题解

树(tree)

记录每个点为根能得到的最优答案，如果小于0就不往上传递即可。

最短路 (Path)

朴素转移是 $f_s \xrightarrow{a_{\mathrm{popc}(s\oplus t)}} f_t$，转移数太多，我们要增加中间状态将冗余的信息进行整合处理。

考虑把 s 变到 t 看成逐位确定是否要修改的过程，注意到只需要记录考虑了几位、改了几位、当前修改成的数即可，也就是不用记录 s。

最后的状态数和边数都是 $O(2^k k^2)$ 的，但非零边只有 $O(2^k k)$ 条，所以复杂度实际上是 $O(2^k k^2)$ 的。

字符串 (string)

首先我们按串所包含的每种字符的个数分类，显然不同类的两个串无法通过操作使得这两个串变得相同，于是我们可以直接处理出不同类之间的串的 f 的总贡献，现在只需要考虑同一类里面的串之间的贡献。

假设 S,T 是同一类的两个串，我们发现：（令 $m=|s_i|$）

- 若 $S=T$，那么 $f(S,T)=0$。
- 若存在 S,T 的相同前缀 $S[1,l-1]=T[1,l-1]$ 和相同后缀 $S[r+1,m]=T[r+1,m]$，且满足 $S[l,r]$ 是有序的，那么我们进行一次操作把 $T[l,r]$ 排序即可，此时 $f(S,T)=1$。
- 其余情况我们可以直接把 S,T 各排一遍序，$f(S,T)=2$。

这个时候你就可以暴力枚举 S,T 并暴力 $O(m)$ 判断了，时间复杂度 $O(n^2m)$，期望得分 20 分。

如果我们知道第一种情况的数量和第二种情况数量，那么就可以知道第三种情况的数量，并计算出这一类里面的贡献了。

第一种情况的数量很好算，使用哈希即可，或者在下面介绍的统计第二种情况的算法中一并算出来。对于第二种情况：

对于某一类来说，我们先把这一类里面的所有串排序为 $a_1,a_2,\cdots,a_k$。

排序的一个好处是若两个串 S,T 满足 $f(S,T)=1$，其中需要进行操作的是 T，那么必然 S 排名比 T 靠前。那么我们直接对于每一个 a_i 计算 a_i 与 $a_{i+1},\cdots,a_k$ 的贡献，那么 a_i 是 S 串，$a_{i+1},\cdots,a_k$ 是 T 串。

排序的另外一个好处就是 $a_{i+1},\cdots,a_k$ 中与 a_i 的最长公共前缀（下面称为 LCP）长度相同的串肯定是在连续的一段区间里面。于是我们把 $a_{i+1},\cdots,a_k$ 按他们与 a_i 的 LCP 的长度分为若干段，分段可以用单调栈维护。

具体来说，我们考虑倒叙枚举 i。设 p_j 表示 a_j 与 a_i 的 LCP 长度，由于 $p_{i+1},\cdots,p_k$ 单调不增，而且 i 变小时我们要做的是对所有 p_j 进行取 $\min$ 操作，于是 p_j 变化的是 p_{i+1} 开始的一段区间，于是我们就可以用单调栈维护了。

那么对于每一段 $[L,R]$，$a_L,\cdots,a_R$ 与 a_i 的 LCP 都是一样的，记为 $[1,p]$。那么如果我们找到 a_i 从 $p+1$ 开始的一个极长有序子串 $[p+1,q]$（找的方法可以是预处理+二分），这个区间必然是 $a_L,\cdots,a_R$ 中与 a_i 是第二种情况的串的一种可行操作区间之一，而且是从 $p+1$ 开始的最长的可行操作区间。于是此时我们只需要求出 $a_L,\cdots,a_R$ 中有多少个串与 a_i 有相同的后缀 $[q+1,m]$。

我们可以一开始先把所有 a_i 反过来插入 Trie 树，然后每次在 Trie 树中找到代表后缀 $[q+1,m]$ 的点 u，那么终止节点在 u 子树内的 a_j 都和 a_i 具有相同的后缀 $[q+1,m]$。于是我们对于 Trie 树中的每个点记录一下一开始插入时经过这个点的 a_j 有哪些，然后询问的时候二分编号在 $[L,R]$ 区间的有多少个即可。这样直接记录看起来很暴力，但是注意到插入时一个串顶多会被记录 $O(m)$ 次，总空间复杂度 $O(nm)$，所以不会有问题。

现在来考虑时间复杂度，注意分段关键字与 LCP 长度有关，于是每次至多有 m 段，于是总时间复杂度是 $O(nm(\log n+\log m))=O(nm\log nm)$。