## 排列

先来考虑一个排列可到达的条件是什么。

如果不是 n 排列，而是 01 序列的话，那么条件是显然的：只要对于任意 i，序列 b 的第 i 个 1 都位于序列 a 的第 i 个 1 的右边（不一定严格），那么 a 就可以到达 b。

对于一个 n 排列 a，以及一个数 k，把 a 中大于 k 的数标为 1，剩下的数标为 0，就能得 到一个 01 序列。如果对于任意的 k，排列 a 对应的 01 序列都能够到达排列 b 对应的序列，那么排列 a 就可以到达排列 b。

它的必要性是显然的。至于充分性，可以观察下面这个移动策略： i 从 n 到 1 的顺序，每次将数字 i 移到它的目标位置，令当前位置为 l，目标位置为 r，当前 (l, r] 区间的最大数字为 a[j]，那么把 a[l] 和 a[j] 交换一下即可。

容易看出这样移动一定是可行的。

那么只要做一个 01 串 DP：

F[i][j]表示到第 i 位，已经用了的集合为 j 的方案数，从一个全 0 的串开始，每次转移是枚举第 i 位放几，即将串中的某个 0 改为 1，最后到达一个全 1 的串，且保证经过的都是合法 01 串。

这个 DP 是 O(n*2^n ) 的。

## Treap

从下到上做。

在每个结点 v 处求出一棵线段树，线段树的下标是一个权值 w，值为：假如 v 上面有一个结点权值是 w，那么那个点能在以 v 为根的子树中获得的最大结点数。

于是我们有两种操作：

(1) 一个区间内加一个数

(2) 多个线段树在每个(线段树的)结点处数值相加

线段树合并。复杂度 O(n log n)。

## Tree

首先可以考虑一下怎么做 k = 1 的情形，即要求同色点都不相邻。

这个可以容斥，将结点 划分成若干个小组，每个小组内都是同色点，并且要求小组里的点连成一个子树，然后再把这些小子树连起来成为整棵树。

因为是容斥，一个小组内如果有奇数条边（偶数个点）就要乘上一个 (−1)。

这里有个小结论，如果 n 个点被拆成小块的大小分别是 n = a1 + a2 + · · ·+ ak，那么连成树的方案数是 n^(k-2)*a1*a2* · · * an，可以用 Prufer 序列证明。

这样只要 DP 如何分组，然后状态里再多记一维当前有几组，就可以了。

然后再看 k > 1，如果预先分好组使得每组形成一个不超过 k 的同色连通块，那么上面的 这个容斥还是可以用的。

所以只要在里面再套一层分组就可以了，相当于要多跑一次 DP。

直接写的复杂度是 O(n^3 ) 的，已经可以 AC 了。

应该可以进一步优化复杂度吧，时间原因咱没有继续往下想，有神犇来分享一下吗