NOIP 2024 模拟赛题解

BSZX

2024 年 9 月 13 日

目录

1 链链链（chain）

考虑贪心。

我们先找出最大值和次大值。由于最后要把所有边删完，所以最大值所在连通块一定会与次大值所在连通块断开，此时贡献一定为最大值 – 次大值。所以不妨先将最大值与次大值断开。那么对于最大值与次大值中间的数，将它们和最大值断开一定更优。

可以使用 ST 表来计算区间最大值，递归地模拟上述过程即可。复杂度 $O(n \log n)$ 。

2 串串串（str）

2.1 算法一

考虑如何根据一个 T 求出字典序最大的 T'。

可以贪心地从前往后扫描字符串，如果该字符还没有出现，则从 z 到 a 进行标号，这样映射的结果是字典序最大的。

枚举所有的子串，暴力转换与比较是 $O(n^3)$，期望得分 10。

2.2 算法二

由刚刚的贪心方法，我们发现该子串一定是原串的后缀串，否则可以通过往后扩展获得字典序更大的答案。

枚举所有的后缀，暴力转换与比较是 $O(n^2)$，期望得分 30。

2.3 算法三

当字符集为 {a} 时，容易得到答案为长度为原串长度的全 z 字符串。

结合算法二可以得到 35。

2.4 算法四

当字符集为 $\{\mathrm{a},\mathrm{b}\}$ 时，考虑快速的比较两个后缀的 T' 的字典序大小。

考虑将连续的相同字符压缩为一个长度，则两个后缀则变成了两个数列。

易得知，贪心求得的 T' 时基于压缩的序列，进行 z 和 y 的交替填充。

因此，我们可以从字符串的字典序比较，变为数列的比较。

可以求出两个数列的最长公共前缀，然后在差异的地方判断字典序大小即可。

结合算法二可以得到 60。

2.5 算法五

考虑算法四启发我们进行拆位，考虑将原串拆分为 26 个 01 字符串，第 i 个字符串第 j 位为 1 当且仅当原串的第 j 为第 i 个小写字母。

我们额外记录两个序列数组 ord_i 和 rk_i，代表从第 i 个字符往后，26 个字母的出现顺序和 26 字母的排名，可以通过从后往前扫描原串以及使用冒泡排序来求得。

这样，比较两个后缀的答案的字典序，则可以通过求出两个后缀答案的最长公共前缀后，查询差异的地方的 rk 来得到顺序。

我们可以通过二分与哈希求得两个后缀答案的最长公共前缀，具体的，考虑两个子串的答案是否相等，可以通过 ord 的顺序依次比较拆位的字符串的哈希值，如果所有字母全部一致则两个子串的答案也相等。

复杂度为 $O(n\Sigma \log_2 n)$，其中 Σ 为字符集大小。

3 字符串（data）

首先考虑如何求 f 函数，对于一个字符串 s，将 A 视作 1，将 B 视作 2，在模 3 意义下相加。排除和为 0 与初始时 ‘AB‘ 交错无法操作的情况，剩下的和为 1 即为 $\{A\}$，和为 2 即为 $\{B\}$。

使用线段树维护每个线段树节点区间会向后贡献和为 $0,1,2$ 的区间个数，与和为 $0,1,2$ 的子区间个数，并维护当前面贡献为 $0,1,2$ 时所造成的贡献。合并是容易的。另外还需要维护交替串数量，只需要维护以 A,B 中一者开头，A,B 中一者的四类串的上述贡献，即区间本身贡献和前面贡献叠加的结果对后面以及答案的贡献。

统计答案时，用和为 x 的区间数量减去和为 x 的交替区间数量就能得到 $\{A\}$ 与 $\{B\}$ 的解，用和为 0 的数量加上和为 1 或 2 的交替区间数量即为 $\varnothing$ 的数量，且容易发现 $\{A,B\}$ 的数量为 0。

4 写文章（write）

由于输入方式的特殊性，考虑建出 Trie 树，在 Trie 树上考虑问题。

考虑进行一个 DP，如果一个字符串不加入词库那么可以直接算出贡献，否则考虑在它最短的能匹配它的前缀处进行贡献。

令 $dp_{x,v}$ 表示 x 的子树内，当前匹配到最短的词库内单词是 v（所在节点），sum_x 表示 x 子树内全部不加入词库的答案，这是好算的。

设 x 的儿子分别是 $s_1, s_2, \cdots, s_k$，假设 v 在 s_{mid} 这个子树内。

对于 $i < mid$，显然 s_i 子树中的任意一个串加入词库都会影响 v，所以只能取 sum_{s_i}。

对于 $i > mid$，该子树内的“词库串”至少花费 dep_{s_i} 个 A，所以可以取 $\min(sum_{s_i}, min_t\{dp_{s_i,t} + (dep_{s_i} \cdot A + B) \cdot c_t\})$。

所以只需要对于每个 i 求出 $min_t\{dp_{s_i,t} + (dep_{s_i} \cdot A + B) \cdot c_t\}$，就可以使用懒标记维护 dp。

发现这是类似斜率优化的形式，需要支持凸包合并，整体向上平移，求 $(dep_{s_i} \cdot A + B)$ 的对应最优值。

可以使用李超线段树合并维护，只需要额外维护截距的懒标记即可。

时间复杂度 $O(n \log n)$，空间复杂度 $O(n)$，复杂度优秀的启发式合并凸包也可通过。