IOI2026 中国国家集训队集中培训 第三试 B. 复读机

IOI 2026 中国国家集训队工作组

2025 年 12 月 4 日

问题（复读机（repeater））

- 有一个信任游戏共 m 轮，你先行动，双方轮流选择合作或欺骗，合作则自己分数 -1 对方 $+3$，欺骗则双方都不变。
- 对手从策略池中随机选择一个 01 串作为策略，0 代表合作，1 代表欺骗，超出字符串长度的部分为复读你的上一次行动。
- n 次动态地修改对手的策略池，你已知对手策略池，但不知道具体的策略。目标是最大化你的期望得分。

数据范围：$n \le 3 \times 10^5$，$m \le 10^6$，$\sum |s_i| \le 4 \times 10^5$。

- 将对手的串建 Trie，那么每次根据对手的回答可以确定往左子树走还是右子树走，走到 u 时要处理的为 u 子树内的所有情况，以及根到 u 路径上会有若干个「复读」。

- 将对手的串建 Trie，那么每次根据对手的回答可以确定往左子树走还是右子树走，走到 u 时要处理的为 u 子树内的所有情况，以及根到 u 路径上会有若干个「复读」。
- 假设有 x 个处在「复读」阶段。如果你在节点 u，你若选择「合作」，那么处在「复读」阶段的串也会选择「合作」，相当于划分为两个子问题：左儿子（合作）有 x 个处在「复读」阶段；右儿子（欺骗）有 0 个处在「复读」阶段。

- 将对手的串建 Trie，那么每次根据对手的回答可以确定往左子树走还是右子树走，走到 u 时要处理的为 u 子树内的所有情况，以及根到 u 路径上会有若干个「复读」。
- 假设有 x 个处在「复读」阶段。如果你在节点 u，你若选择「合作」，那么处在「复读」阶段的串也会选择「合作」，相当于划分为两个子问题：左儿子（合作）有 x 个处在「复读」阶段；右儿子（欺骗）有 0 个处在「复读」阶段。
- 可以直接动态规划，第二维记录 x 的值。对于子任务 1 可以开数组，后面的子任务需要用 map 或者其他记录方法。

- 将对手的串建 Trie，那么每次根据对手的回答可以确定往左子树走还是右子树走，走到 u 时要处理的为 u 子树内的所有情况，以及根到 u 路径上会有若干个「复读」。
- 假设有 x 个处在「复读」阶段。如果你在节点 u，你若选择「合作」，那么处在「复读」阶段的串也会选择「合作」，相当于划分为两个子问题：左儿子（合作）有 x 个处在「复读」阶段；右儿子（欺骗）有 0 个处在「复读」阶段。
- 可以直接动态规划，第二维记录 x 的值。对于子任务 1 可以开数组，后面的子任务需要用 map 或者其他记录方法。
- 记 $S=\sum|s_i|$，则单次询问时间复杂度 $O(S\cdot\min(n,m))$（如果用 map 则额外带 $\log$），期望得分：46。

- 设 $f(u,x)$ 表示在 u 节点 x 个处在「复读」阶段的答案，a_u 表示 u 节点结尾的串个数，b_u 表示 u 子树内 a_u 的和。转移可以写为：

$$f(u,x)=\max\{f(ls_u,x+a_u)+f(rs_u,0)+V,\ f(ls_u,0)+f(rs_u,x+a_u)\},$$

其中 $V=3\times(x+a_u)-(x+b_u)$。

- 设 $f(u,x)$ 表示在 u 节点 x 个处在「复读」阶段的答案，a_u 表示 u 节点结尾的串个数，b_u 表示 u 子树内 a_u 的和。转移可以写为：

  $$f(u,x)=\max\{f(ls_u,x+a_u)+f(rs_u,0)+V,\ f(ls_u,0)+f(rs_u,x+a_u)\},$$

  其中 $V=3\times(x+a_u)-(x+b_u)$。
- 对于 Trie 外的节点，只剩下处在「复读」阶段的，全都合作最优，即：

  $$f(u,x)=2\times(m-deth_u+1)\cdot x.$$

- 设 $f(u,x)$ 表示在 u 节点 x 个处在「复读」阶段的答案，a_u 表示 u 节点结尾的串个数，b_u 表示 u 子树内 a_u 的和。转移可以写为：

  $$f(u,x)=\max\{f(ls_u,x+a_u)+f(rs_u,0)+V,\ f(ls_u,0)+f(rs_u,x+a_u)\},$$

  其中 $V=3\times(x+a_u)-(x+b_u)$。
- 对于 Trie 外的节点，只剩下处在「复读」阶段的，全都合作最优，即：

  $$f(u,x)=2\times(m-deth_u+1)\cdot x.$$

- 显然每个 $f(u)$ 可以看做一些关于 x 的一次函数的 $\max$，问题的答案就是 $f(r,0)$。

- 对每个节点维护所有的一次函数，对于斜率相同的情况只保留更优的。每次修改时也只需要修改 $O(|s_i|)$ 个节点。

- 对每个节点维护所有的一次函数，对于斜率相同的情况只保留更优的。每次修改时也只需要修改 $O(|s_i|)$ 个节点。
- 不难发现每个节点的函数数量不会超过这个子树深度，因为可能的斜率个数是深度级别的。

- 对每个节点维护所有的一次函数，对于斜率相同的情况只保留更优的。每次修改时也只需要修改 $O(|s_i|)$ 个节点。
- 不难发现每个节点的函数数量不会超过这个子树深度，因为可能的斜率个数是深度级别的。
- 总时间复杂度为 $O(S \cdot m)$，期望得分：40。

- 取所有字符串的终止节点作为关键点，考虑关键点构成的虚树，在虚树的每条边上考虑。如果沿着这个边一直走，那么处于复读状态的数量 x 始终不变，直到某个时刻走出 Trie 时 x 变为 0，此时只需要一直欺骗直到下一个关键点。事实上可能的最优策略仅可能为一直合作直到走出 Trie，或一直欺骗直到走出 Trie，不难验证其他策略均可以调整至这两种中一种且使收益更大。

- 取所有字符串的终止节点作为关键点，考虑关键点构成的虚树，在虚树的每条边上考虑。如果沿着这个边一直走，那么处于复读状态的数量 x 始终不变，直到某个时刻走出 Trie 时 x 变为 0，此时只需要一直欺骗直到下一个关键点。事实上可能的最优策略仅可能为一直合作直到走出 Trie，或一直欺骗直到走出 Trie，不难验证其他策略均可以调整至这两种中一种且使收益更大。
- 每个 $f(u)$ 表示中的一次函数个数的量级为 u 子树内关键点个数常数倍：虚树点数是关键点个数的常数倍，而只会在走出 Trie 时产生新的函数。

- 取所有字符串的终止节点作为关键点，考虑关键点构成的虚树，在虚树的每条边上考虑。如果沿着这个边一直走，那么处于复读状态的数量 x 始终不变，直到某个时刻走出 Trie 时 x 变为 0，此时只需要一直欺骗直到下一个关键点。事实上可能的最优策略仅可能为一直合作直到走出 Trie，或一直欺骗直到走出 Trie，不难验证其他策略均可以调整至这两种中一种且使收益更大。
- 每个 $f(u)$ 表示中的一次函数个数的量级为 u 子树内关键点个数常数倍：虚树点数是关键点个数的常数倍，而只会在走出 Trie 时产生新的函数。
- 根据 Trie 的性质，每个节点的子树内的关键点个数和是 $O(S)$ 的，因此对于静态的问题，我们可以直接维护每个节点的函数集合，单组时间复杂度为 $O(S)$。期望得分：61。

- 进一步观察，每个节点的一次函数个数实际上为 $O(\sqrt{S})$ 级别。简要说明：我们不妨把原本的虚树上的点，和上述分析中在每个虚树边上走出 Trie 的点合称为新关键点，那么所有一次函数一定来源于一个新关键点，且斜率只和新关键点的深度有关。

- 进一步观察，每个节点的一次函数个数实际上为 $O(\sqrt{S})$ 级别。简要说明：我们不妨把原本的虚树上的点，和上述分析中在每个虚树边上走出 Trie 的点合称为新关键点，那么所有一次函数一定来源于一个新关键点，且斜率只和新关键点的深度有关。
- 新关键的个数和虚树点数同级别，而虚树点数和关键点个数同级别。实际上可以得到，新关键点的深度和与虚树上点的深度和同级别，而虚树上点的深度和与关键点深度和同级别，即 $O(S)$。

- 进一步观察，每个节点的一次函数个数实际上为 $O(\sqrt{S})$ 级别。简要说明：我们不妨把原本的虚树上的点，和上述分析中在每个虚树边上走出 Trie 的点合称为新关键点，那么所有一次函数一定来源于一个新关键点，且斜率只和新关键点的深度有关。
- 新关键的个数和虚树点数同级别，而虚树点数和关键点个数同级别。实际上可以得到，新关键点的深度和与虚树上点的深度和同级别，而虚树上点的深度和与关键点深度和同级别，即 $O(S)$。
- 那么若有 k 个有效的一次函数，至少需要有 k 个不同斜率，则至少有 k 个不同深度的新关键点，这些新关键点的深度和为 $O(k^2)$，但所有新关键点的深度和是 $O(S)$，这说明了每个节点下有效一次函数个数是 $O(\sqrt{S})$。

- 进一步观察，每个节点的一次函数个数实际上为 $O(\sqrt{S})$ 级别。简要说明：我们不妨把原本的虚树上的点，和上述分析中在每个虚树边上走出 Trie 的点合称为新关键点，那么所有一次函数一定来源于一个新关键点，且斜率只和新关键点的深度有关。
- 新关键的个数和虚树点数同级别，而虚树点数和关键点个数同级别。实际上可以得到，新关键点的深度和与虚树上点的深度和同级别，而虚树上点的深度和与关键点深度和同级别，即 $O(S)$。
- 那么若有 k 个有效的一次函数，至少需要有 k 个不同斜率，则至少有 k 个不同深度的新关键点，这些新关键点的深度和为 $O(k^2)$，但所有新关键点的深度和是 $O(S)$，这说明了每个节点下有效一次函数个数是 $O(\sqrt{S})$。
- 所以我们删除无效的函数后直接维护每个节点上的所有函数，即可做到 $O(S\sqrt{S})$ 的复杂度。，期望得分：100。