1511C - Yet Another Card Deck

Idea: **BledDest**

Tutorial

1511C - Yet Another Card Deck

Let's look at one fixed color. When we search a card of such color, we take the card with minimum index and after we place it on the top of the deck it remains the one with minimum index.

It means that for each color we take and move the same card — one card for each color. In other words, we need to keep track of only k cards, where k is the number of colors ($k \le 50$). As a result, if pos_c is the position of a card of color c then we can simulate a query in the following way: for each color c such that $pos_c < pos_{t_j}$ we increase pos_c by one (since the card will move down) and then set $pos_{t_j} = 1$. Complexity is $O(n + qk)$.

But, if we look closely, we may note that we don't even need array pos_c. We can almost manually find the first card of color t_j and move it to the first position either by series of swaps or, for example, using `rotate` function (present in C++) and it will work fast.

Why? Let's look at one color c. For the first time it will cost $O(n)$ operations to search the corresponding card and move it to the position 1. But after that, at any moment of time, the position of the card won't exceed k, since all cards before are pairwise different (due to the nature of queries). So, all next moves the color c costs only $O(k)$ time.

As a result, the complexity of such almost naive solution is $O(kn + qk)$.

Solution (Neon)

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
  int n, q;
  scanf("%d%d", &n, &q);
  vector<int> a(n);
  for (int& x : a) scanf("%d", &x);
  while (q--) {
    int x;
    scanf("%d", &x);
    int p = find(a.begin(), a.end(), x) - a.begin();
    printf("%d ", p + 1);
    rotate(a.begin(), a.begin() + p, a.begin() + p + 1);
  }
}
```