2025 ICPC 国际大学生程序设计竞赛 东亚区决赛（EC-Final）

SUA 程序设计竞赛命题组

2026 年 2 月 2 日

概况

- 预期难度：
  - BHIJK, ADE, GL, CF
- 实际通过人数排序：
  - BJKHIADELGFC

A. Outstanding Outlines

题意

- 给定 $\triangle ABC$，直线 l_1 经过 A 和 B，直线 l_2 与 l_1 平行且经过 C。“好密铺”是用全等于 $\triangle ABC$ 的三角形密铺 l_1 与 l_2 之间的带状区域，其中一个三角形恰好是 $\triangle ABC$，并且每个三角形都同时和两条直线接触。判断是否存在“好密铺”使得给定点 D 位于某个三角形的边界上。
- 数据组数 $\leq 10^4$。

A. Outstanding Outlines

- “好密铺”的每个三角形都由两条连接 l_1 与 l_2 的**凸性相反**的曲线围成。考虑带状区域严格内部的某个点 P，其同时位于两个三角形的边界上，那么有两条凸性相反的曲线经过 P。如果这两条曲线不重合，就会围出一个不同时和两条直线接触的有限区域，产生矛盾。
- 简单来说，每个三角形都要有两个顶点在同一条直线上，剩下一个顶点在另一条直线上。

A. Outstanding Outlines

- 从 $\triangle ABC$ 开始往两边扩展密铺：
  - 首先可以用 $\triangle ABC$ 旋转 180° 的三角形往两边铺一下（对应前两组样例）。
  - 如果 $\angle A = 90^\circ$，还可以用 $\triangle ABC$ 关于 AC 对称的三角形铺一下，$\angle B = 90^\circ$ 同理（对应后两组样例）。
- 不妨设 $\angle A \le \angle B$，那么：
  - 如果 $\angle B \neq 90^\circ$，可能的边界是两条直线本身、线段 AC、BC 沿着 AB 平移若干个周期得到的线段。
  - 如果 $\angle B = 90^\circ$，记 $Q = A + C - B$，可能的边界还有线段 BQ 沿着 AB 平移若干个周期得到的线段，但不包含 BQ 本身。
- 利用点积判断直角、叉积判断位置即可。一种分数的处理方式是所有横坐标乘以 x_{d_2}，所有纵坐标乘以 y_{d_2}，但是横纵坐标的缩放比不同时变换不保角，需要在变换之前判断直角。

B. Batching Badges

题意

- 给定一个由 ucp? 组成的长度为 $4n$ 的串 S，求有多少种把 ? 替换成 ucp 中之一的方案，使得 S 可以被划分成 n 个不相交的长度为 4 的子序列，且每个子序列都等于 ucup。
- $n \leq 50$，答案对 $998\,244\,353$ 取模。

B. Batching Badges

- 考虑如何判断一个由 ucp 组成的字符串是否满足题设条件。
- 从后向前扫描 S 中的每一个字符，然后贪心维护子序列的划分：
  - 若当前字符是 p，新建一个子序列 p。
  - 若当前字符是 c，找到一个等于 up 的子序列，将其修改为 cup。若不存在 up 子序列，则 S 不符合条件。
  - 若当前字符是 u，找到一个等于 p 的子序列，将其修改为 up。若不存在 p 子序列，找到一个等于 cup 的子序列，将其修改为 ucup。若仍不存在则 S 不符合条件。
- 若最终得到 n 个 ucup 子序列，则 S 符合条件。

B. Batching Badges

- 对于需要计数的原问题，在状态中记录贪心过程中每种子序列的数量即可。
- 即令 $f(i, a, b, c)$ 表示：倒着考虑到第 i 个字符，当前贪心过程中分别有 a, b, c 个 p，up，cup 子序列的方案数。
- 转移按照贪心过程讨论即可，时间复杂度 $O(n^4)$。

C. Potential Peak

题意

- 给定长度为 n 的序列 $a_1, a_2, \ldots, a_n$。
- 对每个 $1 \le k \le n$ 分别求：

$$\sum_{i=1}^{k} \left(\max_{v \ge a_i} \left(a_i - v + \sum_{j=i+1}^{k} [a_i < a_j \le v] \right) \right)$$

- $1 \le n \le 10^5$。

C. Potential Peak

- 容易发现，在 k 不断增加的过程中，对于某个固定的 i，最优的 v 是单调不降的。
- 对于 $v \geq a_i$ 的限制，将 $[a_i, n]$ 的值域区间拆成 $O(\log n)$ 个线段树上节点区间，然后对每个值域区间 $[v_l, v_r]$，求出其对应的 k 的区间 $[k_l, k_r]$。
- 假设已知 $v \in [v_l, v_r]$，则可以将原式变形为：

$$\max_{v \in [l,r]} \left(\sum_{j=i+1}^{k} [v_l < a_j \leq v] - (v - v_l) \right) + \sum_{j=i+1}^{k} [a_i < a_j \leq v_l] + a_i - v_l$$

C. Potential Peak

- 根据 Hall 定理，可以将该式子的求值转化为如下问题：
  - 对于每个 $j \in [i+1, k]$ 且 $a_j \in (v_l, v_r]$，其对应一个二分图的左部点，且可以和右部点中的前 $a_j - v_l$ 个匹配。
  - 式子的值等于二分图的最大匹配中左部点中失配点的数量。
- 故我们可以考虑维护前缀 k 中字典序最大的二分图匹配。即为：动态维护字典序最大的最大匹配中的左部点集合 S。每次加入一个新的 k 满足 $a_k \in (v_l, v_r]$ 时，先检查能否直接将 k 加入 S，若不能，找到一个最小的 $k' \in S$ 满足将 k' 替换成 k 之后存在一组匹配。
- 对于一个固定的值域区间 $(v_l, v_r]$，可以使用线段树维护这个集合 S。每次一个 k' 被移出集合 S 时，会使得所有 $i \le k'$ 的位置 i 值 $+1$。对于每个区间，预处理出这些贡献的主席树。

C. Potential Peak

- 接下来考虑求出，对于每个点 i 和一个值域区间，其对应的 k 的区间。
- 由于决策有单调性，我们只需要能对于两个不交的值域区间 $[v_l, v_r]$ 和 $[u_l, u_r]$，求出两边决策 k 的分界点即可。在预处理之后，我们已经得到了权值计算中用到的所有部分的关于下标 k 的主席树，这里只需要同时在所需的 $O(1)$ 个主席树上同时二分即可。
- 预处理和计算决策点的部分需要对 $O(n \log n)$ 个值域区间分别进行主席树的预处理和二分，时间复杂度为 $O(n \log^2 n)$。

C. Potential Peak

- 最后考虑如何统计答案。
- 对于 k 进行扫描线，动态地维护每个 i 的决策值域区间切换的操作。具体地，对于每个值域区间，使用两个树状数组分别维护当前对应这个值域区间的所有 i，和所有 a_k 属于该值域区间的 k 对应的 k'。
- 在扫描线的过程中每个 k 会对 $O(\log n)$ 个值域区间进行修改，并会产生 $O(n \log n)$ 个 i 的切换区间操作。这部分的时间复杂度同样是 $O(n \log^2 n)$。
- 于是我们在 $O(n \log^2 n)$ 的时间复杂度内解决了整个题。

D. Strategic Stones

题意

- 给一个长度为 n 的，只包含 01 和问号的字符串。
- 两人博弈，先手的每回合会将一个问号变为 0，后手的每回合会将一个问号变为 1，当没有问号时游戏结束。
- 定义游戏结束时的分数为：最长的只包含 0 的区间长度。先手想要最大化分数，后手想要最小化。*em* 若双方均采取最优策略，求最终的分数。
- $n \leq 10^5$，所有数据中 n 的和 $\leq 10^6$。

- 显然，1 的两侧是独立的，不妨对每一个不包含 1 的极长连续段分开处理，对这些取最大值就是最终的答案。

D. Strategic Stones

方法一：

- 一个局面可以用一个非负整数序列 $a_1, a_2, \ldots, a_k$ 表示，这个序列的意思是：我们有 k 个全为 0 的段，用 $k-1$ 个问号隔开，其中第 i 段有 a_i 个 0，形如 000?00?0?....?00?00（如果遇到两个相邻的问号，可以认为它们之间有一个长度为 0 的段，对于头尾的问号也同理）。
- 先手的一次操作可以用合并来刻画：选择一个 $p \in [1, k-1]$，将 a_p, a_{p+1} 合并成 $a_p + a_{p+1} + 1$。
- 后手可以选择“隔离”掉一段前缀或一段后缀，具体的，他可以选择先手操作前的序列中 a_p 前面的问号，也可以选择 a_{p+1} 后面的问号。我们可以看成：删去 $a_1, \ldots, a_{p-1}$，或删去 $a_{p+2}, \ldots, a_k$。

D. Strategic Stones

- 枚举先后手的第一轮操作，不失一般性的，假设最终保留了一个前缀，局面为 $a_1, a_2, \ldots, a_{k-2}, a_{k-1} + a_k + 1$，先手可以选择两种操作。
- 第一种操作：合并最后两项，后手此时将剩下的前缀删去，游戏结束。
- 第二种操作：合并 a_{k-2} 和 a_{k-3}，后手有两种应对方式，其一是将前缀丢掉，此时游戏的分数为 $a_{k-3} + a_{k-2} + a_{k-1} + a_k + 3$，其二是将后缀丢掉，游戏的局面变为 $a_1, \ldots, a_{k-4}, a_{k-3} + a_{k-2} + 1$，是一个子问题，可以通过线性的 dp 求解。
- 先手操作其他地方肯定是不优的，因为后手可以在先手试图和最后一项合并时及时堵上，那么第一步操作就浪费了。
- 时间复杂度 $O(n)$。

D. Strategic Stones

方法二：

- 二分答案 k，双指针预处理出互不包含的极小问号区间，使得只要能把某一个区间全填成 0，就能得到 $\geq k$ 的分数：
  - 如果存在长为 1 的区间，那么先手直接把这个区间填 0 就行；
  - 如果所有区间长度都 ≥ 3，从左往右依次配对相邻问号，后手总可以做到每一对问号里最多有一个 0，这样所有区间都不能被全填成 0；
  - 如果存在区间长度为 2，先手可以先选其中一个位置，后手必须选另外一个位置：
    - 如果这个区间与另一个长为 2 的区间有交（只能交在一个位置），那么先手在相交位置填 0，左右必定能再填一个 0；
    - 如果这个区间与另一个长为 3 的区间有交（只能交在一个位置），先手在相交位置填 0 就可以将这个长为 3 的区间缩成一个长为 2 的区间，然后可以对这个区间继续操作。
  - 对于剩下的情况，先手无法把任何一个区间全填成 0。
- 时间复杂度 $O(n \log n)$。

E. Efficient Express

题意

- 青鱼有 n 个车站，第 i 个车站有等级 a_i，所有车站等级的上限为 t。也就是说，$a_i \le t$。
- 青鱼开通了 m 种不同的列车，第 i 种有参数 x_i, y_i, p_i $(1 \le x_i, y_i \le k)$，表示在 $1 \sim p_i$ 站中停靠等级 $\ge x_i$ 的车站，在 $(p_i + 1) \sim n$ 站中停靠等级 $\ge y_i$ 的车站。
- 定义两个站直接可达当且仅当存在一种列车同时停靠两个站。给定 $p_1 \sim p_m$，$x_1 \sim x_m$，求 $(y_1, y_2, \ldots, y_m)$ 的数量使得任意两个等级相同的站均可直接到达
- $n, m \le 10^3$，$k \le 500$，$1 \le a_i, x_i, y_i \le k$。

E. Efficient Express

- 首先，问题可以转化为每个等级最左与最右的站可以直接到达，分别设为 lm_i, rm_i。
- 设 $k = \min_{1 \le i \le m} \max(x_i, y_i)$，枚举 k。
- 也就是说，对于所有 $x_i < k$ 均有 $y_i \ge k$，$y_i < k$ 均有 $x_i \ge k$。
- 为了满足这一点，在计数时，我们用 $\max(x_i, y_i) \ge k$ 和 $\max(x_i, y_i) \ge k + 1$ 分别算一次容斥一下即可。

E. Efficient Express

- 考虑再去容斥，选定 i 等级不合法，显然 $i < k$（注意此处的 k 不随我们使用的是 k 还是 $k+1$ 而改变）。
- 当选定 i 不合法时，我们希望 lm_i 左边没有 $y_t \le i$，rm_i 右边没有 $x_t \le i$，这边的左右指的是与 p_t 比较。
- 因此是否可以选定 i 可以用 rm_i 右边是否有 $x_t \le i$ 判断。
- 考虑 $dp_{i,j}$，表示从 $k-1$ 往 1 考虑到 i 等级，目前限制最严格的 lm_i 在 j。
- 边 dp 边容斥，转移考虑是否选定每个 i，然后决策所有分界点在 $j \sim j'$ 的列车的选择。
- 在转移的时候，可以类似前缀和将所有列车的方案数乘上去，即可做到总复杂度 $O(nk^2)$。

F. Set of Intervals 2

题意

- 定义一个集合 S 是青的，当且仅当：
  1. $\left(\bigcup_{i=0}^{n}[i,i] \subseteq S \subseteq \bigcup_{0\le l\le r\le n}[l,r]\right)$
  2. $\forall[l_0,r_0]\in S, \forall[l_1,r_1]\in S$ s.t. $l_0 \le l_1 < r_0 \le r_1$, $[l_0,l_1]\in S$ and $[r_0,r_1]\in S$
- 求出所有青的 S 的 $q^{|S|}$ 之和对 998244353 取模的结果。

F. Set of Intervals 2

- 先考虑得到 $\mathrm{poly}(n)$ 的做法。
- 不妨设 $A_p = \{x \mid [p, x] \in S\}$，考察 A_0。下面我们在 $l > 0$ 的区间对应的子集合法的条件下考虑 A_0 要满足什么条件。
- 如果 A_0 非空，考虑 A_0 中的任意元素 t。对于任意 $z \geq t, z \in A_0$，因为同时有 $[0, t], [0, z] \in S$，我们有 $[t, z] \in S$，于是 $A_0 \cap [t, +\infty) \subseteq A_t$。
- 再考虑 A_0 中非 0 的最小元素 y，由前文我们知道 $A_0 - \{0\} \subseteq A_y$。考虑 $y < v \in A_0$，我们取出 v 在 A_y 中的前驱 u，利用 $u, v \in A_y$ 能推出 $[u, v] \in S$。加上 $[0, v] \in S$ 可以得到 $[0, u] \in S$。于是我们知道 $A_0 - \{0\}$ 一定是 A_y 的一段前缀。我们称这个条件为**条件一**。
- 而对于 $0 < s < y$，考虑所有区间 $[s, t]$。如果 $t \geq y$，那么同时有 $[0, y], [s, t]$ 可以推出也有 $[0, s]$，与 y 最小矛盾。所以一定有 $t < y$。我们称这个条件为**条件二**。

F. Set of Intervals 2

- 容易发现这两个条件是必要的。在有这两个条件之后，考虑 $0 \le a < r \le b$ 且 $[0, r], [a, b] \in S$ 的情况。
- $a = 0$ 时，**条件一保证了** $r, b \in A_y$，因此能推出 $[r, b] \in S$。
- $r \ge y$，$a > 0$ 的情况由条件二有 $a \ge y$。由条件一有 $[y, r] \in S$，通过 $[y, r], [a, b] \in S$ 我们知道 $[y, a] \in S$，而利用条件一和 $a < r$ 我们又能推出 $[0, a] \in S$。类似地，由 $[y, r], [a, b] \in S$ 我们也知道 $[r, b] \in S$。
- 这样我们就通过这两个条件推出了 $[0, a], [r, b] \in S$，证明了它们是充要条件。
- 设 $F_{i,j}$ 为 $n = i$ 且 $|A_0| = j$ 时的答案，此处不考虑所有形如 $[i, i]$ 的区间。用前缀和优化可以得到一个 $O(n^3)$ 做法。

F. Set of Intervals 2

- 我们来观察一下这个 $O(n^3)$ 做法的转移，其转移大概形如：

$$F_{i,j} = q^j \sum_{k>0} f_k \sum_{l \geq j-1} F_{i-k,l}$$

- 此处要求 $j > 0$，$j = 0$ 时有 $F_{i,0} = f_{i+1}$。这里的 f_i 在 $i = 1$ 时为 1，在 $i > 1$ 时为 $\sum_j F_{i-2,j}$，即 $n = i - 2$ 的答案。
- 令 $G_j(x) = \sum_i F_{i,j} x^i$，$g(x) = \sum_i f_i x^i$，我们有：

$$G_j(x) = q^j \sum_{l \geq j-1} G_l(x) g(x)$$

$$G_0(x) = \frac{g(x)}{x}$$

F. Set of Intervals 2

- 可以发现每个 G 都是一个关于 g 的多项式除以 x，因此设 $G_j(x) = \frac{H_j(g(x))}{x}$，$[x^m] \sum_j H_j(x)$ 实际上是以下格路计数问题的结果：

问题

- 有一个序列 $a_0, a_1, \ldots, a_{m-1}$，$a_0 = 0$，$0 < a_i \le a_{i-1} + 1$，其权值是 $q^{\sum a_i}$，求所有 a 的权值和。

- 我们设这个值为 p_m，那么其对应的生成函数 $P(x)$ 满足 $P(x) = \frac{x}{1-P(qx)}$，推导方式就是把所有 $a_i = 1$ 的位置拿出来。

F. Set of Intervals 2

- 于是 $g(x)$ 满足以下等式：

$$\frac{g(x)-x}{x^2}=\frac{P(g(x))}{x}$$

- 也就是说 $g(x)$ 的复合逆是 $\frac{x}{1+P(x)}$。
- 于是求出 $P(x)$ 之后只需要多项式快速幂即可得到 $[x^{n+2}]g(x)$，乘上 $[i,i]$ 带来的 q^{n+1} 就是答案。
- $P(x)$ 可以用全在线卷积 $O(n\log^2 n)$ 计算，但也可以把它写作连分式，然后用倍增矩阵乘法的方式得到其分子分母。这样是 $O(n\log n)$ 的。

G. Revolver Roulette

题意

- 交互题，双人轮盘赌游戏（详细规则参见题面），你需要作为先手进行 n 局游戏，赢下其中至少 w 局游戏。
- $(n, w) = (1, 0), (2\,000, 1\,000), (20\,000, 11\,000)$。

- 由于交互器的策略是固定的，交互过程中会重启一次选手程序，以减少选手程序学习交互器的策略的机会。
- 根据中心极限定理，多次独立重复游戏中获胜次数的概率分布趋近于正态分布。如果选手程序的策略达到大约 56.89% 的胜率，要偏离 5 倍标准差才会失败，这是几乎不可能的。

G. Revolver Roulette

- 记当前游戏状态 $state = (phase, known, cage, l, b, hp_1, d_1, m_1, c_1, k_1, hp_2, d_2, m_2, c_2, k_2)$
  - $phase = 0, 1$ 分别表示正常游戏阶段、补充子弹和道具阶段；
  - $known = 0, 1, 2$ 分别表示下一发子弹类型未知、实弹、空弹；
  - $cage = 0, 1, 2$ 分别表示笼子未激活、已激活未生效、已生效；
  - l 表示剩余实弹数量，b 表示剩余空弹数量，那么如果下一发子弹类型未知，是实弹的概率是 $\frac{l}{l+b}$，是空弹的概率是 $\frac{b}{l+b}$；
  - hp_1, d_1, m_1, c_1, k_1 表示当前**先手**的生命值以及每种道具数量；
  - hp_2, d_2, m_2, c_2, k_2 表示当前**后手**的生命值以及每种道具数量。

G. Revolver Roulette

- 对游戏状态 $state$ 建转移图。满足 $hp_1 = 0$ 或 $hp_2 = 0$ 的都是终态，这些状态对应结束的游戏。对于非终态：
  - 对于 $phase = 1$，根据子弹总数以及双方道具总数可以将补充子弹和道具阶段细分成补充子弹、补充先手道具、补充后手道具三个阶段分别模拟状态转移。
  - 对于 $phase = 0$，根据题意模拟每种合法行动的状态转移，对于需要切换玩家回合的情况，交换 $(hp_1, d_1, m_1, c_1, k_1)$ 和 $(hp_2, d_2, m_2, c_2, k_2)$ 即可实现先后手交换。
- 从初始状态 $(1, 0, 0, 0, 0, 5, 0, 0, 0, 0, 5, 0, 0, 0, 0)$ 开始 BFS 建转移图，有效状态数大约是 10^6。

G. Revolver Roulette

- 记 f_s 为当前游戏状态是 s 的先手胜率，对于 $hp_1 = 0$ 的终态有 $f_s = 0$，对于 $hp_2 = 0$ 的终态有 $f_s = 1$。
- 对于非终态 s，记行动集合为 A，如果采用行动 $act \in A$ 之后，以 $p_{s \to s'}$ 概率转移到**不交换先后手**的状态 $s' \in U$，或者以 $p_{s \to s''}$ 概率转移到**交换先后手**的状态 $s'' \in V$，那么有

$$f_s = \max_{act \in A} \{ \sum_{s' \in U} p_{s \to s'} f_{s'} + \sum_{s' \in V} p_{s \to s''} (1 - f_{s''}) \}$$

- 转移图是固定的，考虑使用数值方法求解，对非终态的 f_s 任意赋初始值，顺序枚举 s 用上式迭代更新 f_s。实际在迭代若干轮之后误差就会很小，每个游戏状态对应的最优行动 act 也就几乎确定了，使用迭代出来的策略完成交互过程即可。

H. Hextech High-roll

题意

- 屏幕上有 k 个卡槽，初始数值为 $a_1, \ldots, a_k$。重抽时，会从一个大小为 n 的集合 $\{b_1, \ldots, b_n\}$ 里均匀随机不放回重抽。
- 每个卡槽至多重抽一次；对任意位置重抽后，立刻观察到结果，并可以再决定是否继续重抽。
- 最终从屏幕上 k 张牌里选一张，最大化所选数值的期望。
- $k \le n \le 10^5, |a_i|, |b_i| \le 10^5$，输出答案相对或绝对精度 10^{-4}。

- 观察：重抽所有非最大值卡牌一定不差，因此我们一定至少重抽 $k-1$ 次。此时，只有是否重抽原有最大值的决策。

H. Hextech High-roll

- 不妨假设[1] $b_1 < b_2 < \cdots < b_n$。我们令 $A = \max_i a_i$, $B = b_j$ 为 $k-1$ 次重抽后抽到的最大值，其中 $j \geq k-1$。
- 此时，重抽的收益为

$$\frac{1}{n-k+1}\left(\sum_{i=1}^{j-k+1} B + \sum_{i=j+1}^{n} b_i\right),$$

  不重抽的收益为 $\max(A, B)$，我们总可以取到其中的较大值。
- 枚举 j，我们由上面两项的较大值乘以概率

$$\Pr[B = b_j] = \frac{\binom{j-1}{k-2}}{\binom{n}{k-1}},$$

  求和即是答案。可以线性预处理需要的求和与概率。

[1]让所有的 b 互不相同总可以通过按标号作为次关键字来实现

H. Hextech High-roll

- 注意到直接计算组合数是不可行的，我们需要用数值稳定的方式求出概率。一些可能的方法有：
  - 利用类似

$$\log \binom{n}{m} = \log(n!) - \log(m!) - \log((n-m)!)$$

    求出 $\log(\Pr[B = b_j])$ 后，再计算 $\exp(\log(\Pr[B = b_j]))$.
  - 注意到

$$\Pr[B = b_n] = \frac{\binom{n-1}{k-2}}{\binom{n}{k-1}} = \frac{k-1}{n},$$

    然后可以用相邻项递推，

$$\Pr[B = b_{j-1}] = \Pr[B = b_j] \cdot \frac{j-k}{j-1}.$$

  - 利用上一条的结论，$\Pr[B = b_j]$ 随 j 递减得很快，因此只需要枚举靠近 n 的 $O(\log \varepsilon^{-1} \cdot n/k)$ 项即可。

I. Redundancy Refrain

题意

- 给长度为 n 的序列，序列包含 $0 \sim k$ 的整数。称两个长度相同的下标区间是 anagram，若区间里的元素形成的 multiset 相同，把区间的长度称为 anagram 长度。
- 进行构造：把 0 都改成 $1 \sim k$，使得最大 anagram 长度最小。
- $k \leq n \leq 2 \times 10^5$

I. Redundancy Refrain

- 注意到：最长 anagram 长度就是最远的两个相同元素的距离。可用反证法证明。
- 二分答案 m。考虑每种元素，设它已经出现的位置中，最左边是 l，最右边是 r。假设把 0 都改掉之后，它最左边是 L，最右边是 R，则有 $L \le l$，$r \le R$，且 $R - L \le m$。
- 所有元素的 $[L, R]$ 要覆盖长度为 n 的一段，显然 $R - L$ 越大，覆盖起来就越容易。所以问题就变成：有 k 个长度为 m 的区间，每个区间有个活动范围，怎么才能覆盖长度为 n 的一段。
- 经典的用堆维护贪心：如果当前有多个区间可以选，则选择活动范围右端点最小的区间。
- 复杂度 $\mathcal{O}(n \log^2 n)$。

J. Tree-mendous Transmission

题意

- 给一棵 n 个点的树，每次操作可以标记一条简单路径上的点，要求本次标记的点至多只有一个点在之前已被标记，最少操作次数标记所有点，并输出方案。
- $n \leq 3 \times 10^5$。

J. Tree-mendous Transmission

- 考虑一组合法操作里的所有简单路径，任意两条路径之间没有公共边，否则先后操作这两条路径时，公共边的两个端点都会被标记，产生矛盾。
- 另一方面，对于一组没有公共边的路径，对原树任意定根，按照路径的 LCA 从浅到深的顺序进行操作，每次只有路径的 LCA 可能在之前被标记。
- 因此可以使用树形动态规划求解最小的合法路径集合，记 $f_{u,i}$ 表示标记 u 子树里所有点且有 i 条路径从 u 向上延伸时的最少路径数量。由于任意两条路径不能有公共边，i 只能是 0 或者 1，还原出一组方案即可，时间复杂度 $O(n)$。或者也可以自底向上贪心覆盖。

K. Magically Marked Matching Master

题意

- 给一棵 n 个点的树，在线匹配：边会以均匀随机顺序逐条出现，你必须立刻决定选择是否加入匹配，且最终选出的边集必须是一个最大匹配。
- 在进行在线匹配前，可以编码一个长度 $n-1$ 的二进制串作为提示。更具体地，
  - 第一遍运行时，可以看到整棵树的边集，并在这之后要输出一个长度为 $n-1$ 的二进制串 s；
  - 第二遍运行开始时，只能看到二进制串，随后边按随机顺序依次出现，每出现一条边就必须立即输出是否选择该边。
- $n \leq 500$。

K. Magically Marked Matching Master

- 固定任意一组最大匹配 M，令 $w_e = 1$ 表示 $e \in M$。如果边的顺序两遍一致，那么发送 w 即是答案。
- 如果是任意边顺序，我们必须要用点的标号还原信息。一个可能的构造是：
  - 令 $w_e = x_u \oplus x_v$，其中 $\oplus$ 为异或操作。
  - 按照树的结构，我们可以得到互为取反的两组解 x 和 $\bar{x}$。我们选择其中任意一组，比如 $x_1 = 0$ 的一组，发送 $x_{2 \cdots n}$。
  - 此时，所有匹配中的边均满足 $x_u \neq x_v$，不在匹配中的边均满足 $x_u = x_v$。因此，匹配时只匹配 $x_u \neq x_v$ 的边即可完成任务。
  - 本质上，x 发送的是一个割集，割边恰好为最大匹配。

L. Logical Resonance

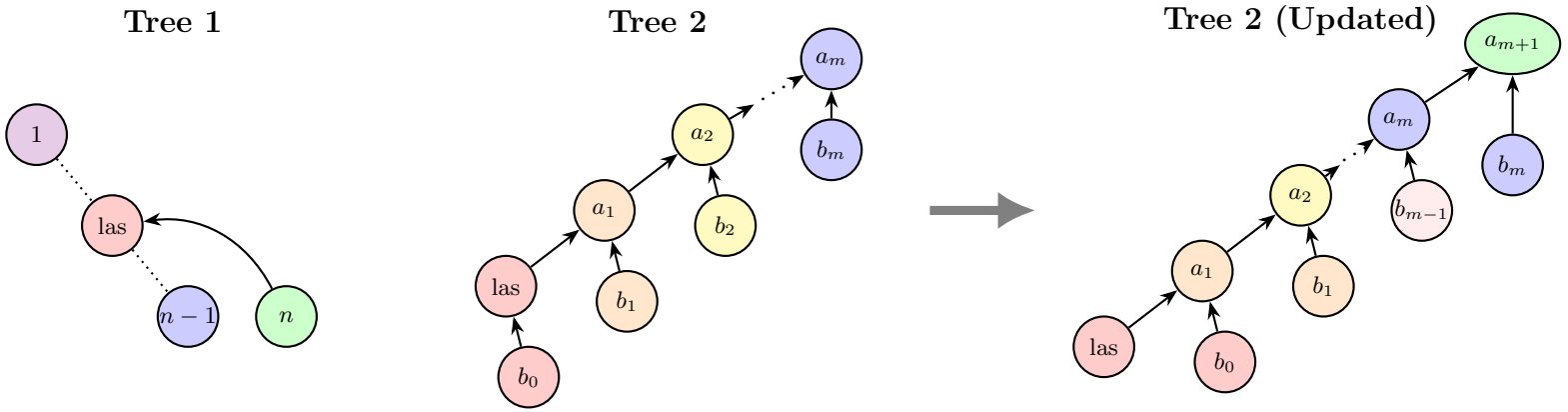
题意

- 现在有两种类型的树 $Tree\ 1$ 以及 $Tree\ 2$，且节点数均为 n。
- $Tree\ 1$ 满足根节点为 1，其余节点 i 的父亲均为编号比它小的节点 p_i。
- $Tree\ 2$ 满足根节点为 n，其余节点 i 的父亲均为编号比它大的节点 q_i。
- 现在需要构建一个 $Tree\ 1$ 到 $Tree\ 2$ 的双射，并且满足对于任意节点 i，i 在 $Tree\ 1$ 中为叶子 $\iff$ i 在 $Tree\ 2$ 中为非叶子。
- $n \le 10^3$，$\sum n^2 \le 10^7$。

L. Logical Resonance

方法一：

- *Tree* 1 映射到 *Tree* 2。
- 先递归构造 $n-1$ 的部分，然后令 $q_{n-1}=n$。
- 假设 *Tree* 1 中 $p_n = las$，拉出 *Tree* 2 中 *las* 开始的链：$\boxed{a_0 = las} \to a_1 \to a_2 \to \cdots \to \boxed{a_m = n-1} \to \boxed{a_{m+1} = n}$。
- 把非链上的所有满足 $q_i = a_t$ 的节点 i，把它们的父亲指向链上的下一个节点，即 $q_i = a_{t+1}$。



L. Logical Resonance

方法一：

- $Tree\ 2$ 映射到 $Tree\ 1$。（把前面部分逆操作一下）
- 记重儿子为标号最大的儿子，可以发现在 $Tree\ 1$ 映射到 $Tree\ 2$ 的每一个步骤结束时，$\boxed{a_0 = las} \to a_1 \to a_2 \to \cdots \to \boxed{a_m = n-1} \to \boxed{a_{m+1} = n}$ 一定是根开始的重链。
- 把 $Tree\ 2$ 从根开始的重链找出来，循环上移，然后把叶子节点（变成了 n）去掉。
- 假设重链的最小值为 las。$Tree\ 1$ 添加边 $p_n = las$。
- 用操作后的 $Tree\ 2$ 递归构造出 $Tree\ 1$ 的 $n-1$ 部分。

方法二：

- $Tree\ 1$ 映射到 $Tree\ 2$。
- 初始 $Tree\ 2$ 为空。从 n 到 1 考虑 $Tree\ 1$ 的每个非叶子 x，其儿子为 $s_1, s_2, \ldots, s_m$。
- 把当前 $Tree\ 2$ 这些点对应的当前连通块的根 $root(s_i)$ 从小到大串起来，同时 $root(s_i)$ 中最小的作为 x 在 $Tree\ 2$ 里的父亲。
- 形式化地讲，假设 $\{root(s_i)\}$ 从小到大为 $b_1, b_2, \ldots, b_m$，那么令 $q_{b_i} = b_{i+1}$，且 $q_x = b_1$。

方法二：

- $Tree\ 2$ 映射到 $Tree\ 1$。
- 初始 $Tree\ 1$ 为空。从 n 到 1 考虑 $Tree\ 2$ 的每个叶子 y，从 y 沿着 $Tree\ 2$ 往上爬直到有一个祖先 u 有比 y 更小的后代。
- 假设 y 往上爬到 u 路径上的节点为 $t_0 = y, t_1, t_2, \ldots, t_{m-1}, t_m = u$，每个点 t_i 在 $Tree\ 1$ 中的当前连通块的根为 $root(t_i)$。
- 将 $root(t_1), root(t_2), \ldots, root(t_m)$ 在 $Tree\ 1$ 的根均设为 y。

Thank you!