자동 인형

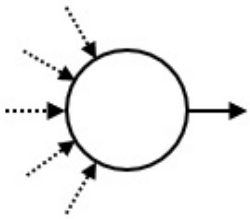
자동 인형은 동일한 과정을 자동으로 반복하는 인형이다. 일본에서는 여러가지 자동 인형이 오래 전 부터 만들어져 왔다.

자동 인형의 동작은 **장치**들로 구성된 **회로**에 의해 조정된다. 장치들은 튜브로 연결되어 있다. 각 장치는 몇 개의 **진입구**와 한 개 혹은 두 개의 **출구**를 가진다. 각 장치가 가질 수 있는 진입구의 개수에는 제한이 없다 (0개일 수 있음). 각 튜브는 어떤 장치의 출구와 어떤 장치의 진입구를 연결한다. 한 장치의 출구와 동일한 장치의 진입구를 연결하는 것도 가능하다. 각 출구와 각 진입구에는 정확히 하나씩의 튜브가 연결되어 있다.

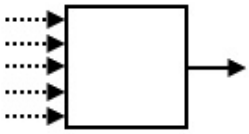
자동 기계의 동작을 설명해 보자. **공** 하나가 어떤 장치에 있다고 하자. 이 공은 회로를 돌아다니게 된다. 각 스텝에서 공은 현재 공이 위치한 장치의 출구 중 하나로 나와서 튜브를 따라 이동한 다음 어떤 장치의 진입구로 들어간다.

장치는 **시작**, **트리거**, **스위치**의 3가지 종류가 있다. 시작 장치는 정확히 1개, 트리거는 M개, 스위치는 S개가 있다. (S는 0일 수 있다.) S의 값은 당신이 결정해야 한다. 각 장치는 유일한 일련번호를 가진다.

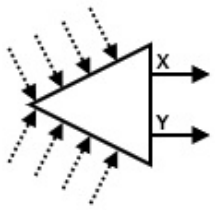
시작 장치는 공이 최초에 놓이는 장치이다. 시작 장치는 1개의 출구를 가진다. 시작 장치의 일련번호는 0이다. (앞으로 단순히 **시작**이라고 부른다.)



트리거 장치는, 공이 들어갈 때마다 인형이 재미있는 동작을 하게 만드는 장치이다. 트리거 장치는 1개의 출구를 가진다. 트리거 장치의 일련번호는 1부터 M까지이다. (앞으로 단순히 **트리거**, 혹은 **트리거** i라고 부른다.)



스위치 장치는 'X'와 'Y'라고 불리는 2개의 출구를 가진다. 스위치 장치의 상태는 'X'나 'Y' 중 하나이다. 공이 스위치 장치에 들어가면 공은 그 스위치 장치의 상태와 같은 출구로 나간다. 직후에, 스위치 장치의 상태는 다른 상태로 바뀐다. 초기에 모든 스위치 장치의 상태는 'X'이다. 스위치 장치의 일련번호는 -1부터 $-S$까지이다. (앞으로 단순히 **스위치**, 혹은 **스위치** $-j$라고 부른다.)



트리거의 개수 M이 입력으로 주어진다. 또, 길이 N인 수열 (혹은 배열) A가 주어진다. 수열 A의 원소는 트리거의 일련번호들이다. 각 트리거의 일련번호는 수열 A에 몇 번 (혹은 0번) 나타난다. 당신은 다음 조건을 만족하는 회로를 구성해야 한다.

- 공은 몇 번의 스텝을 거친 다음에 시작으로 돌아와야 한다.
- 공이 최초로 시작에 다시 돌아올 때, 모든 스위치의 상태는 'X'이어야 한다.
- 공이 최초로 시작에 다시 돌아올 때까지 공은 트리거를 정확히 N번 지나야 한다. 또, 공이 트리거들을 지나간 순서의 일련번호는 $A_0, A_1, \ldots, A_{N-1}$와 **완전히 같아야** 한다.
- P를 공이 최초로 시작에 돌아올 때까지 스위치들의 상태가 바뀐 총 횟수라고 하자. P는 $20\,000\,000$이 넘으면 안된다.

당신이 사용하는 스위치의 개수는 충분히 작아야 한다.

Implementation details

다음 함수를 구현해야 한다.

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

- `M`: 트리거의 개수
- `A`: 크기 N인 배열로서, 공이 지나가야 하는 트리거의 일련번호가 순서대로 주어진다.
- 이 함수는 정확히 한 번 호출된다.
- N은 배열 `A`의 크기이다. 이 값은 구현 명세에서 알려준 방법을 통해서 얻을 수 있다는 데 주의하라.

당신의 프로그램은 다음 함수를 호출해서 답을 제출해야한다.

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

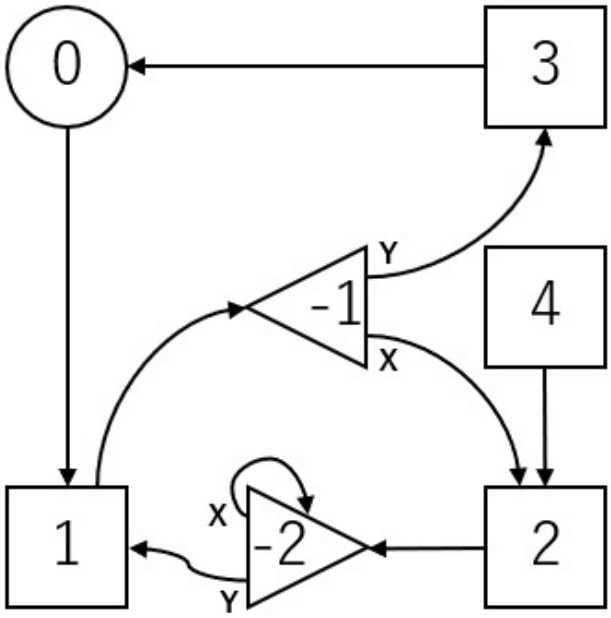
- `C`: 크기 $M+1$인 배열이다. 일련번호가 $i (0 \leq i \leq M)$인 장치의 출구는 일련번호가 `C[i]`인 장치의 진입구에 연결된다.
- `X`, `Y`: 동일한 크기의 배열들이다. 배열의 크기 S는 스위치의 개수이다. 일련번호가 $-j$($1 \leq j \leq S$)인 스위치의 출구 'X'는 일련번호가 `X[j - 1]`인 장치의 진입구에 연결된다. 마찬가지로, 출구 'Y'는 일련번호가 `Y[j - 1]`인 장치의 진입구에 연결된다.
- `C`, `X`, `Y` 의 모든 원소는 $-S$이상 M이하의 정수이어야 한다.
- S는 최대 $400\,000$이어야 한다.
- 이 함수는 정확히 한 번 호출되어야 한다.
- `C`, `X`, `Y`로 표현된 회로는 문제의 조건을 모두 만족해야 한다.

위의 조건들 중 하나라도 만족되지 않으면 평가는 **`Wrong Answer`**이다. 모든 조건이 만족되면 평가는

Accepted이며 점수는 S에 따라 계산된다 (Subtask마다 다름).

Example

$M = 4$, $N = 4$, $A = [1, 2, 1, 3]$라고 하자. 그레이더는 `create_circuit(4, [1, 2, 1, 3])`를 호출한다.



위의 그림은 `answer([1, -1, -2, 0, 2], [2, -2], [3, 1])`에 의해 표현된 회로이다. 그림의 숫자는 일련번호들이다.

두개의 스위치가 사용되었으므로 $S = 2$이다.

초기에 일련번호가 -1과 -2인 두 스위치의 상태는 모두 'X'이다.

공은 다음과 같이 움직인다:

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

- 공이 처음에 스위치 -1에 들어왔을 때, 스위치의 상태는 'X'이다. 따라서, 공은 트리거 2로 들어간다. 스위치 -1의 상태는 'Y'로 바뀐다.
- 공이 두번째로 스위치 -1에 들어왔을 때, 스위치의 상태는 'Y'이므로, 공은 트리거 3으로 간다. 스위치 -1의 상태는 'X'로 다시 바뀐다.

공이 최초로 시작에 다시 돌아올 때까지 트리거 $1, 2, 1, 3$를 순서대로 지난다. 모든 스위치의 상태는 'X'이다. P의 값은 4이다. 회로는 문제의 조건을 만족한다.

압축된 첨부 패키지 파일의 `sample-01-in.txt`는 이 예제에 대응한다. 다른 입출력 예제도 이 패키지에 포함되어 있다.

Constraints

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

Subtasks

각 테스트 케이스에 대한 점수와 제한은 아래와 같다.

1. (2 points) 각 $i (1 \le i \le M)$에 대해서, i는 $A_0, A_1, \ldots, A_{N-1}$에 최대 1번 등장한다.
2. (4 points) 각 $i (1 \le i \le M)$에 대해서, i는 $A_0, A_1, \ldots, A_{N-1}$에 최대 2번 등장한다.
3. (10 points) 각 $i (1 \le i \le M)$에 대해서, i는 $A_0, A_1, \ldots, A_{N-1}$에 최대 4번 등장한다.
4. (10 points) $N = 16$
5. (18 points) $M = 1$
6. (56 points) 추가적인 제한이 없다.

각 테스트 케이스에 대해서, 당신의 프로그램이 **Accepted**인 것으로 판정되었다면 점수는 S의 값에 따라 다음과 같이 계산된다:

- $S \le N + \log_2 N$인 경우 배정된 점수를 모두 받는다.
- Subtasks 5, 6에 속한 테스트 케이스의 경우, $N + \log_2 N < S \le 2N$인 경우 부분 점수를 받는다. 정확한 점수는 $0.5 + 0.4 \times \left(\dfrac{2N - S}{N - \log_2 N} \right)^2$를 테스트 케이스에 배정된 점수에 곱한 값이다.
- 나머지 경우 점수는 0이다.

한 subtask에 대한 점수는 해당 subtask에 속한 테스트 케이스들에서 받은 점수 중 최소값임에 주의하라.

Sample grader

샘플 그레이더는 다음 형식으로 입력을 받는다.

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

샘플 그레이더는 3개의 출력을 생성한다.

첫번째로, 샘플 그레이더는 `out.txt` 파일에 당신의 대답을 저장한다.

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

두번째로, 샘플 그레이더는 공의 움직임을 시뮬레이트한다. 샘플 그레이더는 공이 지나간 장치들의 일련번호를 순서대로 `log.txt`에 저장한다.

세번째로, 샘플 그레이더는 당신의 프로그램에 대한 판정을 표준 출력으로 출력한다.

- 당신의 프로그램이 **Accepted**로 판정되었다면, 샘플 그레이더는 S와 P를 다음 형식으로 출력한

다: `Accepted: S P`.

- 당신의 프로그램이 **`Wrong Answer`**로 판정되었다면, 샘플 그레이더는 다음을 출력한다: `Wrong Answer: MSG`. `MSG`의 내용은 다음 중 하나이다:
  - `answered not exactly once`: 함수 `answer`가 정확히 한번 호출되지 않았다.
  - `wrong array length`: `C`의 크기가 $M + 1$이 아니거나, `X`와 `Y`의 크기가 같지 않다.
  - `over 400000 switches`: S가 $400\,000$보다 크다.
  - `wrong serial number`: `C`, `X`, `Y`의 원소 중에 $-S$보다 작거나M보다 큰 값이 있다.
  - `over 20000000 inversions`: 공이 $20\,000\,000$번의 스위치 상태 변화가 일어난 이후에도 시작으로 돌아오지 않았다.
  - `state 'Y'`: 공이 시작으로 돌아왔을 때 상태가 'Y'인 스위치가 있다.
  - `wrong motion`: 공이 트리거들을 지나가는 순서가 A에 주어진 것과 다르다.

당신의 프로그램이 `Wrong Answer`로 판정되었을 때 `out.txt`나 `log.txt` 파일이 만들어지지 않을 수도 있음에 주의하라.