Los Brazos de Shor

Problema

Te es dado un entero N y una permutación P del los números de 0 a $N-1$. Hay un robot llamado Shor (el hermano mayor del robot Sho). Tiene dos brazos que le ayudan a ordenar arreglos. En un movimiento, Shor puede elegir dos parejas de enteros (i, j) y (k, h), con $0 \leq i \neq j, k \neq h \leq N-1$, e intercambiar los valores de las posiciones de cada pareja. Es decir, después de la operación, el valor $P[i]$ pasa a la posición j, el valor de $P[j]$ pasa a la posición i, el valor de $P[k]$ pasa a la posición h y el valor $P[h]$ pasa a la posición k. Shor tiene dos restricciones al elegir los índices en sus movimientos. No puede elegir una pareja que solo consista de un índice distinto. Tampoco puede elegir dos parejas exactamente iguales en el mismo movimiento.
Shor quiere ordenar la permutación dada, en la menor cantidad de movimientos. Ayuda a Shor a determinar si esto es posible, y además, la cantidad mínima de movimientos necesaria para lograrlo.
El proceso es un poco más complicado. Debes calcular la respuesta, no solamente para el arreglo original, si no también para el arreglo después de Q actualizaciones. En cada actualización, te dan dos enteros $0 \leq i \neq j \leq N-1$, y se intercambian los valores en $P[i]$ y $P[j]$. $P[i]$ pasa a la posición j y $P[j]$ pasa a la posición i. Las actualizaciones son acumulativas.

Detalles de Implementación

Debes implementar la función *El_Robot_Shor()*. Esta función recibe un entero N el tamaño de la permutación, un entero Q la cantidad de actualizaciones, y 3 vectores, p y u, v. El vector p tiene N elementos, y representa el estado original de la permutación. Los vectores u, v tienen Q elementos, y representan los índices a intercambiar en cada actualización. Esta función debe regresar un vector de tamaño $Q+1$ (llamemos dicho vector ans), $ans[i]$ debe ser igual a la cantidad mínima de movimientos necesarios para ordenar el arreglo después de la i-ésima actualización, o -1 si es imposible ($ans[0]$ es la respuesta para el arreglo original, antes de ninguna actualización). La función se vería así:

```
#include <bits/stdc++.h>
using namespace std;

vector<int> El_Robot_Shor(int N, int Q, vector<int> p,
    vector<int> u, vector<int> v) {
    // Implementa esta función.
}
```

El evaluador llamará la función **múltiples** veces por caso.

Ejemplos

Ejemplo 1:

- El evaluador llama la función

  $$\textit{El\_Robot\_Shor}(4,\ 4,\ \{0,\ 1,\ 2,\ 3\},\ \{0,\ 1,\ 2,\ 0\},\ \{3,\ 3,\ 3,\ 3\})$$

  la permutación es $\{0, 1, 2, 3\}$. Después de cada actualización se ve así

Índice	Actualización	Permutación actual
0	$(0, 3)$	$\{3, 1, 2, 0\}$
1	$(1, 3)$	$\{3, 0, 2, 1\}$
2	$(2, 3)$	$\{3, 0, 1, 2\}$
3	$(0, 3)$	$\{2, 0, 1, 3\}$

- El vector que debe regresar la función es $\{0, -1, 1, -1, 1\}$, pues:
  - El arreglo original está ordenado.
  - Después de la primera actualización, se puede demostrar que es imposible hacer una secuencia de movimientos que ordene el arreglo.
  - Después de la segunda actualización, el arreglo puede ser ordenado con un movimiento, eligiendo las parejas $(0, 1)$ y $(1, 3)$.
  - Después de la tercera actualización, se puede demostrar que es imposible hacer una secuencia de movimientos que ordene el arreglo.
  - Después de la cuarta actualización, el arreglo puede ser ordenado con un solo movimiento, eligiendo las parejas $(0, 1)$ y $(1, 2)$.
- También obtendría la mitad de los puntos que represente este caso responder con un vector como $\{0, -1, 0, -1, 0\}$.

Ejemplo 2:

- El evaluador llama la función

  $$\textit{El\_Robot\_Shor}(10,\ 7,\ \{2,\ 3,\ 0,\ 6,\ 5,\ 1,\ 9,\ 8,\ 7,\ 4\},\ \{0,\ 1,\ 9,\ 7,\ 8,\ 2,\ 0\},\ \{3,\ 6,\ 3,\ 9,\ 3,\ 5,\ 3\})$$

- El vector que debe regresar la función es $\{-1, 4, -1, 4, -1, 4, -1, 4\}$.

Consideraciones

- $1 \leq N, Q \leq 2 \times 10^5$.

- El vector p tendrá exactamente N elementos.
- Los vectores u, v tendrán exactamente Q elementos.
- Para cada $0 \leq i \leq N-1$, se cumple que $0 \leq p[i] < N$.
- Para cada $0 \leq i \leq Q-1$, se cumple que $0 \leq u[i] \neq v[i] < N$.
- Se garantiza que el vector p es una permutación de los números de 0 a $N-1$.
- Sea S_N la suma de los valores de N para cada llamada de la función en un caso de prueba. se garantiza que $S_N \leq 2 \times 10^5$.
- Sea S_Q la suma de los valores de Q para cada llamada de la función en un caso de prueba. se garantiza que $S_Q \leq 2 \times 10^5$.

Subtareas

- (10 puntos) $N, S_N, Q, S_Q \leq 4$.
- (20 puntos) Para todo $0 \leq i \leq N-1$, se cumple que $p[i] = i$.
- (30 puntos) $N, S_N, Q, S_Q \leq 2000$.
- (40 puntos) Sin restricciones adicionales.

Además, para cada subtarea, si tu programa determina exitosamente si es posible o no ordenar la permutación (responde -1 cuando no es posible y un entero positivo cuando sí), obtendrás la mitad de los puntos en la subtarea correspondiente.