Hieroglyphs

Un grupo de investigadores esta estudiando las similitudes entre secuencias(sequences) de jeroglificos. Representan cada jeroglifico con un entero no-negativo. Para realizar su estudio, usan los siguientes conceptos sobre secuencias.

Para un secuencia fija A, la secuencia S es llamada **subsecuencia** (**subsequence**) de A si y solo si S puede ser obtenida removiendo algunos elementos(posiblemente ninguno) de A.

La siguiente tabla muestra algunos ejemplos de subsecuencias de una secuencia $A = [3, 2, 1, 2]$.

Subsequence	Como puede ser obtenida de A
[3, 2, 1, 2]	Ningun elemento eliminado.
[2, 1, 2]	[~~3~~, 2, 1, 2]
[3, 2, 2]	[3, 2, ~~1~~, 2]
[3, 2]	[3, ~~2~~, ~~1~~, 2] or [3, 2, ~~1~~, ~~2~~]
[3]	[3, ~~2~~, ~~1~~, ~~2~~]
[ ]	[~~3~~, ~~2~~, ~~1~~, ~~2~~]

Por otro lado, $[3, 3]$ o $[1, 3]$ no son subsecuencias de A.

Consider two sequences of hieroglyphs, A and B. Considere dos secuencias de jeroglificos, A y B. Una secuencia S es llamada **subsecuencia común** (**common subsequence**) de A y B si y solo si S is una subsecuencia tanto de A y B. Además, decimos que una subsecuencia U es una **subsecuencia comun universal** (**universal common subsequence**) de A y B si y solo si las siguientes dos condiciones se cumplen:

- U es una subsecuencia comun de A y B.
- Cada subsecuencia comun de A y B es tambien una subsecuencia de U.

Se puede demostrar que dos secuencias cualesquiera A y B tienen a lo mucho una subsecuencia comun universal.

Los investigadores econtraron dos secuencias de jeroglificos A y B. La secuencia A consiste de N jeroglificos y la secuencia B consiste de M jeroglificos.

Ayuda a los investigadores a calcular una subsecuencia comun universal de las secuencias A y B, o determinar que dicha secuencia no existe.

Detalles de implementación

Debera implememtar la siguiente funcion.

```
std::vector<int> ucs(std::vector<int> A, std::vector<int> B)
```

- A: arreglo de tamaño N describiendo la primera secuencia.
- B: arreglo de tamaño M describiendo la segunda secuencia.
- Si existe una subsecuencia común universal de A y B, El procedimiento debe retornar un array que contenga esta secuencia. En cualquier otro caso, el procedimiento debe retornar $[-1]$ (un arreglo de tamaño 1, cuyo único elemento sea -1).
- Este procedimiento se llama exactamente una vez para cada caso de prueba.

Restricciones

- $1 \leq N \leq 100\,000$
- $1 \leq M \leq 100\,000$
- $0 \leq A[i] \leq 200\,000$ para cada i tal que $0 \leq i < N$
- $0 \leq B[j] \leq 200\,000$ para cada j tal que $0 \leq j < M$

Subtareas

Subtarea	Puntaje	Restricciones Adicionales
1	3	$N = M$; tanto A como B consisten de N enteros **distintos** entre 0 y $N - 1$ (inclusive)
2	15	Para cada entero k, (la cantidad de elementos de A iguales a k) más (la cantidad de elementos de B iguales a k() es a lo mucho 3.
3	10	$A[i] \leq 1$ para todo i tal que $0 \leq i < N$; $B[j] \leq 1$ para todo j tal que $0 \leq j < M$
4	16	Existe una subsecuencia común universal de A y B.
5	14	$N \leq 3000$; $M \leq 3000$
6	42	Sin restricciones adicionales.

Ejemplos

Ejemplo 1

Considere la siguiente llamada.

```
ucs([0, 0, 1, 0, 1, 2], [2, 0, 1, 0, 2])
```

En este caso, la subsecuencias comunes de A y B son las siguientes: $[\,]$, $[0]$, $[1]$, $[2]$, $[0,0]$, $[0,1]$, $[0,2]$, $[1,0]$, $[1,2]$, $[0,0,2]$, $[0,1,0]$, $[0,1,2]$, $[1,0,2]$ y $[0,1,0,2]$.

Ya que $[0,1,0,2]$ es una subsecuencia común de A y B, y todas las subsecuencias comunes de A y B son subsecuencias de $[0,1,0,2]$, la función debe retornar $[0,1,0,2]$.

Ejemplo 2

Considere la siguiente llamada.

```
ucs([0, 0, 2], [1, 1])
```

En este caso, la única subsecuencia común de A y B es la secuencia vacía $[\,]$. Por lo tanto la función debe retornar un arreglo vacío $[\,]$.

Ejemplo 3

Considere la siguiente llamada.

```
ucs([0, 1, 0], [1, 0, 1])
```

En este caso, las subsecuencias comunes de A y B son $[\,]$, $[0]$, $[1]$, $[0,1]$ y $[1,0]$. Se puede demostrar que una subsecuencia común universal no existe. Por lo tanto, la función debe retornar $[-1]$.

Evaluador de Ejemplo (Grader)

Formato de entrada:

```
N  M
A[0]  A[1]  ...  A[N-1]
B[0]  B[1]  ...  B[M-1]
```

Formato de salida:

```
T
R[0]  R[1]  ...  R[T-1]
```

En este caso, R es el arreglo retornado por `ucs` y T es su longitud.