2025“钉耙编程”中国大学生算法设计暑期联赛（7）题解

By Claris

2025 年 8 月 8 日

1 矩形框选

Shortest judge solution: 2141 Bytes.

假设选出的矩形尺寸是 $w \times h$，固定 w 后 h 越大越好，因此取 $h = \lfloor \frac{k}{w} \rfloor$。随着 w 的增大，h 只有 $2\sqrt{k}$ 种不同的取值，数论分块枚举即可。

确定矩形的尺寸后，使用扫描线算法找到和最大的 $w \times h$ 子矩阵：按照横坐标从上到下滑动大小为 w 的窗口，令 f_i 表示纵坐标在 $[i, i+h-1]$ 的窗口内的点的权值和，那么滑动窗口增删点时，对 f 的影响是一段区间加或区间减，答案为 f 的全局最大值。

不妨设 $w \geq h$，那么 $h \leq \sqrt{k}$，此时时间复杂度为 $\mathcal{O}(nk)$，常数很小已经可以通过，但可以做得更好。将 f 数组分成 $\frac{n}{\sqrt{h}}$ 块，每块大小为 $\sqrt{h}$，对于每一块记录块内 f 的最大值 v，以及整体需要增加的标记值 tag。在这里不维护全局最大值，而是维护历史全局最大值。

遇到区间加 p $(p > 0)$ 的操作时：

- 对于中间完整的块，块内最大值 v 需要加上 p，并更新历史全局最大值，块标记值 tag 也需要加上 p。
- 对于两边零碎的最多两块，暴力修改块内每个元素，并将其值加上块标记值 tag 来更新历史全局最大值。

遇到区间减 p $(p > 0)$ 的操作时：

- 对于中间完整的块，块内最大值 v 需要减去 p，块标记值 tag 也需要减去 p。不需要更新历史全局最大值，因为一定不优。
- 对于两边零碎的最多两块，暴力修改块内每个元素并重算块内最大值 v。同理也不需要更新历史全局最大值。

总时间复杂度为 $\mathcal{O}(nk^{\frac{3}{4}})$。

2 龙族栖息地

Shortest judge solution: 1285 Bytes.

题目给出的其实是六边形网格的立方体坐标表示，在这种表示下，(q_1, r_1, s_1) 到 (q_2, r_2, s_2) 的最短路步数为 $\frac{|q_1-q_2|+|r_1-r_2|+|s_1-s_2|}{2}$。设核心房间的坐标是 $(x, y, -x-y)$，则需要最小化：

$$f(x,y)=\sum_{i=1}^{n}(|x-q_i|+|y-r_i|+|-x-y-s_i|)$$

固定 x 找最优的 y 是经典问题，y 取 $\{r_1, r_2, \ldots, r_n, -x-s_1, -x-s_2, \ldots, -x-s_n\}$ 的中位数，可以通过 `nth_element` 在 $\mathcal{O}(n)$ 时间求出。

另一方面，$f(x,y)$ 求和式的每一项都关于 x, y 二维下凸，因此求和后也是二维下凸，于是三分找到最优的 x 即可。

时间复杂度 $\mathcal{O}(n \log a)$。

3 质疑概率

Shortest judge solution: 2336 Bytes.

二分答案，转化为判断是否能将序列切成恰好 k 个子段，使得每一段的 $\frac{\sum b}{\sum a}$ 都不超过 mid。

如果两个相邻的子段都满足条件，那么将它们合并成一个子段后，新的子段仍然满足条件，这说明可以将“恰好 k 个子段”的约束放宽成“至少 k 个子段”的约束。

令 sa, sb 分别为 a, b 的前缀和，一个子段 $(j, i]$ 满足条件当且仅当 $\frac{sb_i-sb_j}{sa_i-sa_j} \le mid$，即 $sb_i - sa_i \cdot mid \le sb_j - sa_j \cdot mid$。令 $c_i = sb_i - sa_i \cdot mid$，计算序列 $c[0..n]$ 的最长不上升子序列即可，要求 c_0 与 c_n 必选。检查 mid 是否可行的时间复杂度为 $\mathcal{O}(n \log n)$。

由于 mid 是一个既约分数，推荐在 Stern-Brocot Tree 上进行二分。

时间复杂度 $\mathcal{O}(n \log n \log ans)$。

4 梦醒时刻

Shortest judge solution: 1761 Bytes.

设 f_i 表示第 i 个人的实际梦醒时刻，按照 $1, 2, \ldots, n$ 的顺序进行递推，那么计算 f_i 时 $f_1, f_2, \ldots, f_{i-1}$ 都已知。

设 v_j 表示仅考虑所有能作用到第 i 个人的噪音时，时刻 j 合计有多少分贝的噪音。将噪音忍耐力设置为 g_i，依次遍历 $v_1, v_2, \ldots, v_m$ 进行模拟，找到第 k_i 次梦境层数减少的时刻 tmp，则 $f_i = \min(tmp+1, t_i)$。计算出第 f_i 后，新增两个事件：

- 在考虑第 l_i 个人时，需先将 v_{f_i} 异或上 d_i。
- 在考虑第 r_i+1 个人时，需先将 v_{f_i} 异或上 d_i。

用线段树维护 v，线段树上每个节点 x 维护以下信息（假设 x 对应区间 $[l, r]$）：

- $maxv(x) = \max(v_l, v_{l+1}, \ldots, v_r)$
- 令它的两个子节点分别表示 $[l, mid]$、$[mid+1, r]$ $(mid = \lfloor \frac{l+r}{2} \rfloor)$，记录 $delta(x)$ 表示如果初始噪音忍耐力是 $\max(v_l, v_{l+1}, \ldots, v_{mid})$，经过 $[mid+1, r]$ 部分的模拟后，梦

境层数将会减少多少次。定义函数 $query(x, g)$ 表示如果初始噪音忍耐力是 g，经过 x 子树内所有值的模拟后，梦境层数将会减少多少次，那么 $delta(x)$ 其实就等于 $query(x$的右儿子$, x$的左儿子的$maxv)$。

考虑利用单侧递归实现 $query(x, g)$：

- 若 $g \geq maxv(x)$，返回 0。
- 若 $g \geq maxv(x$的左儿子$)$，返回 $query(x$的右儿子$, g)$。
- 否则递归 x 的左儿子，右儿子的结果必定为 $delta(x)$。返回 $query(x$的左儿子$, g) + delta(x)$。

每次 $query(x, g)$ 的时间复杂度为 $\mathcal{O}(\log m)$，单点修改需要重算 $\mathcal{O}(\log m)$ 个节点的 $delta$ 值，合计需要 $\mathcal{O}(\log^2 m)$ 时间。

最后考虑如何找到第 k 次梦境层数减少的时刻 $kth(x, k, g)$，其中 g 表示初始噪音忍耐力，x 表示仅考虑 x 的子树：

- 若 x 是叶子，结果显然。
- 令 $cnt = query(x$的左儿子$, g)$。
- 若 $k \leq cnt$，返回 $kth(x$的左儿子$, k, g)$。
- 否则返回 $kth(x$的右儿子$, k - cnt, \max(g, x$的左儿子的$maxv))$。

每次 $kth(x, k, g)$ 将调用 $\mathcal{O}(\log m)$ 次 $query(\cdot, \cdot)$，合计时间复杂度也为 $\mathcal{O}(\log^2 m)$。

总时间复杂度 $\mathcal{O}(n \log^2 m)$。

5 地图编辑器

Shortest judge solution: 762 Bytes.

按照树根所在行号升序绘制每棵树，此时后绘制的优先级一定更高，直接覆盖即可。

6 伤害冷却比

Shortest judge solution: 685 Bytes.

固定技能施放次数 $Y = \lfloor \frac{K}{X \cdot N} \rfloor + 1$ 时，显然 X 越大越好，当 X 取 $\frac{K}{N(Y-1)}$ 时取到最大值 $F(Y) = \frac{K}{N}(1 + \frac{1}{Y-1})$。其中，$F(Y)$ 随 Y 的增大而递减。

综上所述，最优方案只可能是下列两种之一：

- 取 $X = R$，此时技能施放次数 $Y = \lfloor \frac{K}{R \cdot N} \rfloor + 1$，这是技能施放次数为 Y 的所有方案中的最优解。
- 取 X 使得 $F(Y + 1)$ 达到最大值，此时 $X = \frac{K}{N \cdot Y}$，若 $X \geq L$ 则更新答案，这是技能施放次数至少为 $Y + 1$ 的所有方案中的最优解。

使用分数类全整数计算的时候可发现所有中间量都可以控制在 `long long` 范围内。

7 满员电车

Shortest judge solution: 1938 Bytes.

对于询问 (S,T)，如果存在可行通勤路线，那么通勤时间至少是 $d_T - d_S$，当存在某趟正向电车允许在 S 站台上车时取到下界。这种情况易于判断，接下来只需考虑换乘方案，显然最优方案下换乘不会换多次，不然需要支付额外的等车时间。

假设在 S 站台先搭乘第 i 趟逆向电车至 P 站台 $(P < S)$，然后在 P 站台搭乘第 j 趟正向电车至 T：

- 逆向电车上车时刻为 $b_i + d_n - d_S$。
- 到达中转站台 P 的时刻为 $b_i + d_n - d_P$。
- 正向电车上车时刻为 $a_j + d_P$。
- 到达 T 站台的时刻为 $a_j + d_T$。
- 全程的通勤时间为 $(a_j + d_T) - (b_i + d_n - d_S) = (a_j - b_i) + (d_T + d_S - d_n)$，只需最小化 $a_j - b_i$。

容易发现 $a_j - b_i$ 与 T 无关，需预处理出 f_S 表示最小的 $a_j - b_i$，需要满足以下约束：

1. $P < S$。
2. 在站台 S 可以上逆向电车 i，即 $S \in [l_i, r_i]$。
3. 到达中转站台 P 的时刻不晚于正向电车上车时刻，即 $b_i + d_n - d_P \le a_j + d_P$，等价于 $b_i \le a_j - d_n + 2 \cdot d_P$。
4. 在站台 P 可以上正向电车 j。

固定 i 和 j 之后，由于当前考虑的是必须换乘的方案，因此正向电车 i 的上车范围必定无法覆盖 S。因为 P 越靠后越容易满足约束（3），因此最优方案中 P 一定取正向电车 j 允许上车的站台区间的右端点。给每趟正向电车定义 p_j 为最优的 P，以及定义 $c_j = a_j - d_n + 2 \cdot d_{p_j}$。此时，问题转化为预处理出 f_S 表示最小的 $a_j - b_i$，需要满足以下约束：

1. $p_j < S$。
2. $S \in [l_i, r_i]$。
3. $b_i \le c_j$。

考虑维护一个括号序列。将正向电车 j 视作位置 $2 \cdot c_j + 1$ 处的右括号，其权值为 a_j；将逆向电车 i 视作位置 $2 \cdot b_i$ 处的左括号，其权值为 $-b_i$。

按照站台 $1, 2, \ldots, n$ 的顺序扫描每个站台，在 p_j 处加入正向电车 j，在 l_i 处加入逆向电车 i，并在 $r_i + 1$ 处删除逆向电车 i，用线段树维护权值和最大的括号对，即可预处理出每项 f_S。

假设 n, m, p, q 同阶，时间复杂度为 $\mathcal{O}(n \log n)$。

8 飞行训练

Shortest judge solution: 2267 Bytes.

按照 $1, 2, \ldots, n$ 的顺序枚举机场 x，维护 w_z 表示机场 z 有多少种方案可以一步飞到 x，从 $x-1$ 遍历到 x 时，对 w 的影响是总计 $\mathcal{O}(m)$ 次 w 的单点加减一。

为了计算起点为 x 的飞三步回到 x 的方案数，枚举在 x 起飞的一架飞机，不妨设它的降落范围是 $[l, r]$，维护 v_i 表示输入的第 i 架飞机降落后换另一架飞机一步飞到 x 的方案数，那么对答案的贡献其实就是所有起飞点位于 $[l, r]$ 的飞机的 v 值之和。

w 的单点 u 加 1 等价于把所有降落范围覆盖 u 的飞机的 v 值都加上 1，这里介绍两种可以通过的算法。

8.1 算法一

如果直接将一架飞机看作一个三维点 (u, l, r) 的话，就需要一个数据结构维护三维点，支持三维正交修改与求和，使用带懒惰标记的 K-D Tree 维护是 $\mathcal{O}(m^{\frac{5}{3}})$ 的，不能接受。

考虑将一架飞机 (u, l, r) 拆成两个二维点：

- (x, l)，权重为 1。
- $(x, r+1)$，权重为 -1。

对于每个拆出来的二维点维护 v 值，那么 w 的单点 u 加 1 等价于把所有第二维不超过 u 的点的 v 值加上自己的权重。时间复杂度降低至 $\mathcal{O}(m\sqrt{m})$，可以通过，但常数较大。

8.2 算法二

将所有飞机按照起点从小到大排序，将排序结果分为 $\sqrt{m}$ 块，每块 $\sqrt{m}$ 架飞机。

对于每一块，维护以下信息：

- 块内所有飞机的 v 值之和 sum。
- 按块内 $\mathcal{O}(\sqrt{m})$ 架飞机的降落范围，将 $[1, n]$ 离散化至 $\mathcal{O}(\sqrt{m})$ 个区间，需预处理 $[1, n]$ 每个位置在离散化后的哪个区间。
- 离散化后每个位置被块内几架飞机的降落范围所覆盖，记为 $cover_i$。

当 w 的单点 u 需要增加 1 时，需把所有降落范围覆盖 u 的飞机的 v 值都加上 1。枚举每一块，令 u 在块内的离散化结果为 u'，则 sum 需加上 $cover_{u'}$，并塞入标记表示 “块内所有降落范围离散化后覆盖 u' 的飞机的 v 值都要加上 1”，可以通过对于每块开桶实现。

当要查询关于起飞点的区间 v 值和时：

- 对于中间完整的块，对答案的贡献就是 sum。
- 对于两边零碎的最多两块，临时求出标记桶的前缀和，暴力遍历块内每架飞机，即可通过前缀和 $\mathcal{O}(1)$ 查询它的 v 值。

时间复杂度同样为 $\mathcal{O}(m\sqrt{m})$，但空间复杂度较高，也为 $\mathcal{O}(m\sqrt{m})$。

9 崭新的假日

Shortest judge solution: 781 Bytes.

遍历 2025 年 1 月 1 日至 2075 年 12 月 31 日的每一天，分别统计 366 个候选日期每个在范围内有多少天是工作日，贪心选择最小的 k 个。计算指定日期是星期几可以直接模拟，也可以用蔡勒公式。

10 情报污染

Shortest judge solution: 4821 Bytes.

A 类情报和 B 类情报倾向于让每艘潜艇的射程尽量长，而 C 类情报和 D 类情报倾向于让每艘潜艇的射程尽量短。

先分析 C 类情报和 D 类情报，每条 D 类情报 $r_u \le w$ 是一条必须满足的一元约束；每条 C 类情报 $\min(r_u, r_v) \le w$ 等价于 $r_u \le w$ 与 $r_v \le w$ 至少要满足一条，即从两条一元约束挑选一条去必须满足，另一条可满足也可不满足。$\mathcal{O}(2^c)$ 枚举每条 C 类情报应该如何转化为必须满足的一元约束后，可以得到满足这些必须满足的一元约束的前提下每艘潜艇的射程上限。不难发现只要令每艘潜艇的射程都取到自己的上限，就可以最大限度地满足 A 类情报与 B 类情报。于是判断是否存在一组合法的射程是二元约束的选择问题。

考虑 2-SAT 建图，令 x 表示布尔变量 $\mathcal{X}$ 为真的节点，x' 表示布尔变量 $\mathcal{X}$ 为假的节点：

1. 对于每条 D 类情报 (u, w, p)，添加布尔变量 $\mathcal{X}$ 表示 “r_u 是否必须不超过 w”。连边 $x' \to x$，边权为 p，表示可以支付 p 的代价强行让它为真。

2. 对于每条 C 类情报 $\min(r_u, r_v) \le w$，添加布尔变量 $\mathcal{X}$ 表示 “r_u 是否必须不超过 w” 以及 $\mathcal{Y}$ 表示 “r_v 是否必须不超过 w”。连免费边 $x' \to y$ 与 $y' \to x$，表示两条一元约束至少需要满足一条。

3. 枚举每艘潜艇，将它对应的所有变量从小到大排序。设相邻两个变量 $\mathcal{X}$、$\mathcal{Y}$ 分别表示 “r_u 是否必须不超过 p”、“r_u 是否必须不超过 q”，其中 $p < q$。如果 $\mathcal{X}$ 为真，则 $\mathcal{Y}$ 一定为真；如果 $\mathcal{Y}$ 为假，则 $\mathcal{X}$ 一定为假。连免费边 $x \to y$ 与 $y' \to x'$。

4. 对于每条 B 类情报 $\max(r_u, r_v) \ge w$，分别找出 u 和 v 中小于 w 的最大的变量，记作 $\mathcal{X}$、$\mathcal{Y}$，那么这两个变量不能同时为真，连免费边 $x \to y'$ 与 $y \to x'$。这里不需要添加更多的约束是因为会被（3）传递到。

5. 对于每条 A 类情报 $r_u + r_v \ge w$，枚举 u 中的每个变量 $\mathcal{X}$，假设它表示 “r_u 是否必须不超过 p”，找出 v 中小于 $w - p$ 的最大的变量 $\mathcal{Y}$，那么这两个变量不能同时为真，连免费边 $x \to y'$ 与 $y \to x'$。同理也不需要添加更多的约束。

去重后最多有 $m = 2c + d$ 个变量，BCD 三类情报将连出 $\mathcal{O}(b + c + d)$ 条边，但是 A 类情报最多会连出 $\mathcal{O}(a(c + d))$ 条边，不可接受。

令 $size_i$ 表示第 i 艘潜艇有多少变量，那么 $\sum size_i = m$，不妨设 A 类情报中必有 $size_u \le size_v$，否则可以交换 u, v，分情况处理：

- 若 $size_u \le size_v \le k$，则可以在 $\mathcal{O}(k)$ 时间内双指针 u 和 v 的变量集合，连出 $\mathcal{O}(size_u)$ 条边。
- 若 $size_v > k$，那么 $\mathcal{O}(n)$ 预处理出 $f_{v,i}$ 表示 v 中最大的不超过 i 的变量，然后在 $\mathcal{O}(size_u)$ 时间内遍历 u 的每个变量，根据 f 数组 $\mathcal{O}(1)$ 定位 v 中的位置，连出 $\mathcal{O}(size_u)$ 条边。

对于一个 v 来说：

- 如果 $size_v \le \sqrt{m}$，那么每条 A 类情报最多连出 $\mathcal{O}(\sqrt{m})$ 条边。
- 如果 $size_v > \sqrt{m}$，这样的 v 最多 $\mathcal{O}(\sqrt{m})$ 个。由于对于同一个 v 来说 u 互不相同，一个 v 最多连出 $\mathcal{O}(\sum_{i=1}^{n} size_i) = \mathcal{O}(m)$ 条边。

分析可得最终一共会连出 $\mathcal{O}(b + (a+c+d)\sqrt{c+d})$ 条边，可以接受。

再来分析处理 A 类情报的时间复杂度，总计为 $\mathcal{O}(ak + \frac{nm}{k} + m\sqrt{m})$，当 k 取 $\sqrt{\frac{nm}{a}}$ 时取到最优值 $\mathcal{O}(\sqrt{nma} + m\sqrt{m})$。

删去所有收费边，求出图中所有的强连通分量，如果存在某个变量 $\mathcal{X}$ 满足 x 与 x' 位于同一个强连通分量，则无需植入情报。否则，枚举一条植入的 D 类情报，假设它代表边 $x' \to x$，那么最优解植入完毕后 x 与 x' 一定在同一个强连通分量。

对于所有 D 类情报建立一张 d 个点的有向图 $\mathcal{G}$，每个点代表一条 D 类情报。若原图中 D 类情报 i 的真节点通过免费边可以直接或间接到达 D 类情报 j 的假节点，在新图 $\mathcal{G}$ 中连边 $i \to j$，边权为情报 i 的费用，答案就是 $\mathcal{G}$ 的最小环。最小环可以通过 Floyd 算法在 $\mathcal{O}(d^3)$ 时间内求解，但根据本题 2-SAT 图的特殊性质，最小环最多只包含两条边，因此可以暴力枚举所有点对 $\mathcal{O}(d^2)$ 求解。

建立图 $\mathcal{G}$ 需要预处理出 2-SAT 图中每个点可以到达哪些 D 类情报的假节点，利用 `bitset` 加速。

总时间复杂度为 $\mathcal{O}(\sqrt{na(c+d)} + \frac{d\left(b+(a+c+d)\sqrt{c+d}\right)}{w} + d^2)$。

11 切披萨

Shortest judge solution: 2067 Bytes.

离线整体二分求出每块糖果被拿走的时刻。令 $solve(l, r, \mathcal{P}, \mathcal{Q})$ 表示需要确定 $\mathcal{P}$ 这些糖果被 $[l, r]$ 这些操作中的哪次操作拿走，$\mathcal{Q}$ 表示 $[l, r]$ 这些操作重排序后的结果：

- 若 $\mathcal{P}$ 为空，直接返回。
- 若 $l = r$，暴力检查 $\mathcal{P}$ 中每个点是否位于第 l 个操作对应的直线下方并返回。
- 取 $mid = \lfloor \frac{l+r}{2} \rfloor$。
- 依次遍历 $\mathcal{P}$ 中每个点 P，若位于 $[l, mid]$ 某个操作对应的直线下方，划分进入子问题 $solve(l, mid, \cdot, \cdot)$，反之划分进入子问题 $solve(mid+1, r, \cdot, \cdot)$。
- 依次遍历 $\mathcal{Q}$ 中每个操作 Q，若位于 $[l, mid]$，划分进入子问题 $solve(l, mid, \cdot, \cdot)$，反之划分进入子问题 $solve(mid+1, r, \cdot, \cdot)$。

- 递归处理子问题 $solve(l, mid, \cdot, \cdot)$ 与 $solve(mid+1, r, \cdot, \cdot)$。

为了检查一个点 P 是否位于 $[l, mid]$ 某个操作对应的直线下方，需要构建出这些直线的凸壳。为了省去按斜率排序以及按询问点横坐标排序的代价，需要在第一层递归之前将 $\mathcal{P}$ 和 $\mathcal{Q}$ 分别排好序，并在递归过程中划分至子问题时保持其有序。

因为每个点和每个操作只会被递归 $\mathcal{O}(\log m)$ 层，时间复杂度 $\mathcal{O}(n \log n + (n+m)\log m)$。

实现中需要注意的是，坐标范围比较大，为了在 `long long` 范围内进行全整数计算，两条直线求交点时可以进行取整，这是因为询问点的坐标一定是整数。

在线同复杂度的做法也存在，见仅支持删点的动态凸包。

12 字典树逆向

Shortest judge solution: 1203 Bytes.

显然字符串数量 $n = \mathcal{T}$ 中叶子节点的数量，由于字典树每个节点到它所有儿子的边上的字符互不相同，将所有 n 个字符串按字典序排序后，一个点子树内的所有字符串在排序结果中一定是连续的一段区间，因此可以自底向上分配每个点连向它的所有儿子的边上的字符。

假设现在正在处理节点 x，若 x 是叶子节点，那么不需要考虑边上字符分配；否则，令 $y_1, y_2, \ldots, y_k$ 为 x 的所有 k 个儿子，由于是自底向上处理，这里认为所有 k 个儿子对应的子树都已经完成了字符的最优分配，令 $seq(y_i)$ 表示 y_i 子树内所有字符串按字典序从小到大排成的最优字符串序列（不含根到 y_i 这部分公共前缀），现在需要将它们重新排列来得到 $seq(x)$。

定义 $seq(v) = c \times seq(u)$ 表示往 $seq(u)$ 中每个字符串开头都插入字符 c 得到的字符串序列 $seq(v)$；定义 $seq(w) = seq(u) + seq(v)$ 表示将序列 $seq(u)$ 后面拼接上 $seq(v)$ 得到的大小为 $|seq(u)|+|seq(v)|$ 的字符串序列 $seq(w)$；则 $seq(x) = 1 \times seq(y_1) + 2 \times seq(y_2) + \cdots + k \times seq(y_k)$。

考虑其中两个子节点 y_i, y_j $(i < j)$，交换它们更优等价于 $1 \times seq(y_i) + 2 \times seq(y_j) > 1 \times seq(y_j) + 2 \times seq(y_i)$，这等价于以下逻辑：

- 若 $seq(y_i)$ 是 $seq(y_j)$ 的前缀，那么 y_j 排在 y_i 之前更优，反之亦然。
- 若它们不互为前缀，那么字典序较小的序列排在前面更优。

于是可以通过基于比较的排序在 $\mathcal{O}(k \log k)$ 次比较内重排 x 的所有 k 个儿子，得到最优的字符分配方案。

注意到 $seq(\cdot)$ 对应字典树上的一棵子树，于是可以实现递归比较函数 $compare(u, v)$ 表示计算 $seq(u)$ 与 $seq(v)$ 的关系（相等、大于、小于、包含、被包含）：

- 若 u 或 v 至少一方是叶子，那么处理显然。
- 否则令 deg_u 和 deg_v 分别表示 u 和 v 的儿子数，显然也可以在 $\mathcal{O}(\min(deg_u, deg_v))$ 次递归内得出比较结果。

至此，我们实现了比较函数 $compare(u, v)$，令 $size_x$ 表示 x 子树的大小，比较函数的时间复杂度不会高于 $\mathcal{O}(\min(size_u, size_v))$，结合 $\mathcal{O}(k \log k)$ 的排序，固定父亲 x 的每个儿子作为 u 只有 $\mathcal{O}(\log k)$ 次。根据轻重链剖分的性质，如果 u 是轻儿子，那么它对总时间复杂度的贡献为

$\mathcal{O}(size_u \log m)$；如果 u 是重儿子，它的贡献等于所有轻儿子的贡献之和。所以这个算法的总时间复杂度为 $\mathcal{O}(\sum_{i|i\text{是轻儿子}} size_i \log m) = \mathcal{O}(m \log^2 m)$，精细地实现排序可以做到 $\mathcal{O}(m \log m)$。

具体来说，令 $T(m)$ 表示处理 m 个点的树的所需时间，不妨认为 $T(1) = 0$。当 $m \geq 2$ 时，假设根节点有 k 个儿子，它们的子树大小分别为 $s_1, s_2, \ldots, s_k$，考虑将所有 k 个儿子按照子树大小从大到小排序，依次插入平衡树中实现插入排序，那么插入第 i 个儿子时，它的比较次数为 $\mathcal{O}(\log i)$，比较所需的代价总计为 $\mathcal{O}(s_i \log i)$，于是：

$$T(m) = \sum_{i=1}^{k} (T(s_i) + C \cdot s_i \log i)$$

其中 $s_1 \geq s_2 \geq s_3 \geq \cdots \geq s_k$ 且 $\sum_{i=1}^{k} s_i = m - 1$。

假设 $T(s) \leq C \cdot s \log s$ 对 $T(1), T(2), \ldots, T(m-1)$ 成立，注意到 $s_i \leq \frac{m}{i}$，有：

$$\begin{aligned}
T(m) &= \sum_{i=1}^{k} (T(s_i) + C \cdot s_i \log i) \\
&\leq \sum_{i=1}^{k} (C \cdot s_i \log s_i + C \cdot s_i \log i) \\
&= C \sum_{i=1}^{k} s_i \log (s_i \cdot i) \\
&\leq C \sum_{i=1}^{k} s_i \log m \\
&\leq C \cdot m \log m
\end{aligned}$$

因此精细地实现排序可以做到 $\mathcal{O}(m \log m)$。