aparece no máximo duas vezes na sequência $A_0, A_1, \ldots, A_{N-1}$.
3. (10 pontos) Para cada i $(1 \le i \le M)$, o inteiro i aparece no máximo quatro vezes na sequência $A_0, A_1, \ldots, A_{N-1}$.
4. (10 pontos) $N = 16$
5. (18 pontos) $M = 1$
6. (56 pontos) Nenhuma restrição adicional

Para cada caso de teste, se seu programa for julgado **`Accepted`**, sua pontuação será calculada de acordo com valor de S:

- Se $S \le N + \log_2 N$, você recebe a pontuação integral para o caso de teste.
- Para cada caso de teste mas subtarefas 5 e 6, se $N + \log_2 N < S \le 2N$, você receberá pontuação parcial: a pontuação de cada caso de teste é $0.5 + 0.4 \times \left(\dfrac{2N - S}{N - \log_2 N} \right)^2$, multiplicado pela pontuação atribuída à subtarefa.
- Caso contrário, a sua pontuação é 0.

Note que a sua pontuação em cada subtarefa é o mínimo das pontuações dos casos de teste da subtarefa.

Corretor de exemplo

O corretor de exemplo lê a entrada da entrada padrão, no seguinte formato:

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

O corretor de exemplo produz três saídas

Primeiro, o corretor imprime a sua saída num arquivo `out.txt`, no seguinte formato:

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

Segundo, o corretor de exemplo simula os movimentos da bola. Ele imprime, em ordem, os números de série dos componentes em que a bola entrou para um arquivo chamado `log.txt`.

Terceiro, o corretor de exemplo imprime o julgamento da sua resposta para a saída padrão:

- Se seu programa for julgado **`Accepted`**, o corretor de exemplo imprime S e P no formato `Accepted: S P`.
- Se seu programa for julgado **`Wrong Answer`**, o corretor de exemplo imprime `Wrong Answer: MSG`. O significado de `MSG` é:
  - `answered not exactly once`: o procedimento `answer` não é chamado exatamente uma vez.
  - `wrong array length`: o comprimento de `C` não é $M + 1$, ou os comprimentos de `X` e `Y` são diferentes.
  - `over 400000 switches`: S é maior do que $400\,000$.
  - `wrong serial number`: há um elemento de `C`, `X`, ou `Y` menor do que $-S$ ou maior do que M.
  - `over 20000000 inversions`: a bola não retorna à origem em $20\,000\,000$ mudanças de estado dos interruptores.
  - `state 'Y'`: há um interruptor cujo estado é 'Y' no momento em que a bola retorna à origem.
  - `wrong motion`: os gatilhos ativados são diferentes da sequência A.

Note que o corretor pode não criar arquivos `out.txt` e/ou `log.txt` quando o seu programa é julgado `Wrong Answer`.