不老梦（dream）

$n, m \leq 7$

直接暴力搜索出所有的可能排列即可，时间复杂度 $O(n^{m+1})$。

$Q = 1$

我们只关心最后 v 的位置，设 $f_{i,j}$ 表示操作了前 i 次，$pos_v = j$ 的方案数。

容易优化到 $O(nm)$。

$n, m \leq 500$

考虑将序列看成环，那么每次其实就是环反向然后转动一下。

所以有用的状态数只有 $O(n)$。

暴力 dp 可以做到 $O(n^2m + Q)$。

$n, m \leq 5000$

前缀和优化一下即可。

牵丝戏（puppet）

$n \leq 100$

点数只有 $3n$，直接跑 Floyd，复杂度 $O(n^3 + Q)$。

$n \leq 2000$

对于每一个询问 BFS，时间复杂度 $O(nQ)$。

$\text{popcount}(t_i) = 3$

。有手就行。

$\text{popcount}(t_i) = 1$

。有手就行。

~~送了 60 (~~

$n \leq 10^5, Q \leq 5 \times 10^5$

考虑倍增或者线段树。

倍增：设 $f_{i,j,k,l}$，表示从 (i, j) 这个点到 $(i + 2^j, k)$ 这个点的最短路，预处理之后可以单次 $O(\log n)$ 回答询问。

时间复杂度 $O((n + Q) \log n)$。

其实本质上线段树维护的东西是一样的。

$n \le 10^6, Q \le 10^7$

倍增啥的应该过不了吧（

考虑回到 BFS，其实这个本质上是一个 dp。

对于一层的点，状态数量只有 6 种，其中两个点 dis 相同，另一个大/小 1。

又因为这些状态的后继是确定的，建出状态的转移树，那么对于初始状态我们只需要查询它一直跳后继会跳到哪。

只需要实现一个判断一个点是否是另一个点祖先即可。时间复杂度 $O(n+Q)$。

是风动（wind）

$n \le 6$

留给？做法。

里面有一个 $n=1$ 的测试点，大家注意不需要输出任何东西。

$n \le 10$

枚举操作数量然后直接搜索去重即可。

$n \le 18$

其实状态不是很多，如果能用比较厉害的搜索搜出有效状态也许可以通过。

$v_i = 1$

求方案数。

设 $f_{i,0/1}$ 表示考虑前 i 次操作，第 i 次时候操作可以得到多少种排列。

那么，转移 $f_{i,0}$，那么我们 $i+1$ 这一次操作是否操作其实并不重要，没有影响。

$f_{i+1,0} \leftarrow f_{i,0}$

$f_{i+1,1} \leftarrow f_{i,0}$。

转移 $f_{i,1}$，我们需要考虑 $i+1$ 这一次操作操作的话比 i 操作的早还是晚。

$f_{i+1,1} \leftarrow 2f_{i,1}$

$f_{i+1,0} \leftarrow f_{i,1}$。

$n \le 100$

我们需要观测到，如果 $a_j = i+j$，那么这一段的操作先后顺序就被确定了，也就是操作 $[j, i+j-1]$ 有严格的偏序关系。

考虑将其看成折线图，对于一个 i 求答案只需要计算有多少段长度恰好为 i 的下降段。

设 $f_{i,j,k}$ 表示考虑了前 i 个操作，有 j 个长度为要求长度的段，现在这个段长度为 k。

加上外面的枚举，复杂度 $O(n^4)$，应该不会有人写这个（

$n \le 500$

可以前缀和优化前面的，应该不会有人写这个（

$n \le 2000$

上面的 dp 的 $\sum j$ 是 $O(n \ln n)$ 级别的。

所以可以做到 $O(n^2 \ln n)$。

$n \le 8000$

考虑二项式反演，枚举 i，之后钦定恰好 j 段的方案数，求出 $f_{i,j}$ 之后直接二项式反演即可。

求 $f_{i,j}$ 容易做到 $O(n \ln n)$，之后二项式反演复杂度为 $\sum \frac{n^2}{i^2} = O(n^2)$。

注意到 $i = 1$ 的时候都需要特殊求解，但是容易做到 $O(n^2)$。

腐草为萤（firefly）

$n \le 500$

直接模拟即可。

$n \le 2000$

也是容易的。

$n \le 4 \times 10^4$

可以直接莫队+线段树维护，常数比较好应该可以过。

$n \le 2 \times 10^5$

留给 $2 \log$ 做法或者其他复杂度比较高的做法。

$n \le 10^6$

考虑建线段树，对于节点 $[l, r]$ 我们预先将 $L = l$ 且 $R = r$ 的情况处理出来，并维护每一个 a_i 的值，这一部分容易做到 $O(n \log n)$ 的复杂度。

考虑一次查询，我们拿出 $O(\log n)$ 个线段树上节点，那么我们现在只关心跨过不同节点之间的线段，考虑节点顺序排列是 $p_1, p_2, p_3 \dots p_k$，我们只需要知道 p_i 以某一个位置为左端点区间右边最远能到哪，以及以 p_i 为左端点最左能到哪，容易发现相当于我们只需要再额外考虑 $O(\log n)$ 条线段的情况。

用栈可以维护线段的覆盖情况并统计答案。

时间复杂度 $O((n + Q) \log n)$。