《魔术师》解题报告

浙江省诸暨中学 孟煜皓

1 题意简述

给定一个 1 到 n 的排列 p，每次操作可以翻转一个长度为 m 的区间，请你构造一组解使得最终排列有序。

2 数据范围与评分方式

对于全部的数据，$1 \le m \le n \le 1000$。对于有解的数据，保证排列 p 在有解的排列集合中随机生成。

每个测试包附有参数 $lim_1, lim_2, \ldots, lim_5$。设一个测试包的分值为 S，对于该测试包内的测试点，评分方式如下：

- 若该测试点无解：
  - 若选手输出有解，得 0 分。
  - 否则得 S 分。
- 若该测试点有解：
  - 若选手输出无解或选手的方案不合法，得 0 分。
  - 否则设选手的方案操作次数为 k，则得分为

$$\frac{S}{5} \times \sum_{i=1}^{5} [k \le lim_i]$$

该测试包的得分为测试包内每个测试点得分的最小值。

测试包编号	$n \le$	$m \in$	$lim =$	分值
1	10	$[1, n]$	$\{50, 200, 500, 2000, 10000\}$	5
2			$\{120, 300, 1000, 3000, 10000\}$	
3	30		$\{1000, 30000, 60000, 300000, 1000000\}$	
4			$\{2500, 50000, 110000, 300000, 1000000\}$	
5	50		$\{3000, 25000, 150000, 500000, 1000000\}$	15
6			$\{6000, 50000, 300000, 700000, 1500000\}$	5
7	200		$\{20000, 50000, 200000, 500000, 1000000\}$	
8			$\{40000, 100000, 300000, 700000, 1500000\}$	
9	1000	$[4, 5]$	$\{70000, 150000, 300000, 700000, 1500000\}$	
10			$\{100000, 200000, 400000, 800000, 1500000\}$	
11		$[6, 8]$	$\{50000, 100000, 200000, 500000, 1000000\}$	
12			$\{60000, 120000, 250000, 500000, 1000000\}$	

（续表）

测试包编号	$n \leq$	$m \in$	$lim =$	分值
13	1000	$[40, 60]$	$\{10000, 25000, 50000, 200000, 1000000\}$	5
14			$\{12000, 30000, 60000, 200000, 1000000\}$	
15		$[1, n]$	$\{450000, 700000, 950000, 1250000, 1500000\}$	15
16			$\{1000000, 1200000, 1600000, 2200000, 2500000\}$	5

表 1: 各测试包的附加限制

另外，对于编号为奇数的测试包，保证 m 为奇数；对于编号为偶数的测试包，保证 m 为偶数。

3 解题思路

3.1 m 为偶数的情况

3.1.1 shift 操作

首先我们考虑使用 reverse 操作（翻转）实现 lshift2 操作（循环左移 2 格）

$$(1, 2, 3, \ldots, m+1) \Rightarrow$$
$$(1, m+1, \ldots, 3, 2) \Rightarrow$$
$$(3, \ldots, m+1, 1, 2)$$

和 rshift2 操作（循环右移 2 格）

$$(1, \ldots, m-1, m, m+1) \Rightarrow$$
$$(m, m-1, \ldots, 1, m+1) \Rightarrow$$
$$(m, m+1, 1, \ldots, m-1)$$

对于 m 是偶数的情况，$m+1$ 是奇数，所以可以通过上述操作，实现对长度为 $m+1$ 区间的 shift 操作（任意循环移位）。

3.1.2 还原 $m+1$ 到 n

接下来考虑从后往前，将每个数归位。假设现在 $i+1$ 到 n 已经归位，我们要将 i 归位。设 i 所在位置为 x，每进行一次以 x 为左端点的 reverse 都会使得 x 变成 $x+m-1$。所以若 $x+m-1 \leq i$，我们只要一直执行上述操作即可。然后对区间 $[i-m, i]$ 循环右移 $i-x$ 格，完成归位。

该算法可以归位 $m+1$ 到 n 的所有数，期望所需次数大约为

$$\frac{n^2}{4m}+\frac{(n-m)m}{2}$$

事实上，在可用空间较大时，我们可以直接采用 reverse 操作进行最终的归位。考虑先通过一次或两次 reverse 使 x 变为 $i-m+1$（次数由 x 和 i 的奇偶性决定），然后进行一次 reverse 放到 i，完成归位。

该过程在 $i>2m$ 时一定可行，所以当 $n>2m$ 时，我们可以将期望次数降为大约

$$\frac{n^2}{4m}+\frac{5}{2}(n-2m)+\frac{m^2}{2}$$

3.1.3 可还原的长度下界与 shift3 操作

在上一小节中，我们有了一个将 n 减小至 m 的算法。但是显然我们并不能将 n 直接减小至 m，这会导致剩下部分无法还原。

我们尝试找到一个下界 $n=m+k$，使得 $n>m+k$ 时可还原的充要条件与 $n=m+k$ 时的充要条件相同。

注意到在 $m \bmod 4=0$ 时，reverse 操作不会改变逆序对数量的奇偶性。也就是说，在 $m \bmod 4=0$ 时，逆序对数为偶数是排列可还原的必要条件。

对于三个不同的位置 (x_0,x_1,x_2)，如果我们能够找到一种操作方案，使得 x_i 位置的值变为 $x_{(i+1) \bmod 3}$ 位置的值（即对这三个位置进行循环移位操作），那么只要用这样的操作，就可以遍历所有与原排列逆序对奇偶性相同的所有排列。我们称这个操作为 shift3 操作。

首先我们考虑构造出对特定位置的 shift3 操作：

$$\begin{aligned}&(1,2,\ldots,m-1,m,m+1,m+2)\Rightarrow\\&(m,m+1,1,2,\ldots,m-1,m+2)\Rightarrow\\&(m,2,\ldots,m-1,m+2,m+1,1)\end{aligned}$$

上述过程对 $(1,m,m+2)$ 执行了 shift3 操作。

然后我们考虑将 x_0,x_1,x_2 分别移动到 $1,m,m+2$。首先使用两次 shift 操作将 x_0,x_2 移动到 $1,2$。接下来对现在的 x_1 进行分类讨论：

- $x_1=m+1$。直接对左端点为 1 的区间往左 shift 一格即可使得 $x_1=m,x_2=m+2$。
- $x_1=m+2$。

$$\begin{aligned}&(x_0,x_2,\ldots,\#,\#,x_1)\Rightarrow\\&(\#,\#,x_0,x_2,\ldots,x_1)\Rightarrow\\&(\#,\ldots,x_1,\#,x_0,x_2)\Rightarrow\\&(x_0,\#,\ldots,x_1,\#,x_2)\end{aligned}$$

- $x_1 \le m$。对左端点为 1 的区间 shift 使得 x_1 到 $m+1$。此时 x_0, x_2 一定仍然是相邻两个且 $x_0 \ge 2$。所以对左端点为 2 的区间 shift 使得 x_1 到 $m+2$，然后将 x_0, x_2 移回 $1, 2$。此时变为了 $x_1 = m+2$ 的情况。

然后只需要执行一次对 $(1, m, m+2)$ 的 shift3 操作，最后把上面为了移动执行的操作还原即可。

如果我们每次使用 shift3 操作还原一个数，最后会剩下两个数。由于逆序对数量为偶数，最后这两个数一定已经归位。

注意到 shift3 操作只用到了 $m+2$ 个数，所以我们可以取 $n = m+2$，一定可以还原。

根据实现，上述算法的期望操作次数大约在 $7m$ 到 $9m$ 左右。

对于 $m \bmod 4 = 2$ 的情况，只需要在执行上述算法前，将逆序对数量调整成偶数即可。

至此，我们证明了当 m 为偶数，$n \ge m+2$ 时，排列可以还原当且仅当逆序对数量奇偶性满足限制。同时，我们有了一个基于 shift3 操作的算法。

3.1.4 对基于 shift3 操作的算法的改进

注意到，在执行 shift3 操作时，主要的操作开销在于将第三个数移动到指定位置。但事实上，我们每次还原一个数，只需要用到两个位置而非三个位置。换而言之，第三个位置是任意的。

另外，上一节中对特定位置的 shift3 操作，也可以支持对 $(1, 3, m+2)$ 进行 shift3 操作，还可以花费更多次 reverse 来实现对 $(1, 2, m+2)$ 和 $(1, m+1, m+2)$ 的 shift3 操作。

考虑从前往后依次将每个数归位。假设 1 到 $i-1$ 已归位，我们现在需要将 i 归位。假设 i 所在的位置为 x。我们先将 i 移动到 1，然后将 x 移动到 $m+2$。此时已归位的数（即 1 到 $i-1$）一定仍然是连续的一段。

当 $i \le m-2$ 时，位置 3 和位置 m 中至少有一个不是已归位的数。当 $m-2 < i \le m$ 时，位置 2 和位置 $m+1$ 中至少有一个不是已归位的数。所以我们一定可以找到一个合适的位置来进行特定位置的 shift3 操作。

最后将一开始的两次为移动执行的 shift 操作还原即可。

该算法在最坏情况下的操作次数约为 $3m^2$，期望操作次数约为 $2m^2$。

3.1.5 更优秀的算法

上述算法的主要开销在于还原，即将已归位的数放回原来的位置。事实上，我们只要保证已归位的数始终是连续的一段即可。

考虑将已归位的数放在序列的末尾。假设现在需要将 i 放到这些数的后面。设当前 i 所在位置为 x，我们对 x 分类讨论：

- $x = 1$。将左端点为 2 的区间左移一格，然后将左端点为 1 的区间左移一格。此时 i 已经被接在已归位的数的后面，再对左端点为 2 的区间右移一格即可。
- $x > 1$。对左端点为 2 的区间 shift 使得 x 到 $m+2$。此时已归位的数仍然连续，对左端点为 1 的区间进行 shift 将这些数放在 i 的前面即可。

事实上，对于 $x = 1$ 的情况，若除 i 以外未归位的数超过两个，后面两次 shift 操作均可以改为移位两格，次数可以从 $3m$ 降至 $m+4$。

所以我们只要把 3 到 $m+2$ 的数依次使用上述算法归位，由于保证了逆序对奇偶性，剩下的 1 和 2 也一定已经归位。

该算法在最坏情况下的操作次数为 $(m-1)(m+4)+3m = m^2+6m-4$。忽略最后可能出现的 $x = 1$，期望操作次数约为 $m(m+1)+3\ln m$。最后有 $\frac{1}{3}$ 的概率出现 $x = 1$，此时操作次数需要加 $2m$。

3.1.6 总结

总结一下，当 m 为偶数时，根据 n 的值分类讨论：

- $n = m$。只能执行对整个序列的 reverse 操作。
- $n = m+1$。首先执行至多一次 reverse 操作，然后只能使用 shift 操作。
- $n \geq m+2$。我们先将 $m+3$ 到 n 的数归位，然后对前 $m+2$ 个数使用上一节的算法归位，次数大约为

$$\frac{n^2}{4m} + \frac{5}{2}\max(n-2m, 0) + \frac{m \cdot \min(n-m, m)}{2} + m^2 + 3m + 3\ln m$$

该算法可以通过所有 m 为偶数的测试包。

3.2 m 为奇数的情况

m 为奇数时，相比偶数的情况，最大的区别在于，一个数所在位置的奇偶性始终不会改变。所以接下来我们只考虑所有奇数位置是奇数、偶数位置是偶数的排列。

我们仍然可以用 3.1.1 中的方法构造 shift 操作，但是要求循环移位的长度为偶数。

虽然 shift 操作的长度有限制，由于有位置与数的奇偶性限制，我们仍然可以用与 3.1.2 同样的方法还原 $m+1$ 到 n。

3.2.1 $n = m+2$

我们可以将一个排列表示成 $\begin{pmatrix} p_1 & p_3 & p_5 & \cdots \\ & p_2 & p_4 & \cdots \end{pmatrix}$ 的形式。

一次 reverse 操作对下半部分的影响是一次长度为 $\lfloor \frac{m}{2} \rfloor$ 或 $\lfloor \frac{m}{2} \rfloor + 1$ 的 reverse。这意味着下半部分能变成的排列一定可以通过至多一次 reverse，然后进行任意长度的 shift 操作得到。

通过对 $m \bmod 8$ 的所有可能分类讨论，我们发现，若 $m > 3$，且下半部分保持不变，则上半部分的逆序对数量奇偶性也保持不变。这意味着当 $n = m + 2$ 且 $m > 3$ 时，排列可以还原的充要条件为同时满足：

- 下半部分可以还原（即至多一次 reverse 操作和任意长度的 shift 操作）。
- 用任意操作还原下半部分后，上半部分的逆序对数量为偶数。

现在的问题在于如何还原上半部分。

仍然考虑 3.1.3 中的 shift3 操作，该操作在 m 为奇数时仍可行。所以我们只需要用与 3.1.4 几乎相同的算法还原上半部分即可。

3.2.2 可还原的长度下界

类似 m 为偶数的情况，我们仍然考虑找到一个下界 $n = m + k$，使得 $n > m + k$ 时可还原的充要条件与 $n = m + k$ 时的充要条件相同。

在 $m \bmod 8 = 1$ 时，上下两部分的逆序对数量奇偶性均不会改变；在 $m \bmod 8 = 5$ 时，整个排列的逆序对数量奇偶性不会改变。

在 $n = m + 2$ 时我们已经有了对上半部分的 shift3 操作，同样地，在 $n = m + 3$ 时，我们可以对下半部分进行 shift3 操作。这两个操作是独立的，即不会对另一部分产生影响。

所以我们仍然可以使用类似的方法，在一开始将上下两部分的逆序对数量均调整成偶数，然后分别使用 shift3 操作还原上下两部分即可。

至此，我们证明了当 m 为奇数，$n \geq m + 3$ 时，排列可以还原当且仅当上下两部分的逆序对数量奇偶性满足对应限制。

3.2.3 基于 shift3 操作的算法流程

事实上，我们可以先使用 3.1.5 中的算法，在不考虑上半部分的情况下，还原下半部分。

综上所述，我们有了一个较为优秀的算法，大致流程如下：

1. 特殊处理 $n = m$、$n = m + 1$ 以及 $m = 3$ 的情况。
2. 还原 $m + 4$ 到 n。
3. 根据 $n = m + 2$ 还是 $n \geq m + 3$ 选取不同的算法还原下半部分。
4. 使用基于 shift3 的算法还原上半部分。

但是这个算法仍然不够优秀。

3.2.4 建立绑定关系

如果我们能对所有 $2i-1$ 和 $2i$ 建立绑定关系（即使得 $2i-1$ 和 $2i$ 始终相邻），那么我们就可以直接使用 3.1.5 中的算法还原整个排列。

考虑将已建立绑定关系的数放在序列的末尾（不包括 $m+2$ 及以后的部分），每次将 p_2-1 和 p_2 建立绑定关系。我们首先找到 p_2-1 所在的位置 x，分类讨论：

- $x=m+2$。此时只要将以 2 为左端点的区间往左 shift 两格即可。
- 否则，我们先将以 1 为左端点的区间往左 shift 直到 $x=1$，然后将以 2 为左端点的区间往右 shift 相等的步数使 p_2 复原。此时 p_2-1 与 p_2 已经相邻，我们只要再将以 1 为左端点的区间往左 shift 两格，将这一对数放到序列末尾即可。

该算法执行过程中，上半部分和下半部分的逆序对数奇偶性的变化相同，所以只需要保证上半部分和下半部分的逆序对数奇偶性相同即可。

另外，该算法同时适用于 $n=m+2$ 和 $n=m+3$。

该算法可能会改变上下两部分的逆序对数奇偶性，但这只取决于 m，所以只需要在执行该算法前根据需要对原排列进行调整即可。

事实上，若实现优秀，当 $n=m+3$ 时，尽管该算法和建立绑定关系后使用 3.1.5 中的算法均会改变上下两部分的逆序对数奇偶性，但两个算法的影响恰好互相抵消，所以我们只需要一开始将上下两部分的逆序对数均调整成偶数即可。设 $z=\frac{m-1}{2}$，两个算法总的期望操作次数约为

$$8\sum_{i=1}^{z}\frac{1}{i}\sum_{x=0}^{i-1}\min(x,z+1-x)\approx\left(\frac{3}{4}-\frac{\ln 2}{2}\right)m^2\approx 0.403m^2$$

3.2.5 总结

当 m 为奇数时，算法大致流程如下：

1. 判断每个位置上的数是否与该位置奇偶性相同。
2. 特殊处理 $n=m$、$n=m+1$ 以及 $m=3$ 的情况。
3. 还原 $m+4$ 到 n。
4. 根据 n 是否等于 $m+2$ 以及上下两部分的逆序对数奇偶性，判断是否有解并对原排列进行一定的调整。
5. 建立绑定关系。
6. 使用 3.1.5 中的算法还原整个排列。

在 $n\geq m+3$ 时，该算法的期望操作次数约为

$$\frac{n^2}{4m}+\frac{m\cdot\min(n-m,m)}{2}+0.403m^2$$

该算法可以通过所有 m 为奇数的测试包。

4 参考资料

无。