CCPC Qinhuangdao 简要题解

中山大学程序设计竞赛命题组

2023 年 10 月 15 日

前言

- 预计难度：$AG<FJ<DI<M<BCL<E<HK$
- 封榜前难度：$AG<DFJ<M<CI<BELK<H$
- D 比预期过的队伍更多
- M 比 I 更早开出来，导致榜稍微有点歪

G. path

- 题意：在二维矩阵中选一条从左上到右下的路径，使得相邻差值和最大。
- 相邻两个不同的限制是无所谓的，因为对答案没有影响。
- 又因为根据绝对值的性质，有 $|A-B| \leq |A-C| + |C-B|$，所以可以发现每次往右或往下走一步是最优的。
- 而由于权值 $C_{x,y} = a_x + b_y$，所以 $|C_{x',y'} - C_{x,y}|$ 实际等于 $|a_{x+1} - a_x|$ 或 $|b_{y+1} - b_y|$，
- 并且无论路径如何，答案都等于 $\sum_{i=1}^{n-1} |a_{i+1} - a_i| + \sum_{i=1}^{m-1} |b_{i+1} - b_i|$，故直接输出即可。

A. Make SYSU great again I

- 题意：构造一组填数方案，使得第 i 行等于第 i 列的 gcd。
- 按如下方式构造即可，因为任意行和列 gcd 都为 1

	A	B	C	D	E	F
1	1	2				
2		3	4			
3			5	6		
4				7	8	
5					.	.

D. Yet Another Coffee

- 题意：n 个物品，和 m 张优惠券，每个优惠券能用在一个前缀的物品上，对每个 k 求购买恰好 k 个物品的最小花费
- 简单贪心，考虑最靠前的优惠券：
  - 如果它要使用，可反证一定是用在其前缀价格最小的物品上
  - 如果它不使用，则其前缀的物品不会被购买，怎么减无所谓
- 故直接将对其前缀价格最小的物品使用该优惠券即可。
- 最后将所有物品排序，前 k 个物品的价格和即为答案。
- 除去排序，总复杂度 $O(n)$。

J. Book

- 题意：每天必须把相同长度的单词学完，问最少天数把所有单词学完
- 单词长度很小，那么我们可以考虑通过状压 dp 求出答案。
- 对于某个集合 S，其中 x 属于 S 表示长度为 x 的单词已经学过。
- 设 dp_S 为学习完 S 的集合长度的单词的最小天数，转移的时候枚举每个 S 的子集即可，复杂度 $O(n + d3^d)$。

F. ministry of prime

- 题意：修改最少的数字，使得相邻两个元素的和是质数
- 首先介绍" 波利尼亚克猜想"——对所有自然数 k，存在无穷多个素数对 $(p, p+2k)$。
- 据此猜想，我们可以有假设: 对于任意奇偶性相同的 x, y, 存在无穷个 z, 使得 $x+z, y+z$ 为质数.
- 设 $z(x, y)$ 是**最小**的 z, 使得 $x+z, y+z$ 是质数.
  通过暴力程序验证, $\forall(x, y \leq 10^5, x \equiv y(\mod 2)), z(x, y) \leq 1140$.
- 这样调整一个数就能和左右**相同奇偶性**的数相容, 而且实际上调整的数也在 10^5 以内（题目放松了限制）。

F. ministry of prime

- 因为 $1+1=2$ 是特殊的质数，考虑 $dp[i][j]$：
  - $j=0$：$a[i]$ 不变
  - $j=1$：$a[i]$ 改成 1
  - $j=2$：$a[i]$ 改成偶数
  - $j=3$：$a[i]$ 改成 >1 的奇数
- 讨论转移即可，时空复杂度 $O(n)$。

I. Phony

- 题意：给定 n 个数，k 固定，有两类操作:
  - 操作 1, 将最大的数减去 k，重复执行 t 次
  - 操作 2, 求全局第 c 大的数值

I. Phony

- 假设 $k = 1$，则等价于每次将最大值减一。
- 按数值从大到小模拟即可，只需记录当前最大值，以及最大值个数。

I. Phony

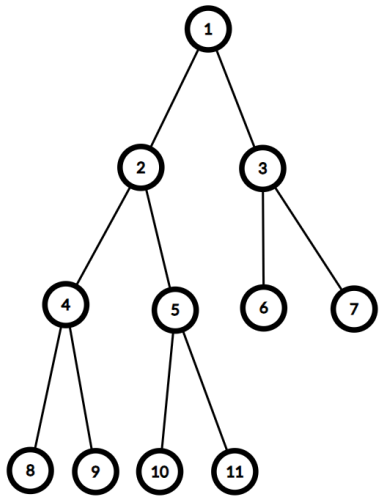
- 对于 $k > 1$ 采用相同的思路。
- 等价于将数 x 拆成一对 $(x/k, x\%k)$，每次先将 $/k$ 最大的段的最大 $\%k$ 对应的 $/k$ 值减一。
- 当 $/k$ 最大的段所有数字都被减过之后，会合并到下一个 $/k$ 的段中。
- 线段树合并/启发式合并/倒叙分段插入都可通过，复杂度 $O((n+q)\log n)$ / $O((n+q)\log^2 n)$

M. inverted

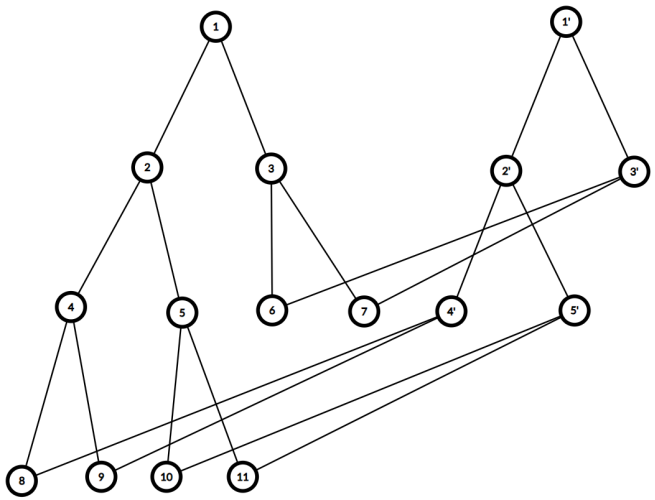
- 粗略题意：多次新加点 x+n，复制节点 x 的连边，求生成树个数。
- $n, q \leq 5000$

M. inverted

- 考虑一棵树经过若干次操作会变成什么，如图所示，对点 $1...5$ 操作之后得到



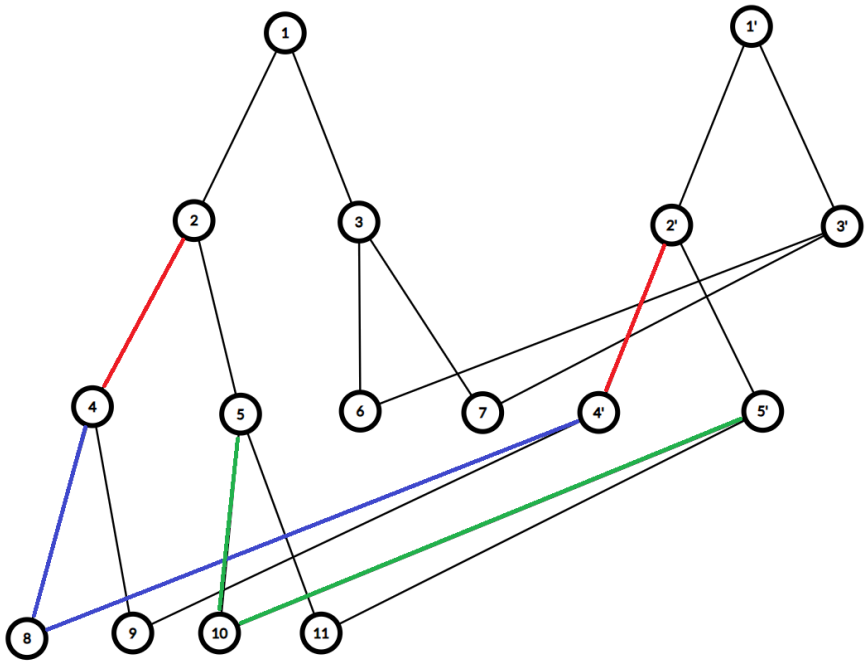
(a) Before



(b) After

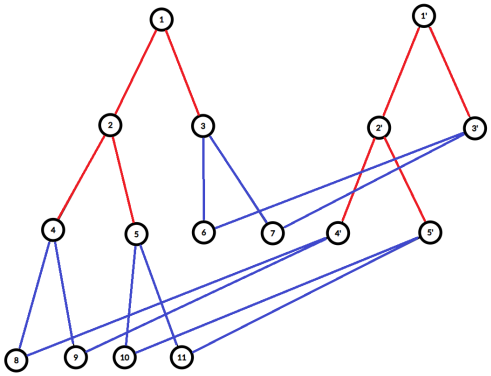
M. inverted

- 记操作过的一对点为成对点（如 1 和 $1'$），剩下的为独立点。求生成树等价于求有多少种断边方案，使得结果是一棵树；考虑上面的图怎么算，由于 $1...5$ 和 $1'...5'$ 是完全对称的，所以可以把两棵树上的边对应起来形成若干对边，每一组边删第一条或第二条等价（删两条会不连通）：

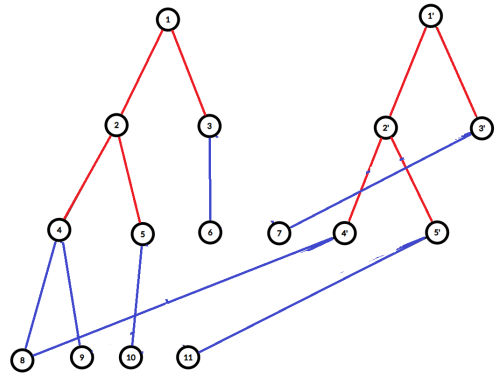


M. inverted

- 记 $1-2,1'-2'$ 这样的边（成对点-成对点）为红边，其余的边（成对点-独立点）为蓝边，观察一下每种断边的情况：



(c) Before

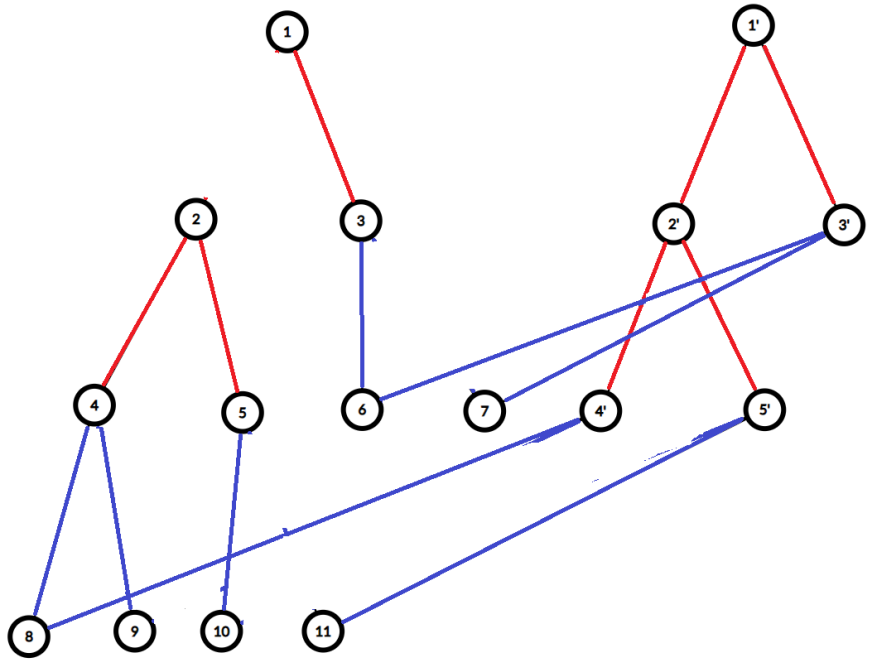


(d) After

假设完全不删红边，那么上图除了每一对蓝边保留两条以外，其余的每对蓝边都要删掉其中一条：
如图，保留了 $4-8$ 和 $4'-8$，其余每对蓝边删掉一条边，得到一棵合法的生成树（记为方案 1）；

M. inverted

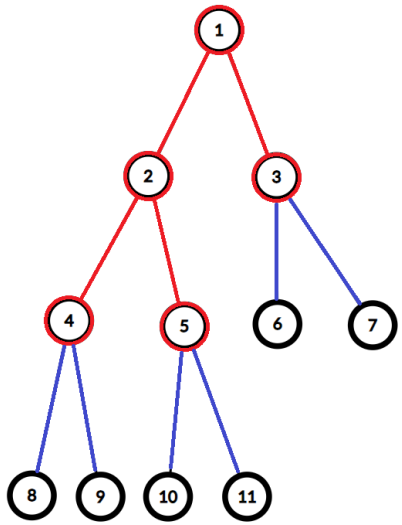
- 考虑删掉一条红边的情况，例如删掉 $1-2$，那么一组合法的解（记为方案 2）为：



即多保留了一条 $3'-6$，不然会断成两个块。

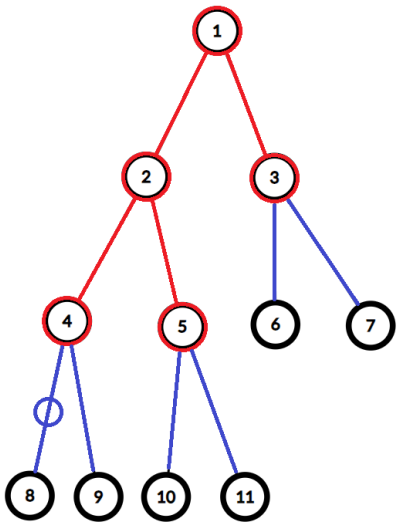
M. inverted

- 现在回到原树观察上述操作，记复制过的点为红点，未复制过的为黑点，红蓝边分别代表实际图种的一对边：

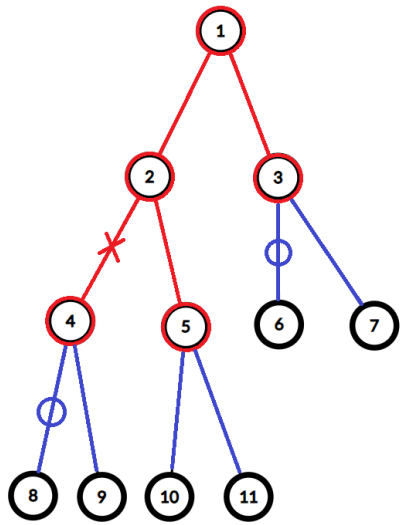


M. inverted

- 记蓝圈为保留一组蓝边（其余蓝边的删其中一条），红叉为删掉一组红边的一条（其余红边不动），则对于方案 1 和方案 2 的删法如下：



(e) 方案 1



(f) 方案 2

M. inverted

- 可以发现，对于每个把标叉红边删掉形成的红点连通块里面，都要有恰好一个点存在恰好一条相连的标圈蓝边。
- 这样问题就简化了，对于每个红点（操作过的点）形成的块 dp，设 $f[i]$ 表示当前点 i 所在的连通块内存在某个相连的标圈蓝边，$g[i]$ 表示不存在，从子树往上合并即可。
- 需要注意上述边实际对应的是一对边，也就是删红边和蓝边时的方案都要 $\times 2$。
- 对于每次新增操作点后的所有块跑一边 dp，时间复杂度 $O(n^2)$。

B. Yet Another Subsequence Problem

- 等价题意：按照给定方式生成 01 串，求子序列个数
- 生成方式可以理解为从 $(0,0)$ 沿折线走到 (A,B)
  - 默认向上走，代表'1'
  - 当在直线 $(0,0)-(A,B)$ 的上方时向右走，代表'0'

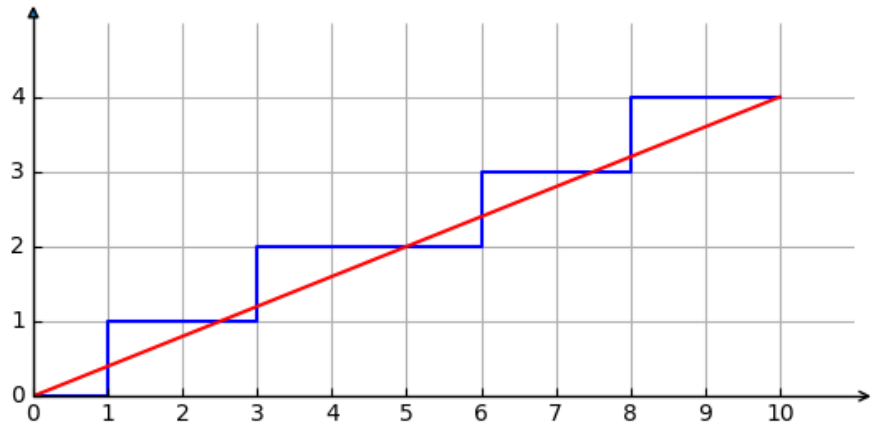


图 1：$gen\_string(10,4) = 01001000100100$

B. Yet Another Subsequence Problem

- 首先求子序列数量可以通过矩阵乘法求得。
- 设 m_0 表示 0 元素对应的矩阵，m_1 类似。
- 考虑怎么优化矩阵乘法，假设 $solve(A, B, m_0, m_1)$ 表示答案，由生成方式可知
  - 当 $B \geq A$，0 后面一定会紧跟至少 $\lfloor \frac{B}{A} \rfloor$ 个 1，将 $011...1$ 看成是新的 0 元素，变成子问题：$solve(A, B\%A, m_0 \cdot m_1.pow(\lfloor \frac{B}{A} \rfloor), m_1)$
  - 当 $B < A$，1 后面一定会紧跟至少 $\lfloor \frac{A-1}{B} \rfloor$ 个 0，将 $100...0$ 看成是新的 1 元素，变成子问题：$solve(A - \lfloor \frac{A-1}{B} \rfloor B, B, m_0, m_1 \cdot m_0.pow(\lfloor \frac{A-1}{B} \rfloor))$
- 只看前两维是一个类欧的形式，复杂度 $O(\log n)$。

C. Palindrome

- 题意：给定一个字符串，区间询问有多少种方案删掉最短的区间使得变成回文串

C. Palindrome

- 对于一个区间 $[l, r]$ 我们首先可以先让首尾相同字符不断相消，最后剩余区间 $[x, y]$，如果这个区间为空则为回文串。
- 这一部分等价于求原串后缀 l 和反串后缀 $n - r + 1$ 的最长公共前缀（ LCP ），再对区间长度取 $\min$。
- 如果不为回文串，那么 $str[x] \neq str[y]$，x, y 其中之一会被删除。
- 删除最少长度，等价于保留最多长度，即保留 x 开头的一个回文串或 y 结尾的一个回文串。
- 这是经典的区间最长回文子串问题，可通过在回文自动机（ PAM ）上倍增跳 $fail$ 实现。

C. Palindrome

- 假设删除 $[x, x+t]$ 是一个最优解，在求方案数时，我们还可以在保持**剩余字符串**不变的前提下，将此区间向左进行滑动。
- 具体地，$str[x-1] = str[x+t]$ 则左移一次，$str[x-2] = str[x+t-1]$ 则左移两次...
- 可以发现这里的左移次数就是求两个反串后缀的 LCP，由于 l 的存在，我们还要对 $x-l$ 取 $\min$。
- 类似地，$[y-t, y]$ 也可以进行右移，且和左边的区间**不交**，所以可以直接相加。
- 可以通过画图简单地证明只有这种形式的串才能满足最短的要求。
- 求 LCP 可以用 [二分 + 哈希] 或者运用后缀数据结构.
- 加上 PAM 倍增的预处理，所以时空复杂度均为 $O(n \log n)$.

L. Yet Another Maximize Permutation Subarrays

- 题意：给定一个排列，交换两个位置使用子排列个数最多
- 设下标数组 $p[i] = j$ 当且仅当 $a[j] = i$，则
  - 在原数组交换两个位置和在下标数组交换两个位置是等价的
  - 原数组的一个子排列，对应于下标数组的前缀连续段
- 故后续我们只在下标数组上讨论

L. Yet Another Maximize Permutation Subarrays

- 设在下标数组交换的两个位置为 $x < y$，只会影响 $i \in [x, y)$ 的前缀连续段情况
- 先特别计算 $x = 1$ 的情况
  - 对于每个前缀 i，可以 $O(1)$ 求得 $p[2...i]$ 加入什么数字后会变成连续段
  - 枚举到 i 的时候，可以通过前面的贡献求得 $y = i$ 时的答案

L. Yet Another Maximize Permutation Subarrays

- 当 $x > 1$ 时，设 $p[1...x]$ 对应的值域为 $[l[x], r[x]]$
- 考虑交换 x, y 哪些前缀会变成新的连续段
- 对于 $i \in [x, y]$，设 $p[1...i]/p[x]$ 的值域 $[l, r]$ 是已知的
  - 可以 $O(1)$ 求得 $[l, r]$ 加入什么数字后会变成连续段
  - 枚举到 y 的时候，可以通过前面的贡献求得答案
- 实际上若 $p[x] \neq l[i]$ 且 $p[x] \neq r[i]$，则去掉 $p[x]$ 会使得前缀 i 一定不是连续段，显然不优
- 故只会交换 $\forall i \in [x, y]$，满足 $p[x] = l[i]$ 或 $p[x] = r[i]$ 的 x, y
- 等价于 $p[x] = l[y]$ 或 $p[x] = r[y]$，则对于每个 y，只有 $O(1)$ 个合法的交换对，总合法交换对数 $O(n)$

L. Yet Another Maximize Permutation Subarrays

- 考虑交换 x, y 哪些原本是连续段的好前缀会消失
- 由于对称性这里只讨论 $p[x] = l[x]$:
  - $i \in [x, y]$ 中满足 $l[i] = p[x]$ 的好前缀都会消失
  - 又因为在 $[1, x)$ 中，不存在 $l[i] = p[x]$
  - 故等价于 $i \in [1, y]$ 中满足 $l[i] = p[x]$ 的好前缀都会消失
- 可以发现
  - 若 $l[y] = p[x]$，即 $[1, y]$ 中满足 $l[i] = l[y]$ 的好前缀都会消失
  - 若 $l[y] \neq p[x]$，则
    - $l[i] = l[y]$ 一定在 y 之前
    - 等价于满足 $l[i] = p[x]$ 的好前缀都会消失
- 也即消失的好前缀只和 x 或 y 有关

L. Yet Another Maximize Permutation Subarrays

- 综上：
  - 满足交换 x, y 会新增新的连续段只有 $O(n)$ 对。
  - 可 $O(1)$ 检验交换之后的连续段个数
- 总复杂度 $O(n)$

E. coloring tape

- 粗略题意：给定一个狭长子带，刷子只能一直往前染色，不能往回走，求本质不同染色个数。

E. coloring tape

- n 很小，不难想到对每一列，将刷子所在的位置用二进制位进行表示，从而将一列的状态映射到 $\{0,1\}^n$ 上进行 dp。
- 可以发现，对于一个状态，我们只需要认为每个刷子的颜色都是互不相同的，而不需要具体得到刷子实际上染了什么颜色，所以颜色属性在状态中不重要。

E. coloring tape

- 考虑从一列转移到另外一列的过程：
  - 首先，由于在推进的过程中，颜色可能会消失，我们需要决定哪些刷子留在这一列，哪些刷子前往下一列进行染色。这在 dp 数组中可以通过一次高维后缀和得到。
  - 其次，对于下一列，我们需要移动这些刷子，使得这一列上所有格子都被染色。
  - 我们将其转化为如下问题：
    - 对于一个 ‘01‘ 序列，需要进行若干次操作，每次操作需要选择一组相邻的 ‘0‘ 和 ‘1‘ 并进行翻转（对应染色操作）。最后需要满足每个 ‘0‘ 都被交换恰好一次（对应不能重复染色）。

E. coloring tape

- 通过对始末状态的分析，不难发现这等价于让原先的序列中每个 ‘0‘ 向左或者向右移动一位，在移动不超界的前提下保证 ‘0‘ 位置的顺序不发生改变。
- 同时，根据状态前后 ‘1‘ 位置的序列亦可以得到这次染色对应的颜色序列。
- 随后枚举所有可能的转移，根据颜色序列判断在该列上转移的可行性，即可得到下一层的 dp 数组。
- 假设可能的转移有 M 个，则最终复杂度为 $O(n3^n + (m + r)M + nm2^n)$。

H. Quake and Rebuild

- 题意：给定一棵树，每个点 $i(i > 1)$ 有一个 $fa_i(fa_i < i)$ 表示 i 的父亲.
- 有两类操作:
  - 操作 1，输入 1 l r $delta$，$\forall i \in [l, r], fa'_i = max(1, fa_i - delta)$.
  - 操作 2，输入 2 k a_1 a_2... a_k ，求 $a_1, a_2, ..., a_k$ 的虚树大小.

H. Quake and Rebuild

- 分块题，以 $\sqrt{n}$ 为块长，$nxt(i)$ 表示从 i 出发到达的第一个**异块**祖先. 记 Δ_t 表示块 t 整体的减少量.
- 每次修改：
  - 对于整块的部分，当 $\Delta_t < \sqrt{n}$ 时，我们暴力重构区间 nxt，当 $\Delta_t \geq \sqrt{n}$ 时，我们显然可以一步跳出块，所以我们不需要重构，每次计算父节点直接考虑 $fa_i - \Delta_t$. 每个块最多重构 $O(\sqrt{n})$ 次，总复杂度 $O((\sqrt{n})^3) = O(n\sqrt{n})$.
  - 对于非整块的部分，我们也需要重构整块，这部分复杂度 $O(m\sqrt{n})$.

H. Quake and Rebuild

- 考虑一个简单的问题，怎么快速求 $LCA(x, y)$.
- 不妨设 $x > y$, 当 $nxt(x) \neq nxt(y)$, 我们直接令 $x = nxt(x)$.
- 否则:
  - 如果在同一个块内，则 LCA 一定在该块内，直接暴力找即可，复杂度 $O(\sqrt{n})$。
  - 如果不在同一个块内，则 LCA 就是 $nxt(x)$。
- 暴力往上跳最多 $O(\sqrt{n})$ 次，块内暴力找 LCA 最多一次，总复杂度 $O(\sqrt{n})$。

H. Quake and Rebuild

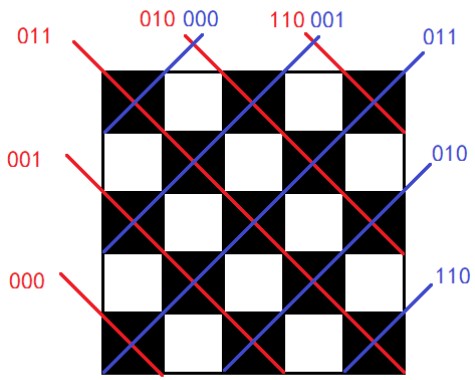
- 回到原题，求虚树可以理解为所有节点一起往上跳找 LCA。
- 套用上述的做法，可以改成从大到小遍历整块：
  - 若该整块内没有点，则可直接忽略
  - 若该整块内所有点的 nxt 各不相同，则各自往上跳到 nxt
  - 若存在两个点的 nxt 相同，则
    - 对该块内的所有点一步一步往上跳，暴力计算该块内遍历到的点的个数即可
    - 每当发生该次情况，等价于以 $O(\sqrt{n})$ 的代价合并至少两个点
    - 最多只有 $\sum k$ 个点，该部分暴力复杂度 $O(\sqrt{n}\sum k)$。
- 实现细节比较多
- 总时间复杂度 $O((n+m+\sum k)\sqrt{n})$，空间复杂度 $O(n)$.

K. Make SYSU great again II

- 题意：构造一组填数方案，使得相邻数字的 *and* 为 0，且每个数字不超过 $4n^2-1$，最多出现 5 次

K. Make SYSU great again II

- 一种可行构造方法，设 $K = \lceil \log_2(n) \rceil$：
  将 $n \times n$ 的网格进行黑白染色，对于黑色的格子将其两条斜线列出，对于两个方向的斜线分别按顺序填上格雷码：



- 将每个黑格子的额两条线对应的数拼接，作为该格子的值。
- 然后其余白格子的值为四周黑格子的值 or 起来再对 $2^{2K} - 1$ 取 *xor*。
- 该构造显然满足相邻 *and* 为 0 的限制。

K. Make SYSU great again II

- 下面证明正确性：
- 对于值域限制：
  - 显然两种对角线的个数不超过 n，那么格雷码的范围为 $[0, 2^K - 1]$，两条对角线合并到黑格子之后范围为 $[0, 2^{2K} - 1]$，不超过 $[0, 4n^2 - 1]$；
  - 白格子是 $[0, 2^{2K} - 1]$ 对 $2^{2K} - 1$ 取反，范围不变，故满足了值的范围限制。
- 对于次数限制：
  - 黑格子由于两条对角线编号唯一，合成的编号也唯一，故每种数最多出现一次；
  - 对于白格子，发现白格子四周的黑格子的 or，实际就是白格子四周两种对角线分别取 or 然后拼接。

K. Make SYSU great again II

- 实际上，取同一个值的白格子不超过 4 个。
- 证明要用到格雷码的性质，因为相邻的格雷码的位差距恰好为 1，所以有 $gray_i \text{ or } gray_{i+1} = \max(gray_i, gray_{i+1})$。
- 然后一条红/蓝对角线最多作为 *max* 影响两条对角线的白格子，那么红蓝对角线的影响拼起来后最多把交叉的 4 个白格子设为一样的值；
- 综合黑格子的 1 次出现，总出现次数不超过 5。
- 如果不用格雷码的话，例如按顺序构造，此时不能把 *or* 改成 *max*，那么次数就可能超过 5 次，