https://atcoder.jp/contests/abc279/tasks/abc279_e

20mins

如果是直接暴力，大模拟

复杂度为：$O(M^2)$

发现一些性质。

操作数在它之前，那么1后退一步。
操作数就是本身，后退一步。
1初始在首位。

解决方法

反过来想：
大佬代码在这里。（jly的代码）

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

using i64 = long long;

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    int N, M;
    std::cin >> N >> M;

    std::vector<int> A(M);
    for (int i = 0; i < M; i++) {
        std::cin >> A[i];
        A[i]--;
    }

    std::vector<int> P(N), Q(N), Pi(N);
    std::iota(P.begin(), P.end(), 0);
    Q = Pi = P;

    for (int i = M - 1; i >= 0; i--) {
        std::swap(Q[A[i]], Q[A[i] + 1]);
    }
    //第一个问题，这里先对一个逆反的操作。
    Q
    for (int i = 0; i < M; i++) {
        std::swap(Q[A[i]], Q[A[i] + 1]);
```

```
29          std::cout << Q[P[0]] + 1 << "\n";
30          std::swap(P[Pi[A[i]]], P[Pi[A[i] + 1]]);
31          std::swap(Pi[A[i]], Pi[A[i] + 1]);
32      }
33      return 0;
34  }
```

$question\quad 1$:——定义出来的三个结构的意义是什么?

Q, P, Pi

$question\quad 2$:一些非常特别的操作。

为什么先做一个逆向的操作?

ans=$Q[P[0]]+1$。

日常破防时刻:

太抽象了，这一类问题。
一点进展都没有，一天下来什么都做不了。
每次比赛都是突破不了自己。
最终无法站得更加高。

日常破完防之后，继续认真想:

突然发现，当时1得方向是不能简单计算得到得。但是可以计算，操作之后各个位置最终出现得情况。

如果知道了1在哪一个位置，也就可以知道了进行了后面的操作之后得位置了。

$Q(k,\ i)$为经过后面一大段变化之后，i位置上的元素最终出现的位置。

$P(k,\ i)$为初始状态下进行前若干次次操作之后，i的位置。

关于怎么计算这两个函数的问题:

对于$Q(i)$可以发现。过程中某一个状态的B的情况。如果当前的，如果当前$P(k-1,i)$为真第k此操作下，两个元素互相改变位置。通过过程模拟记录，可以得到哪两个数字在进行交换位置。于是$swap(P(Pi(k,A[k]),Pi(k,A[k]+1)))$，得到当前状态下，每一个元素的位置。

对于$P(k,i)$，假设最终状态为$1,2....n$发现，逆着进行n次操作之后。得到的一个序列为 $B_1,B_2....B_n$。

结论如下： B_k所保存的数字，就是当前位置上元素在经过后面的操作之后，来到的位置。

证明如下:

假设$B_{i,j}$表示当前的情况是第 i次 操作后的情况。

当 i=1： 最小规模，显然成立。
若对于更小规模的结果，满足上述关系。对于下一步规模更大的情况。体会一下也是成立的。 🙂