草

> 定义一个自然数序列 $a = a_0, a_1, \ldots, a_k$ 是草的，当且仅当对于所有 $0 \le i \le k$，a_i 恰好为序列 a 中数字 i 出现的次数
>
> 给定一棵 n 个点的无根树，每个点有权值 v_i，请求出有多少个有序点对 (u, v)，使得从 u 到 v 的唯一路径上经过的点的权值构成的序列（包含 u, v）是草的。
>
> $n \le 2 \times 10^5$。

你肯定先打个表啊，手玩不如让计算机帮你玩。打完表点分治一下数一下就好。

$$
\begin{aligned}
&n >= 6: \\
&(n-3)\ 2\ 1\ (0\ 0\ 0\ 0)\ (n-6\times 0s)\ 1\ 0\ 0\ 0
\end{aligned} \tag{1}
$$

回文匹配

> 小 Δ 的字符串水平终于达到了普及组水平！他学会了求一个字符串在另一个字符串中的出现次数，但他意识到了这是为人熟知的原题，不能出到互测里。过了几天，小 Δ 开始学习提高组字符串算法，比如求一个字符串的最长回文子串，但这也是为人熟知的原题。但小 Δ 觉得只需要融合一下这两道题，就不算为人熟知的原题了，他出的题目如下。
>
> 对于一个字符串 $S = a_1 a_2 a_3 \cdots a_k$，定义字符串的长度 $|S| = k$，定义 S 的子串 $[l, r]$ 为 $S[l, r] = a_l a_{l+1} \cdots a_r$，并定义其反转为 $S^R = a_k a_{k-1} \cdots a_1$
>
> 定义一个字符串 S 的回文集合是其中所有回文串的位置的集合，也就是 $P(S) = \{(l, r) \in \mathbb{Z}^2 \mid 1 \le l \le r \le |S|, S[l, r] = (S[l, r])^R\}$
>
> 定义两个字符串 S, T 回文匹配，记为 $S \approx T$，当且仅当 $P(S) = P(T)$
>
> 给定 n 个字符串 S_i，令 $f(i, j)$ 为 S_j 有多少个子串与 S_i 回文匹配，也就是 $f(i, j) = \left|\{(l, r) \in \mathbb{Z}^2 \mid 1 \le l \le r \le |S_j|, S_j[l, r] \approx S_i\}\right|$
>
> 给定 q 次询问，每次给定 i, j，求 $f(i, j)$
>
> 本题有两种输入方式，第一种是直接给出 S_i，第二种是把 S_i 给定为其前面的某一个 S_{f_i} 后面接上一个字符 c_i，见输入格式。
>
> $n, q \le 5 \times 10^5$。

把回文自动机的过程给抽象一下，能得到一个 kmp 或者 AC 自动机一样的模型。

Mergesort Strikes Back

> 一个长度为 n 的随机排列，进行深度为 k 的归并排序（$[1 \ldots n]$ 为第一层），求期望逆序对个数。答案对一个素数取模。$n, k \le 10^5, 10^8 \le mod \le 10^9$。

考虑会把序列分成若干段，段内的逆序数平凡，只要考虑段之间的。显然只有两种长度，只要算三种的段间贡献就够。而这个归并的形式，就是每个前缀最大值后面带着一些小弟参与最终排序。

不妨设两段的长度分别是 n, m，我们枚举点对 $i \in [1, n], j \in [1, m]$ 之间的贡献，显然只用考虑 i, j 之前的 $i+j$ 个数就够了，而 $i+j$ 个数中的最大值在 i 前的概率是 $\frac{i-1}{i+j}$，此时 $p_i < p_j$ 是一半。那么总贡献应该就是 $\frac{1}{2}\left(1-\frac{2}{i+j}\right)$。

Balanced Reversals

> 单次询问给出长为偶数 n 的 01 串 a 和 b，每次可以把一个长度为偶数的前缀翻转，构造至多 $n+1$ 次操作把串 a 变成串 b 或输出 -1 表示无解。$n \le 4000$。

记 00 为 X，01 为 Y^+，10 为 Y^-，11 为 Z。从后往前构造，难在于不一定有合适的 Y。当这样的时候，我们随机一个 Y，多花一步操作把它搞了；而有 $\frac{1}{4}$ 的概率可以少花一步，所以很松，多随几遍能过。

发现上述过程最难在于 Y 的正负性会被频繁改变，我们考虑从前往后构造，维护一个前缀恰好等于目标前缀的翻转，那么任何没有被接入的 Y 的正负性都不会被误伤，我们只要在开始前花一步把 Y^+ 的数量调整至合适就好。

「CTSC2011」字符串重排

> 对于两个字符串 $A = a_1a_2\cdots a_n$ 和 $B = b_1b_2\cdots b_m$，定义其最长公共前缀长度 $\mathrm{LCP}(A, B)$ 如下：
> $\mathrm{LCP}(A, B) = \max\{k|0 \le k \le n, k \le m, a_1a_2\cdots a_k = b_1b_2\cdots b_k\}$。
> 给定 n 个由小写字母组成的两两不同的非空字符串 $S_1, S_2, \cdots, S_n$，对于一个 1 到 n 的排列 $P = (p_1, p_2, \cdots, p_n)$，定义 P 的价值 $W(P)$ 如下：
>
> $$W(P) = \sum_{i=2}^{n}(\mathrm{LCP}(S_{p_{i-1}}, S_{p_i}))^2 \tag{2}$$
>
> 我们设能够产生最大价值的排列为 P_G^*。
>
> 此外，还有 q 个附加任务。对于第 i 个任务，给定两个 1 到 n 之间的不同的整数 X_i 和 Y_i。对于排列 P，若 P 在满足 $W(P) = W(P_G^*)$ 的前提条件之下，同时满足第 X_i 个字符串 S_{X_i} 恰好排在第 Y_i 个字符串 S_{Y_i} 之前， 即 $\mathrm{pos}(S_{X_i}) + 1 = \mathrm{pos}(S_{Y_i})$，其中 $\mathrm{pos}(S_i)$ 表示字符串 S_i 在排列中的位置，则排列 P 还将获得 2^i 的奖励。所有任务的奖励之和称之为总任务奖励。
>
> 我们设能够使得总任务奖励最大的排列为 P_B^*。
>
> 试求：
>
> 1. $W(P_G^*)$，即可能产生的最大价值；
> 2. P_B^*，在保证最大价值前提下，可以使总任务奖励最大的排列。

对于求 $W(P_G^*)$ 的最大值，我们在 Trie 上贪心就好了，对于每个在同一个子树内的点，要求它们在 P 上是连续的一段。不妨把每个带有 $endpos$ 的点建一个虚点，给它挂到原本的点下方，这样每个非叶子结点其本身都是空的。而每个子树对应的数组，就是以某种顺序把其儿子们的数组给拼起来。

$\text{pos}(S_{X_i})+1=\text{pos}(S_{Y_i})$ 相当于要求，u 到 lca 的路径上的点 w，都是以 X_i 结尾，并且 w 是 fa_w 的数组的最后一段。设 DL_u 表示 u 的第一个儿子必须是谁，DR_u 同理；还有在 `lca` 处，X_i 后面紧跟着 Y_i，用链表和并查集就能维护 `lca` 处。怎么快速维护 DL_u, DR_u 上的矛盾呢？

我们记一个 TL_u，表示 u 不能是父亲的第一个结点，然后用树状数组和 `dfs` 序，讨论一些情况，就能 $O(n\log n)$ 维护了；但是事实上，我们可以把 `Trie` 给二度缩点，然后暴力，总复杂度是 $O(n)$ 的。

Send Tree to Charlie

> 给出一棵 N 个节点的树，初始时第 i 个节点上有一个数字 i。
>
> 每次你可以选择一条未被删除的边，交换这条边连接的两个点上的数字后，删除这条边。现给定一个序列 a，第 i 个位置上若不为 0，则代表要求最终第 i 个位置上的数要为 a_i。否则没有特殊要求。试求出，执行 $N-1$ 次操作之后，有多少种最终局面满足序列 a 的要求。答案对 10^9+7 取模。
>
> $2\le N\le 5\times 10^5$，$0\le a_i\le N$。

就跟树上的数一样维护链表就好了。不同在于，这题的每一个限制都需要被满足，所以暴力就好了，复杂度是对的。方案数的话，对每个点分别数一下它的边有多少种可能的顺序就好了。