IOI2026 中国国家队选拔 第一试《谜题 II》解题报告

IOI2026 中国国家集训队工作组

2026 年 2 月 9 日

题意转化

- 对于原题意下的限制 $A_{x_1,y_1} < A_{x_2,y_2}$，连接有向边 $(x_1, y_1) \to (x_2, y_2)$。
- 那么题意转化为，对一个特殊的网格图做拓扑序计数：这个网格图 2 行 2^k 列，所有相邻的点之间都连了有向边。
- 以下设 $n = 2^{k+1}$ 为网格图点数。

部分分

- 枚举所有 $n!$ 种排列然后判断是否合法，时间复杂度 $O(tn!\,\mathrm{poly}(n))$。
- 进一步地，通过状压 DP 计算任意 n 个点有向图的拓扑序数量，时间复杂度 $O(t2^n\,\mathrm{poly}(n))$。

部分分

- 考虑动态规划，设 $dp_{i,j,k}$ 表示如果只确定前 i 列内所有值的相对大小，第 i 列上面的值是 j，下面的值是 k 的方案数对 2 取模后的结果。
- 转移直接枚举第 $i+1$ 列两个位置被填入的值，时间复杂度 $O(tn^5)$。
- 可以分步转移优化到 $O(tn^4)$，进一步前缀和优化可以做到 $O(tn^3)$。
- 由于只求答案模 2 的结果，还可以压位做到 $O(tn^3/w)$。

容斥转化

- 根据容斥原理，对一张任意的图 G，可以任选 G 中有向边 $u \to v$，那么图 G 的拓扑序数量（下面简称答案）就等于从 G 中删去 $u \to v$ 得到图的答案，减去从 G 中反转 $u \to v$ 这条边方向得到图的答案。
- 考虑 2^k 列以及模 2 到底有什么用：总点数是 2^{k+1}，如果图不弱连通，答案一定是 0，原因是合并各个弱连通块答案会乘二项式系数，但这个系数模 2 一定是 0。
- 推论：如果视为无向图后一条边是这张无向图的割边，这条边的方向不影响答案，可以随意反向。
- 此外，由于模 2，容斥系数 -1 也可以直接忽略。

容斥转化

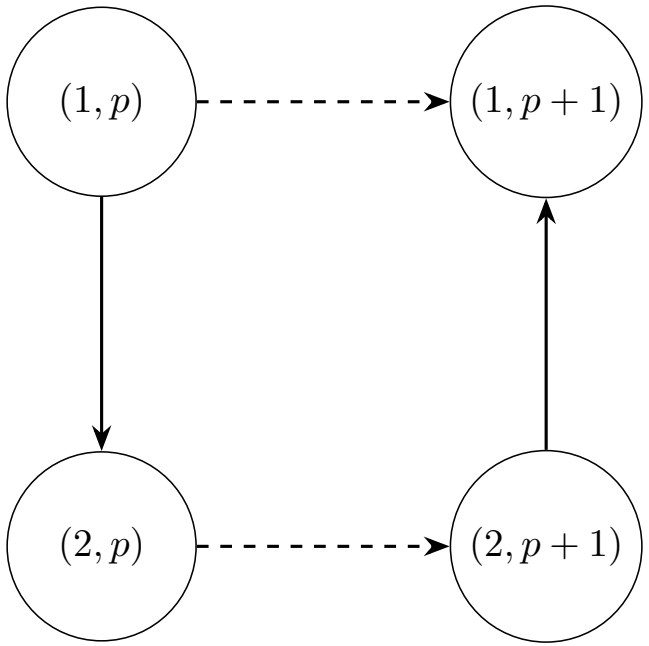
- 通过“逐步容斥”，也就是每次选定一个子问题，选定其中一条边对它进行容斥从而该子问题分裂成两个，最终可以使得所有子问题满足：
  - 网格图中所有横向的边一定指向右边，所有点至多有一条出边，且图弱连通。
- 其中图弱连通的性质不是很重要，因为如果不满足就直接丢弃这个子问题就可以了。
- 这样得到的子问题是每个点儿子数量不超过 2 的根向树，答案是 1 当且仅当对所有儿子数为 2 的点，其两个儿子的子树大小写作二进制无交。
- 接下来我们给出一个达成该目的的算法。方便起见我们先考虑 $b_i \neq b_{i+1}$ 的特殊性质，也就是所有竖着的边方向交错的子任务。

归纳考虑

- 对 p 进行归纳，作出以下归纳假设：
  - 只对前 p 列内部的边进行容斥，就已经使得网格图前 p 列内部已经满足上述性质。
  - 第 p 列的竖边没有被改变方向。
- $p = 1$ 是归纳起点，考虑 $p \to p + 1$ 该如何做。
- 最简单的想法是直接容斥让 $p, p + 1$ 列之间的两条横边要么向右要么不存在，这样所有横向边指向右边的性质就直接满足了。
- 对于所有点至多有一条出边的性质，由于 $p - 1, p$ 列之间的横边一定向右，从而 $(1, p), (2, p)$ 两个点的出边只有可能是 $p, p + 1$ 列之间的横边，或者连接 $(1, p), (2, p)$ 之间的边。这里还需要进一步讨论。

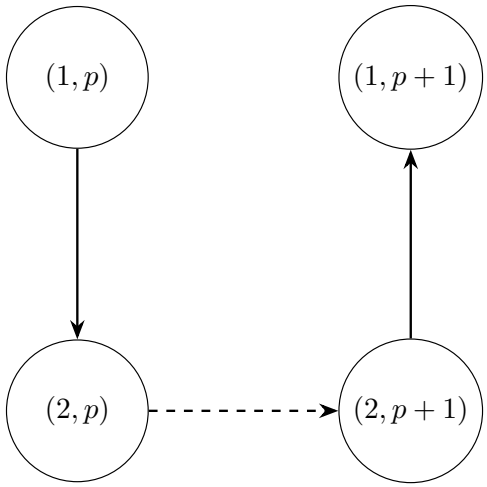
容斥细节

- 现在的情况如下图所示，其中虚线边表示这条边可能存在也可能不存在，实线边表示这条边必然存在：



分类讨论

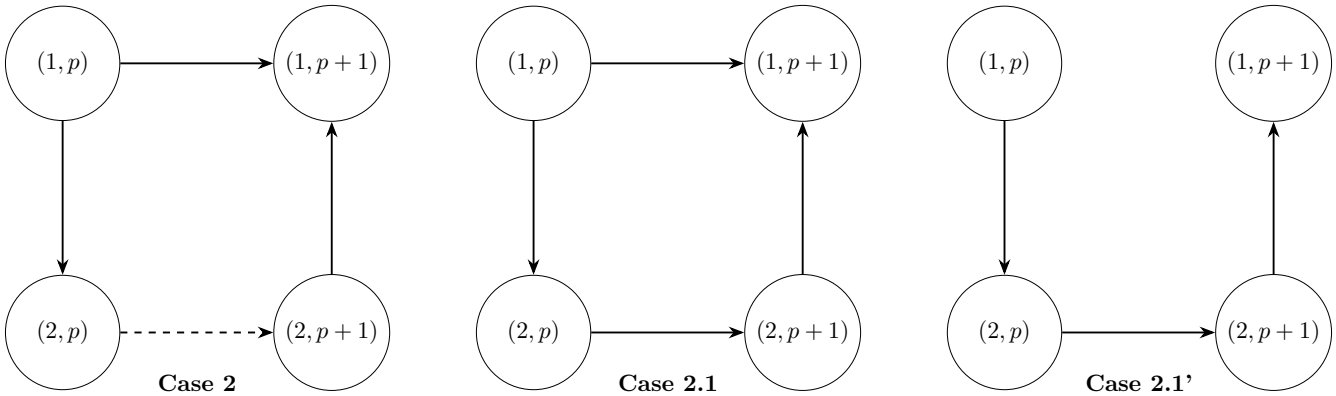
- 先对 $(1,p),(1,p+1)$ 之间的边是否存在分类讨论：
- Case 1：$(1,p)\to(1,p+1)$ 的边不存在，如下图所示：



- 此时 $(2,p)\to(2,p+1)$ 若不存在则图不连通，故只需考虑其存在的情况。

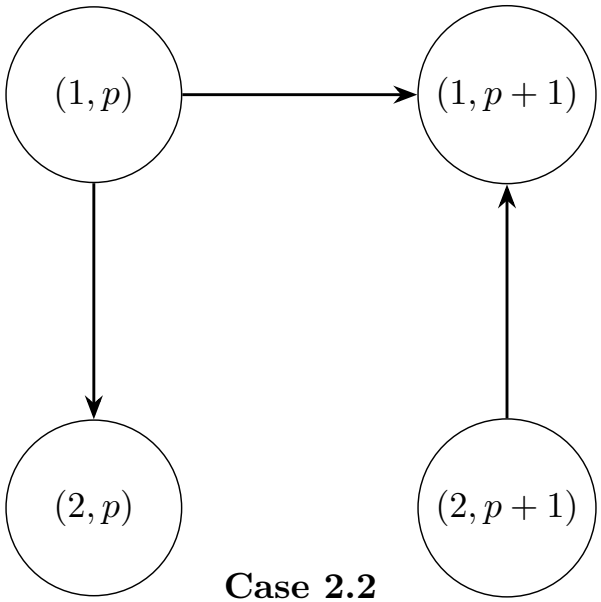
分类讨论

- Case 2：$(1,p) \to (1,p+1)$ 的边存在。对 $(2,p),(2,p+1)$ 之间的边是否存在进一步讨论：
- Case 2.1：$(2,p) \to (2,p+1)$ 的边存在。可达性相同的 DAG 的答案一样，删去边 $(1,p) \to (1,p+1)$ 不改变可达性，所以可进行下图所示的转化。可以发现，转化后的图和 Case 1 的图一致。

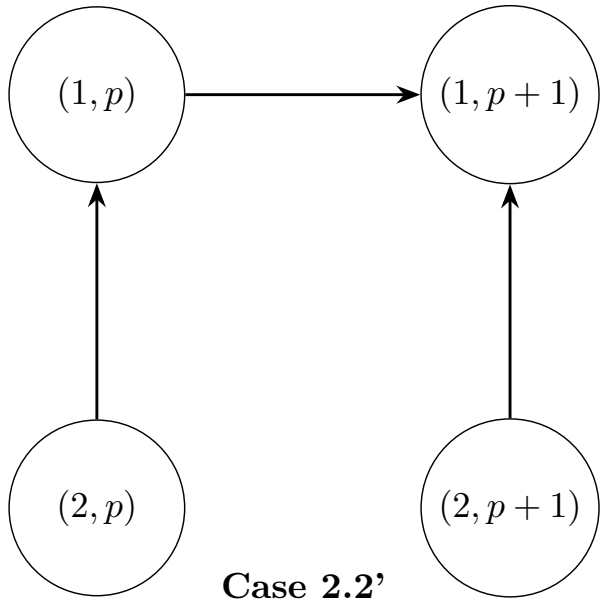


分类讨论

- Case 2.2：$(2,p) \to (2,p+1)$ 的边不存在。此时 $(1,p)$ 和 $(2,p)$ 之间这条边是割边，因此可以认为它向上。



Case 2.2



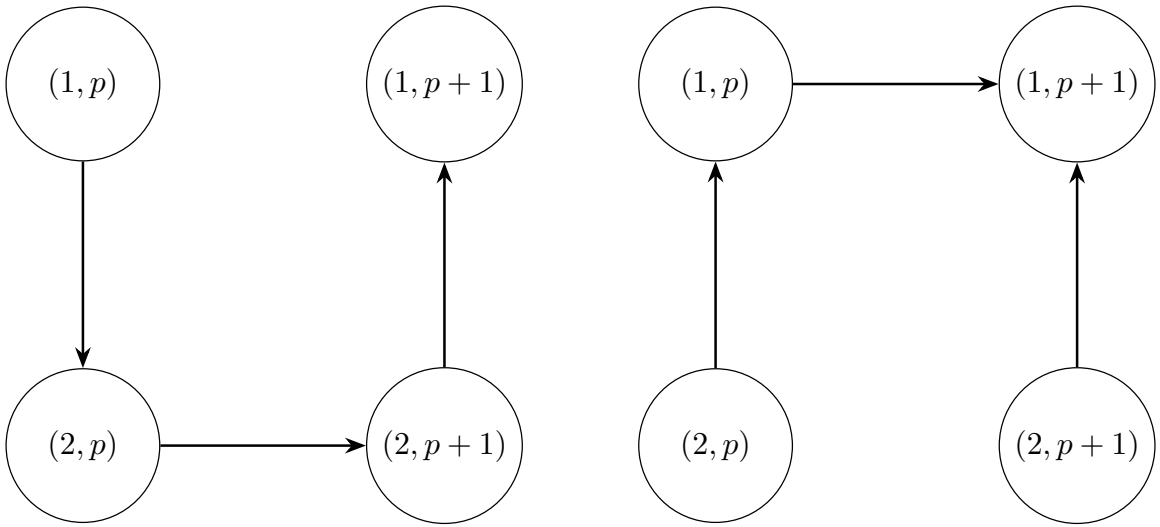
Case 2.2'

情况整理

- 上述分析表明当 $b_p = 1, b_{p+1} = 0$ 时，对于原先的子问题会分裂为：
  - Case 1 要求第一步容斥后 $(1, p) \to (1, p+1)$ 的边不存在，即原先方向是 $(1, p+1) \to (1, p)$，条件是 $a_p = 0$。还要求 $(2, p) \to (2, p+1)$ 的边存在，但不论这条边原先的方向，第一步容斥后总会得到一个向右的子问题，这里没有限制。
  - Case 2.1 要求第一步容斥后两条边均存在，同上这里没有限制。
  - Case 2.2 要求第一步容斥后 $(1, p) \to (1, p+1)$ 存在而 $(2, p) \to (2, p+1)$ 不存在，条件是 $c_p = 0$。

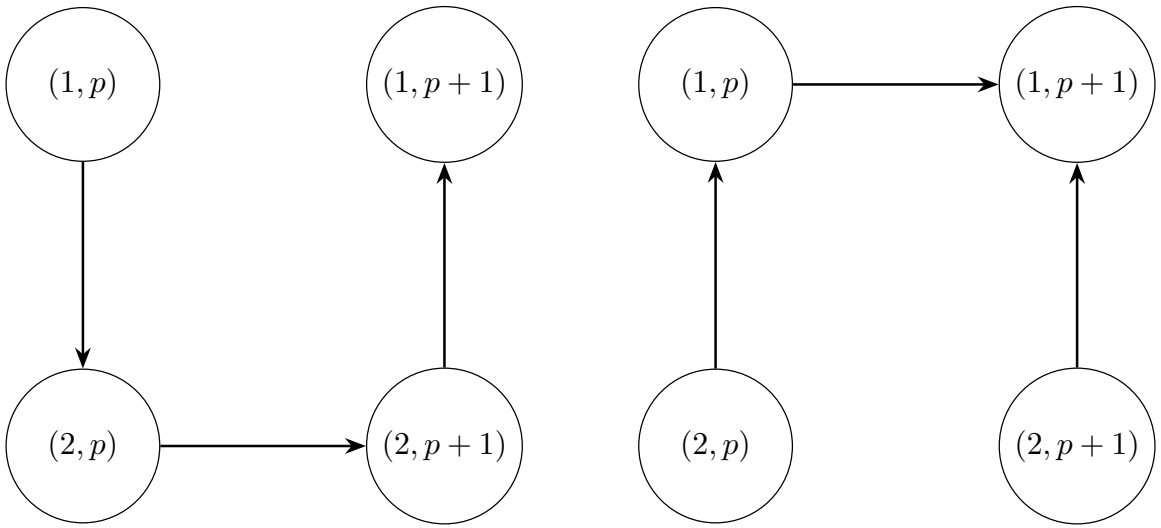
情况整理

- 简单来说：
  - Case 1：$a_p = 0$ 时得到一个如下左图的子问题。
  - Case 2.1：无论如何，得到一个如下左图的子问题。
  - Case 2.2：$c_p = 0$ 时得到一个如下右图的子问题。



情况整理

- 现在已经容斥得到期望的根向树的结构，假设对于一个已有的子问题，其二元组信息是 (x, y)，表示该子问题中 $1 \sim p-1$ 列的点一直沿着出边走，首个走到 $(1, p)$ 的点有 x 个，$(2, p)$ 的则是 y 个，在该容斥过程中：
  - $a_p = 1$ 且 $valid(x+1, y)$ 时得到一个 $(0, x+y+2)$ 的子问题。
  - $c_p = 0$ 且 $valid(x, y+1)$ 时得到一个 $(x+y+2, 0)$ 的子问题。
- 其中 $valid(s, t) = 1$ 当且仅当 s, t 二进制下无交。

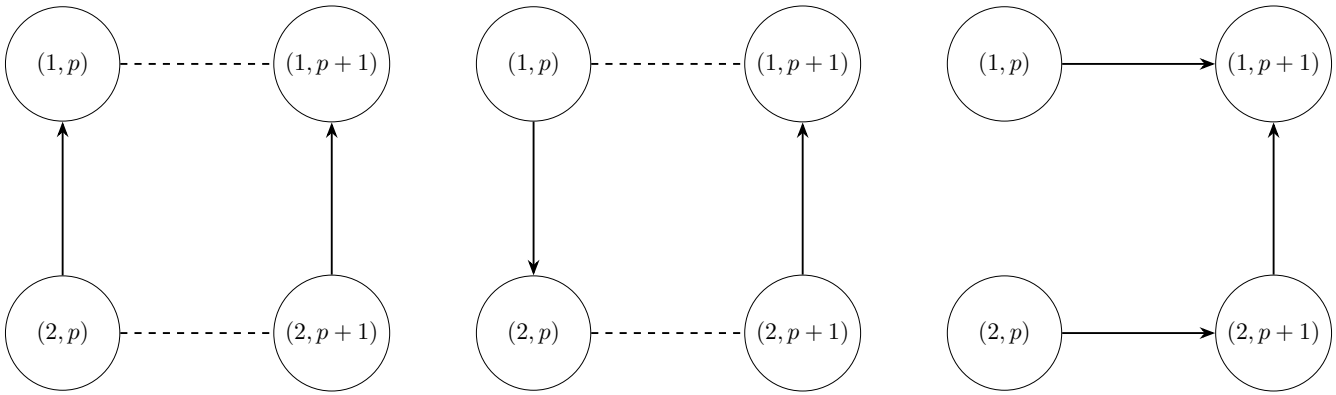


特殊性质的完整解法

- $b_p = 0,\ b_{p+1} = 1$ 的转移是对称的，注意到容斥进行到第 p 列时二元组必然形如 $(2(p-1), 0)$ 或 $(0, 2(p-1))$，记录有多少子问题对应这两种二元组 DP 转移即可，时间复杂度 $O(t2^k)$。
- 进一步化简可以得到答案为 1 的充要条件：对于所有 $1 \le i < 2^k$，均有 $a_i = c_i$。
- 另一方面，直接找规律也可能得到这个结论。

容斥

- 若 $b_p = b_{p+1}$，则可以先对 $(2,p) \to (1,p)$ 这条边容斥，得到的子问题中边方向变为 $(1,p) \to (2,p)$，是前面讨论的情形。
- 而 $(1,p),(2,p)$ 间边不存在的，由于原本前 p 列是树，现在切开 $(1,p),(2,p)$ 间边后 $(1,p),(2,p)$ 两点必然不连通，从而 $(1,p),(1,p+1)$ 间的边和 $(2,p),(2,p+1)$ 间的边都是割边，都转成向右即可。



一般情况的初步解法

- 所以 $b_p = b_{p+1}$ 的情况只需要添加 $(x, y) \to (x+1, y+1)$ 的转移即可。
- 注意到 $x + y = 2(p-2)$ 恒成立，直接对每种二元组记录对应方案数转移，时间复杂度 $O(t4^k)$。
- 进一步的优化是考虑二元组之间的转移要么是一直都 $+1$，要么是直接到达某一项为零的状态。
- 可以只记录所有某一项为零的状态的真实 DP 值，然后直接考虑这些状态间的转移，具体来说直接考虑进行多少次都 $+1$ 操作后汇聚到一个 $(0, *)$ 的状态或 $(*, 0)$ 的状态。
- “汇聚”转移时有转移前两个二进制数无交的限制，这保证了有效的转移只有 $O(3^k)$ 个，只做有效的转移可以做到 $O(t3^k)$。

优化方向一：压位

- 最后观察到 a, b, c 只控制一些潜在可行的转移是否真的进行，因此可以将代码写成“永远执行所有 $O(3^k)$ 种潜在转移，而在转移时与上这个转移的条件”的形式。
- DP 状态只有 1 bit 的信息，且每组数据中转移结构完全相同，因此可以压位在一轮计算内同时维护 w 组数据的 DP 值，其中 w 是计算机字长。
- 精细实现后，时间复杂度为 $O(t2^k + \lceil t/w \rceil 3^k)$。

优化方向二：分治

- 不考虑压位，每次取一个极长的 b 相同的连续段进行转移。
- 一种与上述介绍的 DP 方式殊途同归的状态设计会需要解决以下问题：
- 设 g, f 为序列，g_i 由 $f_{1\sim i}$ 确定。需要处理所有转移 $g_j \to f_i$ $(j < i, p(i,j) := (i+j-1)\,\mathrm{and}(2(2^k - i - 1)) = 0)$。
- 考虑分治解决 $[l, r)$ 内的转移，不妨设 $r - l = 2^t$。
- 分治时，需要额外计算 $h_{[l,r)}(x, y) := \sum_{j\in[l,r)} [p(y, j) \bmod 2^t = 0][\lfloor \frac{y+j-1}{2^t} \rfloor = x] g_j$，其中 $x \in [\lfloor \frac{r}{2^t} \rfloor - 2, \lfloor \frac{r}{2^t} \rfloor], y \in [0, 2^t)$。
- 分治过程如下：
  - 递归解决 $[l, l + 2^{t-1})$。
  - 利用 $h_{[l,l+2^{t-1})}(\cdot,\cdot)$，处理 $[l, l + 2^{t-1})$ 对 $[l + 2^{t-1}, r)$ 的贡献。
  - 递归解决 $[l + 2^{t-1}, r)$。
  - 计算 $h_{[l,r)}(\cdot,\cdot)$，共 $O(2^t)$ 个值。

优化方向二：分治

- 实际情况会稍麻烦一些，但总之可以在 $\Theta(len \log len)$ 时间内完成一个长度为 len 的连续段的转移。
- 如果使用之前介绍的状态设计，也可以类似地分治记录信息来优化。
- 时间复杂度为 $\Theta(t2^k k)$。

结合起来

- 事实上两个优化方向可以结合，具体来说分治合并信息时可以压位做到 $\Theta(\frac{2^t}{w})$ 完成长度为 2^t 的区间的信息合并，当然在此之前要先递归到左侧，在此之后要递归到右侧。
- 结合起来，可以在 $\Theta(2^k)$ 时间内计算一组数据的答案，总复杂度为 $\Theta(t2^k)$。