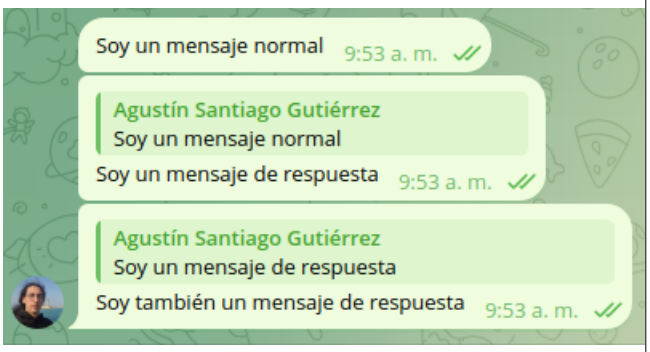
Reordenando mensajes

Contribución de Agustín Santiago Gutiérrez

Descripción del problema

María participa de un chat grupal utilizando su aplicación de mensajería instantánea favorita, la OIApp. En el grupo se discuten un montón de cosas mediante *mensajes*. La aplicación muestra los mensajes ordenados de arriba hacia abajo, supuestamente en orden cronológico.

Cada mensaje puede ser o bien un *mensaje normal*, o bien un *mensaje de respuesta*. Un mensaje de respuesta m_2 se crea en la aplicación seleccionando cualquier mensaje m_1 preexistente, y tipeando la correspondiente respuesta. Decimos que m_1 es el *mensaje citado* en el mensaje de respuesta m_2. Los mensajes normales no tienen ningún mensaje citado.



María sospecha que la OIApp no le muestra los mensajes en el verdadero orden cronológico en el que se enviaron, ya que ha observado en múltiples ocasiones que un mensaje no tenga ningún sentido en el momento en que se recibe, pero que luego lleguen mensajes adicionales haciendo que el anterior repentinamente tenga sentido. De hecho, este tipo de comportamiento es habitual por los impredecibles retrasos en los paquetes enviados por internet, que pueden hacer que un mensaje llegue antes que otro que se envió con una mayor anterioridad.

Sin embargo, María jamás ha visto que un mensaje de respuesta llegue antes que su mensaje citado, y por lo tanto sabe que la aplicación nunca invierte el orden relativo entre un mensaje de respuesta y su mensaje citado. Es decir, **se sabe que un mensaje de respuesta aparecerá después de su mensaje citado**.

María posee el registro detallado de toda la historia de conversación en el grupo, al cual envió exactamente dos mensajes A y B. Como la OIApp del teléfono de María no reordena los mensajes que ella misma envía, **se sabe que la OIApp le mostrará el mensaje A antes que el B**.

Aparte de lo explicado en los dos párrafos anteriores, no se sabe nada más acerca del posible orden de los mensajes.

Debes escribir una función que dados todos los mensajes del grupo, calcule la cantidad total de ordenamientos posibles de los mensajes en la OIApp, de acuerdo a las reglas explicadas. Notar que esta cantidad podría ser 0 si la entrada no es compatible con ningún ordenamiento. Como la respuesta puede ser muy grande, se debe retornar únicamente el resto de dividir al total por $10^9 + 7$.

Notar que como María no confía en el orden en el que la OIApp muestra los mensajes, no se preocupó por ordenarlos de ninguna manera en particular, por lo cual el orden de los mensajes recibidos y su numeración en la entrada para tu programa son completamente arbitrarios.

Descripción de la función

Se debe implementar una función `chat(A,B,C)`. Sus parámetros son:

- `A,B`: Enteros que indican el primer y el segundo mensaje que envió María, respectivamente.
- `C`: Arreglo de N enteros, que indican el mensaje citado por cada mensaje. $C[i]$ es el mensaje citado por el mensaje i, o bien -1 si el mensaje i es un mensaje normal. $0 \leq i < N$

La función debe retornar un único entero: la cantidad total de ordenamientos posibles, módulo $10^9 + 7$.

Evaluador local

El evaluador local lee de la entrada estándar con el siguiente formato:

- Tres enteros N, A, B
- N enteros $C[i]$

Escribe en la salida estándar el entero retornado por la función `chat`.

Restricciones

$2 \leq N \leq 3000$

$0 \leq A, B < N$

$A \neq B$

$-1 \leq C[i] < N$

$C[i] \neq i$

Ejemplos

Si la entrada es:

```
6 0 5
-1 0 -1 1 2 -1
```

La salida correcta es:

```
45
```

Si en cambio la entrada es:

```
4 0 1
-1 -1 0 2
```

La salida sería:

```
3
```

Y si fuera:

```
4 0 1
2 0 1 2
```

La salida sería:

```
0
```

Subtareas

1. $N \leq 8$ (4 puntos)
2. $N \leq 18$ (4 puntos)
3. $C[B] = -1$ (16 puntos)
4. $N \leq 40$ (16 puntos)
5. $N \leq 200$ (16 puntos)
6. Sin más restricción (44 puntos)