2025 年四川省大学生程序设计竞赛题解

SCCPC2025 solution

hdu 出题组

杭州电子科技大学

2025.6.8

难度

- easy: IFJKH
- middle: ACED
- hard: LBG

I - 本质不同后缀

- 给定 N 个字符串，求这些字符串本质不同的后缀个数。
- $N \leq 3 \times 10^5, \sum |S_i| \leq 3 \times 10^5$。

I - 本质不同后缀

- 每个串翻转过来，变成本质不同的前缀个数。

I - 本质不同后缀

- 每个串翻转过来，变成本质不同的前缀个数。
- 可以通过直接把反串插入 trie 树，输出节点个数 -1 即可，也可以通过别的字符串算法通过本题。

I - 本质不同后缀

- 每个串翻转过来，变成本质不同的前缀个数。
- 可以通过直接把反串插入 trie 树，输出节点个数 -1 即可，也可以通过别的字符串算法通过本题。
- 时间复杂度 $O(\sum |S_i|)$。

F - 逆序对

- 给一个挖空的 01 串，填空后最小化串的逆序对数量。
- $n \leq 10^6, \sum n \leq 2 \times 10^6$。

F - 逆序对

- 贪心的，我们填充的数值必然是一段 1 后一段 0。如果填充的有 0 在 1 前面，则交换它们必然不劣。

F - 逆序对

- 贪心的，我们填充的数值必然是一段 1 后一段 0。如果填充的有 0 在 1 前面，则交换它们必然不劣。
- 枚举填充 1 和 0 的分界，维护当前逆序对数取 max 即可。

F - 逆序对

- 贪心的，我们填充的数值必然是一段 1 后一段 0。如果填充的有 0 在 1 前面，则交换它们必然不劣。
- 枚举填充 1 和 0 的分界，维护当前逆序对数取 max 即可。
- 具体而言可以先默认全填 0 计算出逆序对数，然后从前往后枚举每个需要填充的位置从 0 更改为 1，维护前缀 1 的个数和后缀 0 的个数即可完成对整体逆序对数的维护。

F - 逆序对

- 贪心的，我们填充的数值必然是一段 1 后一段 0。如果填充的有 0 在 1 前面，则交换它们必然不劣。
- 枚举填充 1 和 0 的分界，维护当前逆序对数取 max 即可。
- 具体而言可以先默认全填 0 计算出逆序对数，然后从前往后枚举每个需要填充的位置从 0 更改为 1，维护前缀 1 的个数和后缀 0 的个数即可完成对整体逆序对数的维护。
- 更进一步的，一个位置如果填 1 则前面全填 1，填 0 则后面全填 0。那么实际上，我们可以根据其填 1 时前面 1 的个数和填 0 时后面 0 的个数的大小关系来判断填 1 还是填 0。

F - 逆序对

- 贪心的，我们填充的数值必然是一段 1 后一段 0。如果填充的有 0 在 1 前面，则交换它们必然不劣。
- 枚举填充 1 和 0 的分界，维护当前逆序对数取 max 即可。
- 具体而言可以先默认全填 0 计算出逆序对数，然后从前往后枚举每个需要填充的位置从 0 更改为 1，维护前缀 1 的个数和后缀 0 的个数即可完成对整体逆序对数的维护。
- 更进一步的，一个位置如果填 1 则前面全填 1，填 0 则后面全填 0。那么实际上，我们可以根据其填 1 时前面 1 的个数和填 0 时后面 0 的个数的大小关系来判断填 1 还是填 0。
- 时间复杂度 $O(n)$。

J - 四川省赛

- 点权为大写英文字母的树，数树链 SCCPC 的数量。
- $n \leq 10^6, \sum n \leq 2 \times 10^6$。

J - 四川省赛

- 本题的可行做法有很多。

J - 四川省赛

- 本题的可行做法有很多。
- 一个可行的做法是树形 DP，设 $f_{i,j}$ 表示从 i 的子树中某个点到 i 匹配 SCCPC 长度为 j 的前缀的方案数，$g_{i,j}$ 相应表示后缀的方案数。转移和计算答案类似于树直径 DP 一样，细节较多。

J - 四川省赛

- 本题的可行做法有很多。
- 一个可行的做法是树形 DP，设 $f_{i,j}$ 表示从 i 的子树中某个点到 i 匹配 SCCPC 长度为 j 的前缀的方案数，$g_{i,j}$ 相应表示后缀的方案数。转移和计算答案类似于树直径 DP 一样，细节较多。
- 另一个可行的做法是枚举中间点，此时两边均只有两个字符，找每个点出发匹配多少个 CS 和 PC 即可计算前后缀的方案数合并得到中间点确定的答案。注意此时 PC 会算重。

J - 四川省赛

- 本题的可行做法有很多。
- 一个可行的做法是树形 DP，设 $f_{i,j}$ 表示从 i 的子树中某个点到 i 匹配 SCCPC 长度为 j 的前缀的方案数，$g_{i,j}$ 相应表示后缀的方案数。转移和计算答案类似于树直径 DP 一样，细节较多。
- 另一个可行的做法是枚举中间点，此时两边均只有两个字符，找每个点出发匹配多少个 CS 和 PC 即可计算前后缀的方案数合并得到中间点确定的答案。注意此时 PC 会算重。
- 时间复杂度 $O(n)$。

K - 点分治

- 给定排列表示点分治每次分治的中心，问最后点分树上每个点的父亲。
- $n \leq 10^5, \sum n \leq 10^6$。

K - 点分治

- 点分治的过程可以看成，每次在连通块内删除一个点，分裂出来的每个连通块的父亲即这个点，即把分裂出来的每个连通块的根的父亲设为这个点。

K - 点分治

- 点分治的过程可以看成，每次在连通块内删除一个点，分裂出来的每个连通块的父亲即这个点，即把分裂出来的每个连通块的根的父亲设为这个点。
- 正着做很难知道分裂后每个连通块的根，不妨考虑倒着做。

K - 点分治

- 点分治的过程可以看成，每次在连通块内删除一个点，分裂出来的每个连通块的父亲即这个点，即把分裂出来的每个连通块的根的父亲设为这个点。
- 正着做很难知道分裂后每个连通块的根，不妨考虑倒着做。
- 倒着做后，每次枚举到一个新的点 p_i，相当于把这个点加入图中，并且合并其在原树上相邻的当前已经存在的连通块，这些连通块的根的父亲即当前点 p_i，之后 p_i 就成为了合并后连通块的根。

K - 点分治

- 点分治的过程可以看成，每次在连通块内删除一个点，分裂出来的每个连通块的父亲即这个点，即把分裂出来的每个连通块的根的父亲设为这个点。
- 正着做很难知道分裂后每个连通块的根，不妨考虑倒着做。
- 倒着做后，每次枚举到一个新的点 p_i，相当于把这个点加入图中，并且合并其在原树上相邻的当前已经存在的连通块，这些连通块的根的父亲即当前点 p_i，之后 p_i 就成为了合并后连通块的根。
- 并查集实现即可，时间复杂度 $O(\sum n \log n)$。

H - 胡图图

- T 组数据，从点 (x, y) 可以走到 $(x \pm 1, y \pm 1), (x \pm 1, y \pm 2), (x \pm 2, y \pm 1), (x \pm 2, y \pm 2)$，问走到 (X, Y) 的最小步数。
- $T \leq 10^6, |x|, |y|, |X|, |Y| \leq 10^9$。

H - 胡图图

- 两维可以拆开来独立考虑。

H - 胡图图

- 两维可以拆开来独立考虑。
- 会发现如果可以 c 步从 x 走到 X，那么 $c+2, c+4, \cdots$ 都可以（可以来回浪费 2 步）。

H - 胡图图

- 两维可以拆开来独立考虑。
- 会发现如果可以 c 步从 x 走到 X，那么 $c+2, c+4, \cdots$ 都可以（可以来回浪费 2 步）。
- 因此答案为：

$$\min(\max(f(x-X), f(y-Y)), \max(f(x-X), f(y-Y)))$$

H - 胡图图

- 两维可以拆开来独立考虑。
- 会发现如果可以 c 步从 x 走到 X，那么 $c+2, c+4, \cdots$ 都可以（可以来回浪费 2 步）。
- 因此答案为：

$$\min(\max(f(x-X), f(y-Y)), \max(f(x-X), f(y-Y)))$$

- 其中 $f_{1/0}(x)$ 表示走奇/偶步从 0 到 x 的最少操作次数。

H - 胡图图

- 两维可以拆开来独立考虑。
- 会发现如果可以 c 步从 x 走到 X，那么 $c+2, c+4, \cdots$ 都可以（可以来回浪费 2 步）。
- 因此答案为：

$$\min(\max(f(x-X), f(y-Y)), \max(f(x-X), f(y-Y)))$$

- 其中 $f_{1/0}(x)$ 表示走奇/偶步从 0 到 x 的最少操作次数。
- 会发现大部分：

$$f_1(x) = \left\lfloor \frac{x}{2} \right\rfloor + \text{even}\left(\left\lfloor \frac{x}{2} \right\rfloor\right)$$
$$f_0(x) = \left\lfloor \frac{x}{2} \right\rfloor + \text{odd}\left(\left\lfloor \frac{x}{2} \right\rfloor\right)$$

H - 胡图图

- 两维可以拆开来独立考虑。
- 会发现如果可以 c 步从 x 走到 X，那么 $c+2, c+4, \cdots$ 都可以（可以来回浪费 2 步）。
- 因此答案为：

$$\min(\max(f(x-X), f(y-Y)), \max(f(x-X), f(y-Y)))$$

- 其中 $f_{1/0}(x)$ 表示走奇/偶步从 0 到 x 的最少操作次数。
- 会发现大部分：

$$f_1(x) = \left\lfloor \frac{x}{2} \right\rfloor + \text{even}\left(\left\lfloor \frac{x}{2} \right\rfloor\right)$$
$$f_0(x) = \left\lfloor \frac{x}{2} \right\rfloor + \text{odd}\left(\left\lfloor \frac{x}{2} \right\rfloor\right)$$

- 只有 $f(0) = 3$ 是个例，注意特判。

A - 最小乘积

- 每条边有属性 (a_i, b_i)，定义路径权值为：

$$\left(\sum_{i \in P} a_i\right) \times \left(\sum_{i \in P} b_i\right)$$

- 求节点 1 到节点 N 的所有可能路径中，最小的权值是多少。
- $N \leq 300, M \leq 1000, 1 \leq a_i, b_i \leq 200$。

A - 最小乘积

- 不难想到设计 DP 状态 $f_{sum,u}$ 表示目前到达 u 节点且 $\sum a_i = sum$ 时，$\sum b_i$ 的最小值。

A - 最小乘积

- 不难想到设计 DP 状态 $f_{sum,u}$ 表示目前到达 u 节点且 $\sum a_i = sum$ 时，$\sum b_i$ 的最小值。
- 注意到 $a_i > 0$，sum 相同的状态之间没有转移。

A - 最小乘积

- 不难想到设计 DP 状态 $f_{sum,u}$ 表示目前到达 u 节点且 $\sum a_i = sum$ 时，$\sum b_i$ 的最小值。
- 注意到 $a_i > 0$，sum 相同的状态之间没有转移。
- 所以可以直接枚举 sum 再枚举可行的边，直接转移。

A - 最小乘积

- 不难想到设计 DP 状态 $f_{sum,u}$ 表示目前到达 u 节点且 $\sum a_i = sum$ 时，$\sum b_i$ 的最小值。
- 注意到 $a_i > 0$，sum 相同的状态之间没有转移。
- 所以可以直接枚举 sum 再枚举可行的边，直接转移。
- 时间复杂度 $O(NM\max\{a_i\})$。

C - 最优时间

- $$S(x) = \{d \mid d \mid x\} \cup \left\{ kx \mid 2 \le k \le \left\lfloor \frac{N}{x} \right\rfloor \right\}$$
- 假设当前状态是 x，每秒可以决定等概率变成 $S(x)$ 中的一个数，或者保持不变，每秒结束后 $x \leftarrow x - 1$，给定初始状态 x，需要求出最优决策下到 0 的期望时间。
- $N \le 10^5$。

C - 最优时间

- 这个问题直接做并不容易做，考虑迭代做法。

C - 最优时间

- 这个问题直接做并不容易做，考虑迭代做法。
- 假设 $f_i^{(k)}$ 是迭代 k 轮之后 f_i 的值。

C - 最优时间

- 这个问题直接做并不容易做，考虑迭代做法。
- 假设 $f_i^{(k)}$ 是迭代 k 轮之后 f_i 的值。
- 初始时 $f_i^{(0)} = i$，也就是每一秒都保持不变，等待 i 秒之后自动变成 0。

C - 最优时间

- 这个问题直接做并不容易做，考虑迭代做法。
- 假设 $f_i^{(k)}$ 是迭代 k 轮之后 f_i 的值。
- 初始时 $f_i^{(0)} = i$，也就是每一秒都保持不变，等待 i 秒之后自动变成 0。
- 每一轮迭代的式子为：

$$f_i^{(k+1)} = \min\left(f_{i-1}^{(k)}, \frac{\sum_{j \in S(i)} f_{j-1}^{(k)}}{|S(i)|}\right) + 1$$

C - 最优时间

- 这个问题直接做并不容易做，考虑迭代做法。
- 假设 $f_i^{(k)}$ 是迭代 k 轮之后 f_i 的值。
- 初始时 $f_i^{(0)} = i$，也就是每一秒都保持不变，等待 i 秒之后自动变成 0。
- 每一轮迭代的式子为：

$$f_i^{(k+1)} = \min\left(f_{i-1}^{(k)}, \frac{\sum_{j \in S(i)} f_{j-1}^{(k)}}{|S(i)|}\right) + 1$$

- 目标是 $f_{\infty,i}$。

C - 最优时间

- 证明这个迭代是收敛的：

C - 最优时间

- 证明这个迭代是收敛的：
  - 单调不增：$f_i^{(k+1)} \le f_i^{(k)}$，对于每个 i 都成立，考虑 $k=0$ 的时候，是显然的，又因为

$$\min\left(f_{i-1}^{(k)}, \frac{\sum_{j\in S(i)} f_{j-1}^{(k)}}{|S(i)|}\right)+1 \le \min\left(f_{i-1}^{(k-1)}, \frac{\sum_{j\in S(i)} f_{j-1}^{(k-1)}}{|S(i)|}\right)+1$$

    做数学归纳，可知单调不增。

C - 最优时间

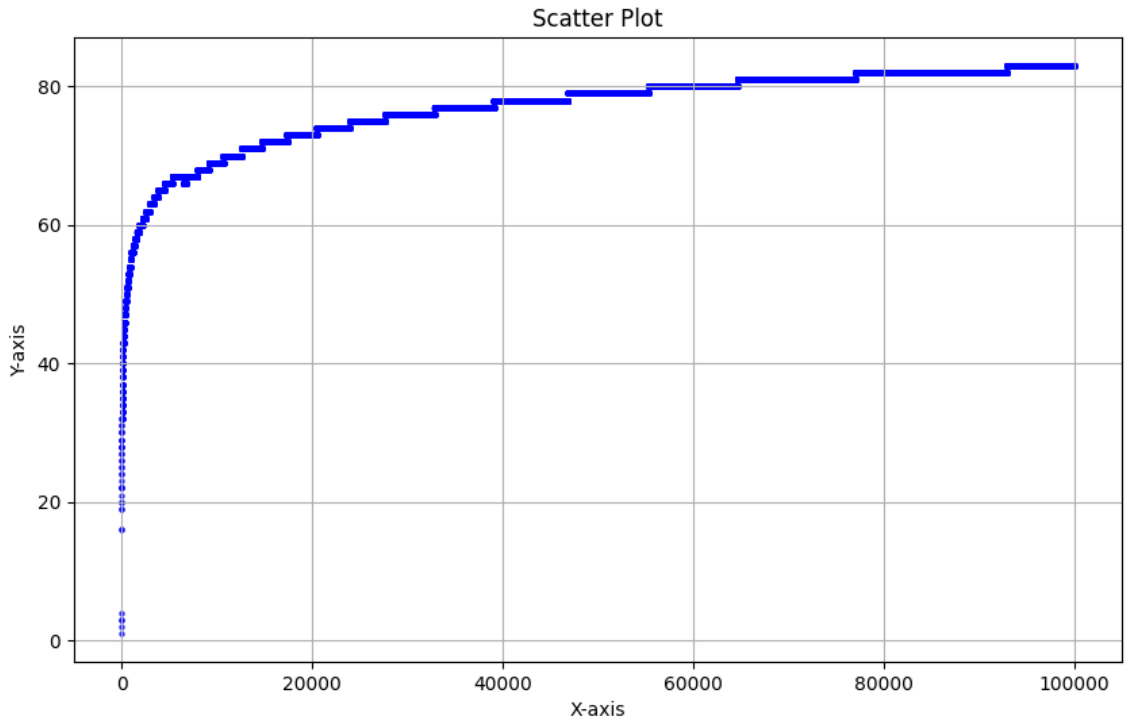
- 证明这个迭代是收敛的：
  - 单调不增：$f_i^{(k+1)} \le f_i^{(k)}$，对于每个 i 都成立，考虑 $k=0$ 的时候，是显然的，又因为

$$\min\left(f_{i-1}^{(k)}, \frac{\sum_{j\in S(i)} f_{j-1}^{(k)}}{|S(i)|}\right)+1 \le \min\left(f_{i-1}^{(k-1)}, \frac{\sum_{j\in S(i)} f_{j-1}^{(k-1)}}{|S(i)|}\right)+1$$

    做数学归纳，可知单调不增。
  - 有下界：期望时间一定大于等于 0。

C - 最优时间

- 在 $1 \le N \le 10^5$ 的收敛到 10^{-6} 的次数如下图所示，最大次数不会超过 100 次，可以迭代 100 次通过本题。



E - 竞赛图

- 求至少存在一个 k 元环的 n 个点竞赛图个数，答案对 998244353 取模。
- $n, k \leq 10^5$。

E - 竞赛图

- 引理 1：对竞赛图缩点之后形成的 DAG 形如一条链。

E - 竞赛图

- 引理 1：对竞赛图缩点之后形成的 DAG 形如一条链。
- 考虑归纳，新加入一个点之后，如果这个点和某一个 SCC 同时拥有出边和入边，那么它就属于这个 SCC，否则可以推出一个点和某一个 SCC 之间的边一定是同向的，那么此时这个点在这条链上一定存在一个分界线，使得前半部分是入边后半部分是出边。

E - 竞赛图

- 引理 2：一个 $n(n \geq 4)$ 阶强连通竞赛图一定存在一个 $n-1$ 阶子强连通竞赛图。

E - 竞赛图

- 引理 2：一个 $n(n \geq 4)$ 阶强连通竞赛图一定存在一个 $n-1$ 阶子强连通竞赛图。
- 考虑归纳，对于任意一个 $n-1$ 阶竞赛图 G，如果其添加一个点 z 之后变成了一个强连通竞赛图，那么：

E - 竞赛图

- 引理 2：一个 $n(n \geq 4)$ 阶强连通竞赛图一定存在一个 $n-1$ 阶子强连通竞赛图。
- 考虑归纳，对于任意一个 $n-1$ 阶竞赛图 G，如果其添加一个点 z 之后变成了一个强连通竞赛图，那么：
- 如果对 G 进行强连通分量的缩点之后，按拓扑序排好，形成的 SCC 为 $v_1, v_2, \cdots, v_m$。

E - 竞赛图

- 引理 2：一个 $n(n \geq 4)$ 阶强连通竞赛图一定存在一个 $n-1$ 阶子强连通竞赛图。
- 考虑归纳，对于任意一个 $n-1$ 阶竞赛图 G，如果其添加一个点 z 之后变成了一个强连通竞赛图，那么：
- 如果对 G 进行强连通分量的缩点之后，按拓扑序排好，形成的 SCC 为 $v_1, v_2, \cdots, v_m$。
  1. $m = 1$ 显然。

E - 竞赛图

- 引理 2：一个 $n(n \geq 4)$ 阶强连通竞赛图一定存在一个 $n-1$ 阶子强连通竞赛图。
- 考虑归纳，对于任意一个 $n-1$ 阶竞赛图 G，如果其添加一个点 z 之后变成了一个强连通竞赛图，那么：
- 如果对 G 进行强连通分量的缩点之后，按拓扑序排好，形成的 SCC 为 $v_1, v_2, \cdots, v_m$。
  1. $m = 1$ 显然。
  2. $m \geq 2$，此时因为新图是强连通图，所以存在 $x \in v_1, y \in v_m$ 满足 $(y, z), (z, x) \in E$，因为 $n \geq 4$，所以删除其它任意一个点，因为 x, y, z 没有被删除。这样显然不会影响其所在 SCC 内部的强连通性，并且也不会影响整体的强连通性。

E - 竞赛图

- 引理 3：强连通 $n(n \geq 3)$ 阶竞赛图存在一条哈密顿回路。

E - 竞赛图

- 引理 3：强连通 $n(n \geq 3)$ 阶竞赛图存在一条哈密顿回路。
- 考虑归纳，$n = 3$ 显然，考虑由 $n - 1$ 阶竞赛图构造出 n 阶竞赛图的哈密顿回路：

E - 竞赛图

- 引理 3：强连通 $n(n \geq 3)$ 阶竞赛图存在一条哈密顿回路。
- 考虑归纳，$n = 3$ 显然，考虑由 $n-1$ 阶竞赛图构造出 n 阶竞赛图的哈密顿回路：
- 如果点 n 和 $[1, n-1]$ 中所有点的连边都是出边/入边的，那么此图不强连通。

E - 竞赛图

- 引理 3：强连通 $n(n \geq 3)$ 阶竞赛图存在一条哈密顿回路。
- 考虑归纳，$n = 3$ 显然，考虑由 $n-1$ 阶竞赛图构造出 n 阶竞赛图的哈密顿回路：
- 如果点 n 和 $[1, n-1]$ 中所有点的连边都是出边/入边的，那么此图不强连通。
- 否则哈密顿路径 p 中存在 i，有 $p_i \to n, n \to p_{i+1}$，那么将 n 插入其中就可以。

E - 竞赛图

- 综上，只需要图中存在一个大小至少为 k 的强连通块即可。

E - 竞赛图

- 综上，只需要图中存在一个大小至少为 k 的强连通块即可。
- 考虑怎么计算大小为 n 的强连通竞赛图个数，记作 f_n，容斥计算不连通的方案数，枚举缩点后链末端连通块大小 i，有

$$f_n = 2^{\binom{n}{2}} - \sum_{i=1}^{n-1} f_i \times 2^{\binom{n-i}{2}}$$

利用分治 NTT 可以快速计算。

E - 竞赛图

- 综上，只需要图中存在一个大小至少为 k 的强连通块即可。
- 考虑怎么计算大小为 n 的强连通竞赛图个数，记作 f_n，容斥计算不连通的方案数，枚举缩点后链末端连通块大小 i，有

$$f_n = 2^{\binom{n}{2}} - \sum_{i=1}^{n-1} f_i \times 2^{\binom{n-i}{2}}$$

  利用分治 NTT 可以快速计算。
- 考虑怎么计算所有强连通块小于 k 的个数，令

$$F(x) = \sum_{i=1}^{k-1} \frac{f_i}{i!} \times x^i$$

  那么方案数为

$$[x^n] \frac{1}{1 - F(x)}$$

D - 三分图

- 对于一个长度为 n 的全排列，以如下方式生成图：
- 对于 $1 \le i < j \le n$，如果 $p_i > p_j$，则有无向边 (i, j)。
- 若生成图为三分图，则是一个好的全排列。
- 给出全排列 p，问有多少个好的全排列 q，其字典序大于 p，答案对 998244353 取模。
- $T, n \le 300$。

D - 三分图

- 首先不可能存在四元组 $1 \leq \alpha < \beta < \gamma < \delta \leq n$，满足 $q_\alpha > q_\beta > q_\gamma > q_\delta$。

D - 三分图

- 首先不可能存在四元组 $1 \leq \alpha < \beta < \gamma < \delta \leq n$，满足 $q_\alpha > q_\beta > q_\gamma > q_\delta$。
- 那么由 Dilworth 引理，整个排列可以被划分成三个单调上升子序列，我们将每个单调上升子序列染成同一种颜色，此时一定是一种合法的三分图染色方案。

D - 三分图

- 首先不可能存在四元组 $1 \leq \alpha < \beta < \gamma < \delta \leq n$，满足 $q_\alpha > q_\beta > q_\gamma > q_\delta$。
- 那么由 Dilworth 引理，整个排列可以被划分成三个单调上升子序列，我们将每个单调上升子序列染成同一种颜色，此时一定是一种合法的三分图染色方案。
- 因此问题等价于求解有多少个全排列 q，可以被划分成三个单调上升子序列。

D - 三分图

- 先不考虑计数，考虑给出全排列 q 如何作判定性问题，初始时我们定义一个长度为 3 的序列 $f = \{0, 0, 0\}$，表示这三个上升子序列当前末尾元素，依次枚举 $i \in [1, n]$，那么 q_i 可以接到这个三元序列中任意一个比它小的位置，从贪心的角度，我们必然选择最大的那个，如果最终可以把 $q_1, \cdots, q_n$，全部安放上去，说明是一个好的序列。

D - 三分图

- 再考虑没有字典序的要求，有多少个长度为 n 的序列。

D - 三分图

- 再考虑没有字典序的要求，有多少个长度为 n 的序列。
- 不妨假设 $f_0 \le f_1 \le f_2$，我们用 $DP(a, b, c)$ 表示剩下的元素中，有 a 个元素 $> f_0$，有 b 个元素 $> f_1$，有 c 个元素 $> f_2$ 时可以填充的方案数。

D - 三分图

- 再考虑没有字典序的要求，有多少个长度为 n 的序列。
- 不妨假设 $f_0 \le f_1 \le f_2$，我们用 $DP(a, b, c)$ 表示剩下的元素中，有 a 个元素 $> f_0$，有 b 个元素 $> f_1$，有 c 个元素 $> f_2$ 时可以填充的方案数。
- 若我们填充了一个 $> f_1$ 的元素，假设它是剩下元素中第 x 大的，那么会转移到 $DP(x-1, b-1, c-1)$。

D - 三分图

- 再考虑没有字典序的要求，有多少个长度为 n 的序列。
- 不妨假设 $f_0 \le f_1 \le f_2$，我们用 $DP(a, b, c)$ 表示剩下的元素中，有 a 个元素 $> f_0$，有 b 个元素 $> f_1$，有 c 个元素 $> f_2$ 时可以填充的方案数。
- 若我们填充了一个 $> f_1$ 的元素，假设它是剩下元素中第 x 大的，那么会转移到 $DP(x-1, b-1, c-1)$。
- 若我们填充了一个 $> f_2$ 的元素，从贪心的角度它必然小于 f_1，假设它是剩下元素中第 x 大的，那么会转移到 $DP(a, x-a-1, c-1)$。

D - 三分图

- 再考虑没有字典序的要求，有多少个长度为 n 的序列。
- 不妨假设 $f_0 \le f_1 \le f_2$，我们用 $DP(a, b, c)$ 表示剩下的元素中，有 a 个元素 $> f_0$，有 b 个元素 $> f_1$，有 c 个元素 $> f_2$ 时可以填充的方案数。
- 若我们填充了一个 $> f_1$ 的元素，假设它是剩下元素中第 x 大的，那么会转移到 $DP(x-1, b-1, c-1)$。
- 若我们填充了一个 $> f_2$ 的元素，从贪心的角度它必然小于 f_1，假设它是剩下元素中第 x 大的，那么会转移到 $DP(a, x-a-1, c-1)$。
- 若我们填充了一个 $> f_3$ 的元素，它只能填充剩下最小的那个元素，且必须大于 f_2，转移到 $DP(a, b, c-1)$。

D - 三分图

- 再考虑没有字典序的要求，有多少个长度为 n 的序列。
- 不妨假设 $f_0 \le f_1 \le f_2$，我们用 $DP(a, b, c)$ 表示剩下的元素中，有 a 个元素 $> f_0$，有 b 个元素 $> f_1$，有 c 个元素 $> f_2$ 时可以填充的方案数。
- 若我们填充了一个 $> f_1$ 的元素，假设它是剩下元素中第 x 大的，那么会转移到 $DP(x-1, b-1, c-1)$。
- 若我们填充了一个 $> f_2$ 的元素，从贪心的角度它必然小于 f_1，假设它是剩下元素中第 x 大的，那么会转移到 $DP(a, x-a-1, c-1)$。
- 若我们填充了一个 $> f_3$ 的元素，它只能填充剩下最小的那个元素，且必须大于 f_2，转移到 $DP(a, b, c-1)$。
- 前两种转移通过前缀和优化后，可以在 $O(V^3)$ 的时间复杂度内预处理整个 DP 数组，其中 V 为值域。

D - 三分图

- 接下来考虑字典序的要求，类比数位 DP 的思想，我们可以枚举 p 的每一个前缀 $p[1:x]$，在后面穿插一个比 $p[x+1]$ 大的元素，后面的后缀任取。此时根据这个前缀，我们可以确定出到当前位置序列 f 的值以及 a, b, c 的具体参数 $O(1)$ 得到答案，时间复杂度为 $O(n^2)$，也可以利用前缀和作进一步优化到 $O(n)$。

D - 三分图

- 接下来考虑字典序的要求，类比数位 DP 的思想，我们可以枚举 p 的每一个前缀 $p[1:x]$，在后面穿插一个比 $p[x+1]$ 大的元素，后面的后缀任取。此时根据这个前缀，我们可以确定出到当前位置序列 f 的值以及 a, b, c 的具体参数 $O(1)$ 得到答案，时间复杂度为 $O(n^2)$，也可以利用前缀和作进一步优化到 $O(n)$。
- 综上，总时间复杂度为 $O(V^3 + \sum_{1 \le i \le T} n_i)$。

L - abc

- 定义区间价值为区间中出现次数最多的字符的出现次数减出现次数最少的字符的出现次数，字符集大小为 3。
- $N \leq 2 \times 10^5$。

L - abc

- 注意，如果字符集大小为 2，我们可以假设 a 为 1，b 为 -1，然后计算前缀和 S，算出 $\sum_{0\le i\le j\le N}|S_i - S_j|$，然后注意如果直接这么计算，全 a/b 区间也会被统计，应当在答案中减掉。

L - abc

- 注意，如果字符集大小为 2，我们可以假设 a 为 1，b 为 -1，然后计算前缀和 S，算出 $\sum_{0\le i\le j\le N}|S_i-S_j|$，然后注意如果直接这么计算，全 a/b 区间也会被统计，应当在答案中减掉。
- 字符集为 3 的做法可以参考字符集为 2，根据：

$$\max(|a-b|,|a-c|,|b-c|)=\frac{|a-b|+|a-c|+|b-c|}{2}$$

可以由字符集为 2 的情况推广到字符集为 3 的情况。

L - abc

- 注意，如果字符集大小为 2，我们可以假设 a 为 1，b 为 -1，然后计算前缀和 S，算出 $\sum_{0\le i\le j\le N}|S_i-S_j|$，然后注意如果直接这么计算，全 a/b 区间也会被统计，应当在答案中减掉。
- 字符集为 3 的做法可以参考字符集为 2，根据：

$$\max(|a-b|,|a-c|,|b-c|)=\frac{|a-b|+|a-c|+|b-c|}{2}$$

  可以由字符集为 2 的情况推广到字符集为 3 的情况。
- 算出的答案包括：同时包含三种字符的区间价值，包含了两种字符的区间价值（但是会把 0 认为是最小值），包含了一种字符的区间价值。

L - abc

- 想要剔除掉某两种字符的区间价值，只需要把这两者出现次数的最小值再减去即可，可以找到极长的包含这两种字符区间：

L - abc

- 想要剔除掉某两种字符的区间价值，只需要把这两者出现次数的最小值再减去即可，可以找到极长的包含这两种字符区间：
  - 按照字符集大小为 2 的做法，我们计算出来的答案是 $\sum(\max - \min)$。

L - abc

- 想要剔除掉某两种字符的区间价值，只需要把这两者出现次数的最小值再减去即可，可以找到极长的包含这两种字符区间：
  - 按照字符集大小为 2 的做法，我们计算出来的答案是 $\sum(\max - \min)$。
  - 注意到 $\sum(\max + \min)$ 其实并不难求，在字符集为 2 的时候恰好是区间长度。

L - abc

- 想要剔除掉某两种字符的区间价值，只需要把这两者出现次数的最小值再减去即可，可以找到极长的包含这两种字符区间：
  - 按照字符集大小为 2 的做法，我们计算出来的答案是 $\sum(\max - \min)$。
  - 注意到 $\sum(\max + \min)$ 其实并不难求，在字符集为 2 的时候恰好是区间长度。
  - 解方程即可知道 $\sum \min$ 的值。

L - abc

- 想要剔除掉某两种字符的区间价值，只需要把这两者出现次数的最小值再减去即可，可以找到极长的包含这两种字符区间：
  - 按照字符集大小为 2 的做法，我们计算出来的答案是 $\sum(\max - \min)$。
  - 注意到 $\sum(\max + \min)$ 其实并不难求，在字符集为 2 的时候恰好是区间长度。
  - 解方程即可知道 $\sum \min$ 的值。
- 剩下的问题只有：求出 $\sum_{0 \le i \le j \le N} |S_i - S_j|$，可以使用排序，计算每个数值对应的系数，可以做到 $O(n \log n)$。

L - abc

- 想要剔除掉某两种字符的区间价值，只需要把这两者出现次数的最小值再减去即可，可以找到极长的包含这两种字符区间：
  - 按照字符集大小为 2 的做法，我们计算出来的答案是 $\sum(\max - \min)$。
  - 注意到 $\sum(\max + \min)$ 其实并不难求，在字符集为 2 的时候恰好是区间长度。
  - 解方程即可知道 $\sum \min$ 的值。
- 剩下的问题只有：求出 $\sum_{0 \le i \le j \le N} |S_i - S_j|$，可以使用排序，计算每个数值对应的系数，可以做到 $O(n \log n)$。
- 注意对于一段长度为 n 的区间，$0 \le \max_i\{S_i\} - \min_i\{S_i\} \le n$，通过这个可以用前缀和优化到 $O(n)$。

B - 三进制

- 交互题。
- 定义一个 n 位三进制的加密算法为，将每一位的 012 分别排列映射。
- 你可以询问不超过 2 次任意 n 位三进制数在加密下的加法结果（带最高位且最高位不加密），得到每一位的排列映射。
- $n \leq 10^5, \sum n \leq 10^6$。

B - 三进制

- 先不管低位进位，那么对于一位加法而言：

$$0 + 1 \equiv 1 \pmod 3$$
$$0 + 2 \equiv 2 \pmod 3$$
$$1 + 2 \equiv 0 \pmod 3$$

B - 三进制

- 先不管低位进位，那么对于一位加法而言：

$$0+1 \equiv 1 \pmod 3$$
$$0+2 \equiv 2 \pmod 3$$
$$1+2 \equiv 0 \pmod 3$$

- 注意到只有 $1+2$ 的结果和原来的两个加数不同，同时存在进位，所以如果确定不存在进位，1 次询问可以区分这种情况和前两种情况。同时不管是哪种情况，我们都可以确定 0 是谁。

B - 三进制

- 如果确定低位进位了，那么对于一位加法而言：

$$1+0+1 \equiv 2 \pmod 3$$
$$1+0+2 \equiv 0 \pmod 3$$
$$1+1+2 \equiv 1 \pmod 3$$

B - 三进制

- 如果确定低位进位了，那么对于一位加法而言：

$$1+0+1 \equiv 2 \pmod 3$$
$$1+0+2 \equiv 0 \pmod 3$$
$$1+1+2 \equiv 1 \pmod 3$$

- 注意到只有 $0+1$ 的结果和原来的两个加数不同，同时不存在进位，所以如果确定存在进位，1 次询问可以区分这种情况和后两种情况。同时不管是哪种情况，我们都可以确定 2 是谁。

B - 三进制

- 那么第一次询问我们可以直接问 $00\cdots0$ 和 $11\cdots1$。可以确定最低位必然不存在低位进位，那么后续的高位是否存在低位进位就可以通过低位结果得到。

B - 三进制

- 那么第一次询问我们可以直接问 $00\cdots0$ 和 $11\cdots1$。可以确定最低位必然不存在低位进位，那么后续的高位是否存在低位进位就可以通过低位结果得到。
- 第二次询问，最低位的 1 和 2 只能通过是否溢出来区分，即：

$$1+1=02$$

$$2+2=11$$

B - 三进制

- 那么第一次询问我们可以直接问 $00\cdots0$ 和 $11\cdots1$。可以确定最低位必然不存在低位进位，那么后续的高位是否存在低位进位就可以通过低位结果得到。
- 第二次询问，最低位的 1 和 2 只能通过是否溢出来区分，即：

$$1+1=02$$

$$2+2=11$$

- 不妨考虑每一位都通过是否溢出来区分剩下两个数字。

B - 三进制

- 对于确定 0 需要区分 1 和 2 的情况：

$$
\begin{aligned}
1+1 &= 02 \\
1+1+1 &= 10 \\
2+2 &= 11 \\
1+2+2 &= 12
\end{aligned}
$$

B - 三进制

- 对于确定 0 需要区分 1 和 2 的情况：

$$1+1=02$$
$$1+1+1=10$$
$$2+2=11$$
$$1+2+2=12$$

- 能够同时区分上面四种情况：如果没有进位就是情况 1；如果有进位且这一位是 0 就是情况 2；如果有进位且这一位是 1/2，则通过是否和加数相同区分情况 3/4。

B - 三进制

- 对于确定 2 需要区分 0 和 1 的情况：

$$0+0=00$$
$$1+0+0=01$$
$$1+1=02$$
$$1+1+1=10$$

B - 三进制

- 对于确定 2 需要区分 0 和 1 的情况：

$$0+0=00$$
$$1+0+0=01$$
$$1+1=02$$
$$1+1+1=10$$

- 能够同时区分上面四种情况：如果有进位就是情况 4；如果没有进位且这一位是 2 就是情况 3；如果没有进位且这一位是 0/1，则通过是否和加数相同区分情况 1/2。

G - 丢番图

- 给定长度为 n 的序列 a 和定值 t，解方程

$$\forall 0 \le j < n, \sum_{i=1}^{n} a_i^j \times x_i \equiv t^j \pmod{998244353}$$

- $n \le 5 \times 10^4$。

G - 丢番图

- 由 Cramer 法则：对于线性方程组 $Ax = b$，若 $\det(A) \neq 0$，则有 $x_i = \frac{\det(A_i)}{\det(A)}$，其中 A_i 为将 A 的第 i 列换成 b。

G - 丢番图

- 由 Cramer 法则：对于线性方程组 $Ax = b$，若 $\det(A) \neq 0$，则有 $x_i = \frac{\det(A_i)}{\det(A)}$，其中 A_i 为将 A 的第 i 列换成 b。
- Vandermonde 行列式：若矩阵 A 满足 $A_{i,j} = x_j^i$，那么 $\det(A) = \prod_{0 \leq i \leq j < n}(x_j - x_i)$。

G - 丢番图

- 由 Cramer 法则：对于线性方程组 $Ax = b$，若 $\det(A) \neq 0$，则有 $x_i = \frac{\det(A_i)}{\det(A)}$，其中 A_i 为将 A 的第 i 列换成 b。
- Vandermonde 行列式：若矩阵 A 满足 $A_{i,j} = x_j^i$，那么 $\det(A) = \prod_{0 \leq i \leq j < n}(x_j - x_i)$。
- 考虑 $F(\alpha) = \prod_{0 \leq i < n}(\alpha - x_i)$，那么令 $P_i(\alpha) = \frac{\prod_{0 \leq j < n}(\alpha - x_j)}{(\alpha - x_i)}$，我们只关心 $P_i(x_i)$，由洛必达法则，答案等于 $F'(x_i)$，预处理出 $F'(\alpha)$，利用多点求值快速计算，时间复杂度为 $O(n \log n)$。

Good Luck & Have Fun !