Shor's Arms

Problem

You are given an integer N and a permutation P of the numbers 0 through $N-1$. There is a robot called Shor (the older brother of the robot Sho). It has two arms that help it sort arrays. In one move, Shor can choose two pairs of integers (i, j) and (k, h), with $0 \leq i \neq j, k \neq h \leq N-1$, and swap the values at the positions of each pair. That is, after the operation, the value $P[i]$ goes to position j, the value of $P[j]$ goes to position i, the value of $P[k]$ goes to position h, and the value of $P[h]$ goes to position k. Shor has two restrictions when choosing the indices in its moves. It cannot choose a pair that consists of only one distinct index. It also cannot choose two identical pairs in the same move.

Shor wants to sort the given permutation in the fewest number of moves. Help Shor determine if this is possible, and if so, the minimum number of moves needed to achieve it.

The process is a bit more complicated. You must calculate the answer not only for the original array, but also for the array after Q updates. In each update, you are given two integers $0 \leq i \neq j \leq N-1$, and the values at $P[i]$ and $P[j]$ are swapped. $P[i]$ goes to position j and $P[j]$ goes to position i. The updates are cumulative.

Implementation Details

You must implement the function *El_Robot_Shor()*. This function receives an integer N the size of the permutation, an integer Q the number of updates, and 3 vectors, p and u, v. The vector p has N elements and represents the original state of the permutation. The vectors u, v have Q elements and represent the indices to be swapped in each update. This function must return a vector of size $Q+1$ (let's call this vector ans), $ans[i]$ should be equal to the minimum number of moves needed to sort the array after the i-th update, or -1 if it is impossible ($ans[0]$ is the answer for the original array, before any updates). The function would look like this:

```
#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) {
    // Implement this function.
}
```

The grader will call the function **multiple** times per case.

Examples

Example 1:

- The grader calls the function

  El_Robot_Shor(4, 4, {0, 1, 2, 3}, {0, 1, 2, 0}, {3, 3, 3, 3})

  the permutation is $\{0, 1, 2, 3\}$. After each update it looks like this:

Index	Update	Current Permutation
0	$(0, 3)$	$\{3, 1, 2, 0\}$
1	$(1, 3)$	$\{3, 0, 2, 1\}$
2	$(2, 3)$	$\{3, 0, 1, 2\}$
3	$(0, 3)$	$\{2, 0, 1, 3\}$

- The vector the function should return is $\{0, -1, 1, -1, 1\}$, because:
  - The original array is sorted.
  - After the first update, it can be shown that it is impossible to make a sequence of moves that sorts the array.
  - After the second update, the array can be sorted with one move, choosing the pairs $(0, 1)$ and $(1, 3)$.
  - After the third update, it can be shown that it is impossible to make a sequence of moves that sorts the array.
  - After the fourth update, the array can be sorted with a single move, choosing the pairs $(0, 1)$ and $(1, 2)$.
- You would also get half the points for this case by answering with a vector like $\{0, -1, 0, -1, 0\}$.

Example 2:

- The grader calls the function

  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})

- The vector the function should return is $\{-1, 4, -1, 4, -1, 4, -1, 4\}$.

Limits

- $1 \leq N, Q \leq 2 \times 10^5$.
- The vector p will have exactly N elements.

- The vectors u, v will have exactly Q elements.
- For each $0 \leq i \leq N-1$, it holds that $0 \leq p[i] < N$.
- For each $0 \leq i \leq Q-1$, it holds that $0 \leq u[i] \neq v[i] < N$.
- It is guaranteed that the vector p is a permutation of the numbers from 0 to $N-1$.
- Let S_N be the sum of the values of N for each function call in a test case. It is guaranteed that $S_N \leq 2 \times 10^5$.
- Let S_Q be the sum of the values of Q for each function call in a test case. It is guaranteed that $S_Q \leq 2 \times 10^5$.

Subtasks

- (10 points) $N, S_N, Q, S_Q \leq 4$.
- (20 points) For all $0 \leq i \leq N-1$, it holds that $p[i] = i$.
- (30 points) $N, S_N, Q, S_Q \leq 2000$.
- (40 points) No additional restrictions.

Additionally, for each subtask, if your program successfully determines whether it is possible to sort the permutation (returns -1 when it is not possible and a positive integer when it is), you will receive half the points in the corresponding subtask.