The 3rd Universal Cup. Stage 16: Nanjing

SUA 程序设计竞赛命题组

2024 年 11 月 3 日

概况

- 预期难度：
  - Easy：E、J；
  - Easy-Medium：B、K；
  - Medium：G、I、M；
  - Medium-Hard：A、C、F；
  - Hard：D、H；
  - Impossible：L。
- 实际通过人数排序：
  - EJKGBCIMFAHDL。

E. Left Shifting 3

题意

- 给一个字符串，可以左移至多 k 次，最大化左移之后 `nanjing` 子串的数量。
- $n \leq 2 \times 10^5$，$0 \leq k \leq 10^9$。

- 两个 `nanjing` 子串不可能相交，因此只要左移之后，字符串的开头不会“截断”一个 `nanjing` 子串，就能得到最大的答案。
- nanjing 子串的长度是 7，所以左移 7 次之内肯定有个位置不会截断 `nanjing` 子串。因此模拟 $\min(k, 7)$ 次左移并计算答案即可。
- 时间复杂度 $\mathcal{O}(n)$。

J. Social Media

题意

- 一个社交平台有 k 个用户，其中有 n 个是你的好友。平台上还有 m 条评论。每个评论与两个用户 a_i, b_i 相关，可以看到该评论当且仅当你同时和 a_i 与 b_i 两位用户是好友。现在可以添加至多两个用户为好友，求出能看到的评论数量的最大值。
- $n \leq k \leq 2 \times 10^5$，$m \leq 2 \times 10^5$。

- 将评论分为三类：
  - 相关的两个用户都是初始存在的好友；
  - 相关的用户中恰有一个是初始存在的好友；
  - 相关的两个用户都是新添加的好友。

J. Social Media

- 对于第一类评论，在任何情况下都可见，直接统计出这样的评论个数即可。
- 对于第二类评论，我们可以对于每个非初始好友 u 统计 c_u，表示与 u 和初始好友相关的评论数量。则可见的第二类评论的数量是新添加的用户的 c 值之和。
- 若不存在可见的第三类评论，则应当按照 c 值从大到小的顺序贪心选择新增的好友。若存在可见的第三类评论，枚举该评论后，与其相关的两个用户必须都是新增的好友，也即可以完全确定新增的好友是哪两位，于是将其 c 值相加后更新答案即可。
- 时间复杂度 $\mathcal{O}(n+m)$。

K. Strips

题意

- 有 w 个格子排成一行，从左到右编号从 1 到 w。这些格子中，有 n 个是红色的，m 个是黑色的，剩下的格子是白色的。
- 接下来需要用一些不相交的纸条覆盖所有红色格子。每张纸条必须覆盖 k 个连续的格子。每个红色格子都要被覆盖，每个黑色格子都不能被覆盖，且纸条数要尽量小。构造一个方案。
- $n, m \leq 10^5$，$w \leq 10^9$。

- 黑色格子把所有格子分成了几段，我们可以对每段分别求解。
- 分段后，问题变为：一些格子排成一行，其中一些格子是红色的。用最少纸条覆盖所有红色格子，且纸条不超过边界。

K. Strips

- 先假设没有边界的限制，只最小化纸条数量。这是一个非常经典的贪心问题：从左到右枚举红色格子，若当前格子未被覆盖，则添加一张以该格子为左端点的纸条。
- 加入边界后可能出现的问题只有：最后一张纸条的右端点超过了右边界。那我们尝试调整一下，把最后一张纸条的右端点和当前段的右端点对其。
- 调整之后，该纸条可能会和上一张纸条发生重叠，那么把上一张纸条也往左移，依次调整。
- 如果调整完之后，第一张纸条超出了左端点则无解。因为这说明最少纸条数乘以纸条长度已经大于本段长度。否则我们就得到了一个合法解。
- 复杂度 $\mathcal{O}((n+m)\log(n+m))$，主要是排序的复杂度。

G. Binary Tree

题意

- 交互题。给定一棵 n 个点的二叉树，树上有一个特殊节点 s，设 $d(u, v)$ 表示树上两点的距离。每次询问提供两个点 u, v，返回 $d(u, s)$ 和 $d(v, s)$ 的大小关系。
- 在 $\lfloor \log_2 n \rfloor$ 次询问内求出 s 的编号。
- $n \leq 10^5$。

G. Binary Tree

- “交互的本质是二分”。树上怎么二分呢？考虑树的重心。
- 节点为 n 的树重心 u 满足：去掉 u 之后，剩下的连通块点数小于等于 $\frac{n}{2}$。因为给定的是二叉树，所以我们可以对 u 的度数进行讨论。
  - u 度数为 0：答案就是 u。
  - u 度数为 1：说明 $n = 2$，询问 u 和邻居即可知道答案。
  - u 度数为 2：询问 u 的两个邻居，即可把问题规模缩小至 $\lfloor \frac{n}{2} \rfloor$。
  - u 度数为 3：询问三个连通块里较大的两个。最小的那个连通块加一，一定小于等于 $\lfloor \frac{n}{2} \rfloor$，可以按 n 的奇偶性讨论来反证 $3 \times \lfloor \frac{n}{2} \rfloor + 1 \leq n$ 是不满足的。
- 因此每次询问可以把问题规模从 n 变成 $\lfloor \frac{n}{2} \rfloor$，就能在 $\lfloor \log_2 n \rfloor$ 次询问内找到答案。

B. Birthday Gift

题意

- 给定一个由 012 组成的字符串。将所有 2 修改为 0 和 1 中的一个，然后不断删去字符串中两个相邻的相同的字符。求最后可能得到的最小字符串长度。
- $n \leq 2 \times 10^5$。

- 将字符串所有偶数位置取反。具体地，对于所有偶数下标的字符，若原来该字符为 0 则修改为 1，反之修改为 0。

B. Birthday Gift

- 则原串中的两个相邻的相同的字符，对应了新串中两个相邻的不同的字符。不断删去两个相邻的不同的字符直到不能删除，得到的字符串要么全部为 0，要么全部为 1。这是因为，如果字符串中同时存在 0 和 1，则必然有一个 0 和一个 1 相邻，于是可以继续进行操作。
- 每次操作会删除一个 0 和一个 1。故最终的字符串长度是初始字符串中两种字符个数的差。对于每个 2，将其改为当前出现次数较小的那种字符，即可保证出现次数的差尽量小。
- 时间复杂度 $\mathcal{O}(n)$。

C. Topology

题意

- 给定一棵由 n 个点组成的外向树，满足父亲的编号小于儿子。对于每个 $1 \leq i \leq n$，求出这棵树的满足编号为 i 的点出现在第 i 个位置的拓扑序数量，对 998244353 取模。
- $n \leq 5000$。

- 考虑对于所有 i, j 求出 $f(i, j)$，表示编号为 i 的点在拓扑序中出现在第 j 位的方案数。答案即为所有的 $f(i, i)$。

C. Topology

- 为了方便转移，设 $g(i,j)$ 表示不考虑子树 i 内部的顺序时，点 i 出现在第 j 个位置的方案数。从 $g(i,j)$ 乘上子树 i 的拓扑序个数即可得到 $f(i,j)$。设子树 u 的大小为 s_u。
- 设点 u 的父亲为 p，考虑从 $f(p,x)$ 转移到 $f(u,y)$ 的转移系数。
- 首先，需要确定 u 子树中的 s_u 个点，在剩余的 s_p-1 个空位中的位置。这里需要保证最靠前的点，也就是 u，出现在第 y 个位置，所以选择的方案数为 $\binom{n-y}{s_u-1}$。
- 其次，需要确定在 p 子树中，但不在 u 子树中的 s_p-s_u 个点在拓扑序中的相对顺序。一棵 n 个点的外向树的拓扑序个数为：

$$\frac{n!}{\prod_{i=1}^{n} s_i}$$

- 在预处理 s_i 的乘积之后容易计算。

C. Topology

- 暴力进行转移的复杂度不可接受。
- 观察转移的系数，发现这个系数与 x 无关，仅需要保证 $x < y$ 即可。于是对 $f(p, x)$ 进行前缀和之后可以 $\mathcal{O}(1)$ 进行一个状态的转移。
- 时间复杂度 $\mathcal{O}(n^2)$。

I. Bingo

题意

- 给定一个长度为 nm 的整数序列，将其打乱之后按顺序填入 $n \times m$ 的网格中。定义 bingo 整数为最小的整数 k，满足存在至少一行或者一列，其中所有的数都 $\leq k$。求出所有 $(nm)!$ 种打乱方式的 bingo 整数之和，对 998244353 取模。
- $nm \leq 2 \times 10^5$。

- 定义一行或一列的 bingo 整数为其中填入的数的 $\max$，则所求的 bingo 整数为所有行列的 bingo 整数的 $\min$。
- 对于 $\min$ 不好处理的情况，考虑使用 $\min - \max$ 容斥。

- 回顾 $\min - \max$ 容斥的形式：

$$\min_{x \in S} x = \sum_{T \subseteq S, T \neq \varnothing} (-1)^{|T|-1} \max_{x \in T} x$$

- 在本题中 S 是所有行和列的集合。$\max_{x \in T} x$ 为所有 T 中的行和列的 bingo 数的最大值。回顾 bingo 数的定义可知，这个值是所有被 T 中行和列覆盖到的格子中的数的最大值。
- 由于填数时所有格子互不区分，在所有 $(nm)!$ 种方案中这个值的和，仅与被覆盖到的格子数量有关，设这个数量为 c。

I. Bingo

- 将序列从小到大排序。c 个格子中的最大值为 a_i，当且仅当 a_i 在这 c 个格子中出现，且 $a_{i+1}, a_{i+2}, \ldots, a_{nm}$ 均没有出现，方案数为：

$$\binom{i-1}{c-1} c!(nm-c)!$$

- 将组合数拆开成阶乘的形式，可以分成三部分，分别只和 $i, c, i+c$ 有关，可以使用卷积加速求值。
- 对每个 c 求出答案之后，枚举 T 中选择了 x 行 y 列，则选择 T 的方案为 $\binom{n}{x}\binom{m}{y}$，c 为 $mx+ny-xy$，将所有情况的答案乘上容斥系数求和即可。
- 时间复杂度 $\mathcal{O}(nm\log(nm))$。

M. Ordainer of Inexorable Judgment

题意

- 水神的攻击范围是一条以原点为端点的射线，和所有距离射线 $\leq d$ 的点的集合。射线的初始方向向量给定，会以每个单位时间 1 弧度的顺序旋转。给定一个 n 个顶点的凸多边形，求区间 $[0, t]$ 中攻击范围与目标有交的时间区间长度。
- 保证凸多边形目标距离原点 $> d$。
- $n \leq 100$。

- 首先计算能攻击到目标的极角范围。能攻击到目标当且仅当存在射线上的一个点距离凸多边形 $\leq d$。考虑将凸多边形扩展 d 的大小。具体地，将凸多边形的每个顶点扩展成一个半径为 d 的圆，连接形成一个顶点圆滑的凸多边形。能攻击到目标当且仅当射线与这个新的图形相交。

M. Ordainer of Inexorable Judgment

- 为了求出可行的极角区间，只需要求出极角最小和最大的切线的极角即可。过原点的新图形的切线，一定与某个顶点扩展成的圆相切。
- 求出原点过所有圆的切线。由于区间可能会跨过 2π 的位置，无法直接求最大最小值。但由于题目保证凸多边形距离原点 $> d$，也就是必然存在一个半平面满足凸多边形扩展成的图形在这个半平面内。在半平面内取出最靠左和最靠右的切线即可。
- 将时间区间 $[0, t]$ 切分成若干个整周期 $[0, 2\pi]$ 和一个不整的区间 $[0, \theta]$，与可行区间求交即可计算答案。
- 时间复杂度 $\mathcal{O}(n)$。

F. Subway

题意

- 有 n 个地铁站和 k 条地铁线路，可以乘坐地铁或者在某站处换乘。乘坐第 i 条线路的第 j 段需要 $w_{i,j}$ 单位时间。从某站的 x 号线路换乘到 y 号线路需要 $a_x \cdot b_y$ 单位时间。求出从站点 1 到每个站点的最短路。
- $n, k, \sum(p_i - 1) \leq 2 \times 10^5$。

- 考虑用状态 (i, j) 表示当前在第 i 个站，并在第 j 条线路上。在状态间求最短路来得到答案。
- 乘坐地铁的转移是简单的，只需要从 (i, j) 转移到第 j 条线路的下一站 x 对应的状态 (x, j) 即可。

F. Subway

- 考虑如何处理换乘的转移。将一个站点的线路按照 b 从小到大排序，可以发现转移到 b 较小的线路一定代价比 b 较大的线路小，也就是说，b 较小的状态一定先于较大的状态被加入优先队列。于是可以考虑进行懒惰转移，在每个结点较小的 b 被转移出去之后再考虑下一条线路。
- 对于一个结点上换乘的转移，相当于是给定若干直线，查询某个 x 处的最小值。使用李超树或者动态凸包可以维护。
- 时间复杂度 $\mathcal{O}((n+k+\sum p_i)\log(n+k+\sum p_i))$。

A. Hey, Have You Seen My Kangaroo?

题意

- 有一个 n 行 m 列的网格，其中有一些位置是墙，其余位置是袋鼠。
- 给定一个长度为 k 的由 `LRUD` 组成的操作序列，不断循环执行这个操作序列。对于每个 i，求出使得含有袋鼠的格子数量 $\leq i$ 的最早时刻。不可能达到输出 `-1`。
- $nm \leq 2 \times 10^5$，$k \leq 200$。

- 可以将问题转化成：对于每个 i，求出进行了前 i 步操作之后剩余的含有袋鼠的格子数量。

A. Hey, Have You Seen My Kangaroo?

- 先求出所有整段周期处的答案，也就是求出所有 k 的倍数时刻的答案。设位于格子 u 的袋鼠进行一个周期的操作后到达格子 go_u，连出所有有向边 $u \to go_u$。所有点的出度都为 1，整张图构成一个内向的基环树森林。
- 每进行一个周期的操作之后，一个袋鼠会移动到基环树上的父亲结点处。求出每个点 u 的子树的高度 h_u，u 点在第 $[1, h_u]$ 个整周期后存在袋鼠。对于环上的点，每个整周期后袋鼠都会存在，可以看做环上的点的 $h_u = +\infty$。

A. Hey, Have You Seen My Kangaroo?

- 然后求出非整段周期处的答案。先求出第一个周期中的合并情况。模拟前 k 次操作，记录每次操作导致的袋鼠的合并，可以求出第一个周期每个时刻的答案。
- 对于之后的周期，它们的合并情况与第一个周期相似。具体地，对于模拟的前 k 次操作中的每次合并，设合并发生的时刻为 i，合并的两个点为 u, v。u 点和 v 点分别在第 $[1, h_u]$ 和第 $[1, h_v]$ 个整周期处存在袋鼠，故这次合并在周期 $[1, \min(h_u, h_v)]$ 中的每个周期的第 i 个时刻都会产生合并，让答案 -1。
- 将两个点 u, v 合并之后，新的格子的 h 值为 h_u 和 h_v 中的较大值。

A. Hey, Have You Seen My Kangaroo?

- 求出所有合并之后，做前缀和即可求出每个时刻剩余的有袋鼠的格子数量。
- 时间复杂度 $\mathcal{O}(nmk)$，空间复杂度 $\mathcal{O}(nm)$。
- 由于采用了特殊的空间限制，无法开下一个 $\mathcal{O}(nmk)$ 大小的整形数组，这是为了尽量避免寻址常数过大导致时间复杂度正确的算法运行超时。

H. Border Jump 2

题意

- 多组询问，每次给定一个字符串，每次操作可以删除开头或者结尾的一个字符。或者选择一个前缀，使得这个前缀的反串是当前串的后缀，将当前串修改为这个前缀，并将分数 $+1$。求可能获得的最高分数。
- $|s| \leq 10^5$，$\sum |s| \leq 2 \times 10^5$。

H. Border Jump 2

引理一

- 如果 s 为回文串，则答案至少为 $\left\lceil \frac{|s|}{2} \right\rceil - 1$。
- 如果 s 不为回文串，则对其进行“选择前缀”操作得到的串长度不超过 $\left\lceil \frac{|s|}{2} \right\rceil - 1$。

证明

- 对于回文串 s，可以选择长度为 $|s| - 1$ 的前缀，让分数 $+1$，并删除开头的字符得到一个长度为 $|s| - 2$ 的回文串。
- 故由归纳可以证明引理上半部分。
- 对于非回文串 s，若其前缀 t 的反串是后缀，且两个串出现部分有重合，则可以推出 s 的反串和 s 本身相等，与 s 非回文矛盾，故引理下半部分得证。

引理二

- 对于每次执行“选择前缀”操作时的字符串组成的序列 $t_1, t_2, \ldots, t_k$，$|t_1| > |t_2| > \cdots > |t_k|$。至少存在一种使得答案最优的操作方案，使得存在一个 $0 \le p \le k$，满足 $t_1, t_2, \ldots, t_p$ 是非回文串，$t_{p+1}, t_{p+2}, \ldots, t_k$ 是回文串。

证明

- 考虑最小的 p 使得 t_p 是回文串。如果这样的 p 不存在则引理已经成立。
- 若存在 $p' > p$ 使得 $t_{p'}$ 不是回文串，则由引理一，对 $t_{p'}$ 进行“选择前缀”操作后，可能得到的后续分数不超过得到串的长度 -1，也就是不超过 $\left\lceil \frac{|t_{p'}|}{2} \right\rceil - 2$。算上这次“选择前缀”操作，和从 $t_{p'-1}$ 到 $t_{p'}$ 的操作，不超过 $\left\lceil \frac{|t_{p'}|}{2} \right\rceil$。但是 $t_{p'-1}$ 是回文串，故由引理一，可以达到的分数至少为 $\left\lceil \frac{|t_{p'-1}|}{2} \right\rceil - 1$。由于 $|t_{p'-1}| > |t_{p'}|$，故保持一直是回文串的得分至少为当前状况得分 -1。

证明

- 然后证明不能全部取到等号。
- 全部取到等号至少需要 $t_{p'-1}$ 的长度为 $t_{p'}$ 的长度 $+1$ 且为偶数。且 $t_{p'}$ 有一个长度为 $\frac{t_{p'}-1}{2}$ 的前缀，其反串为一个后缀，故可以推出 $t_{p'-1}$ 所有字符均相等，此时 $t_{p'}$ 也是回文串，导出矛盾。
- 故引理成立。

H. Border Jump 2

- 对于第一个回文串 t_p，可以证明 t_p 是某个左端点开始的最长的回文前缀，且存在一种最优操作序列使得 $t_1, t_2, \ldots, t_p$ 的左端点相同，证明暂略。
- 于是可以枚举左端点，从最长回文前缀开始，不断每次找到其反串的下一个出现位置，直到反串不存在。由于每次扩展长度至少变大一倍，所以至多扩展 $\log |s|$ 次。
- 用线段树或者主席树，配合后缀自动机维护出现位置即可。
- 时间复杂度 $\mathcal{O}(|s| \log^2 |s|)$，空间复杂度 $\mathcal{O}(|s|)$ 或 $\mathcal{O}(|s| \log |s|)$。

D. Toe-Tac-Tics

题意

- 给定 n 个 3×3 的井字棋棋盘。初始每个棋盘上有若干个 x 和 o。Alice 和 Bob 轮流进行操作，Alice 每次在棋盘上放入一个 x，Bob 每次在棋盘上放入一个 o。进行操作后，如果存在一个棋盘上一行/一列/一条对角线上有三个相同的字符，则进行这次操作的玩家输掉游戏。在棋盘被填满之后，不能进行任何操作的玩家输掉游戏。
- $n \leq 10^5$。

- 单个棋盘的游戏可以通过记忆化搜索判断胜负，但是现在有很多个棋盘，我们需要考虑所有棋盘的“和”。对于单个棋盘，如果能计算出每个局面下 Alice 比 Bob “多走几步”，整个游戏的胜负会由所有棋盘的这个值之“和”决定。我们称这个值为游戏 G 的值，记为 $G = \{G^L | G^R\}$，这里 G^L 和 G^R 分别表示左边 (Alice) 和右边 (Bob) 操作后的游戏的集合，并且我们不再区分游戏本身和游戏的值。

D. Toe-Tac-Tics

- 对于游戏 $G = \{L|R\}$，如果 L 和 R 都是数且 $L < R$，那么 G 为区间 (L, R) 内“最简单的数”。具体来说，如果区间内有整数，那么“最简单的数”为最接近 0 的整数，否则为区间内分母是 2 的正整数幂且幂次最小的分数，不难发现这个数是唯一的。
- 考虑游戏 $G = \{G^L|G^R\}$ 的化简，对于 $G_1, G_2 \in G^L$，如果 $G_1 \leq G_2$，也就是 G_1 对左边不优于 G_2，就可以直接在 G_L 中去掉 G_1，这样如果 G^L 非空且都是数，化简后就只剩下一个数 L，类似地对 G_R 有一个数 R。特别地，$G^L = \varnothing$ 可以认为 $L = -\infty$，$G^R = \varnothing$ 可以认为 $R = +\infty$。此时如果有 $L < R$ 就可以利用上述规则计算 G 的值，否则 G 不能表示为数。

D. Toe-Tac-Tics

- 现在我们可以处理游戏的值都是数的情形，但是初始全空的棋盘是一个先手必胜的局面，不能被表示为数，我们需要对上述计算规则进行扩充。考虑一个石子的 nim 游戏 $G = \{0|0\}$，无论是 L 先操作还是 R 先操作都会变为后手必胜的局面 0，因此是一个先手必胜局面，记该局面的值为 $*$。扩展到多个石子的 nim 游戏就得到了 nimber，记为 $*n = \{*0, *1, \ldots, *(n-1)|*0, *1, \ldots, *(n-1)\}$。特别地，$*0 = \{|\} = 0$，$*1 = \{0|0\} = *$。

D. Toe-Tac-Tics

- 考虑游戏 $G = \{G^L | G^R\}$，这里 G_L 和 G_R 中的所有游戏均能被表示为 $a + *b$ 的形式，其中 a 是数，b 是非负整数。对于两个值不同的游戏 $G_1 = a_1 + *b_1$ 和 $G_2 = a_2 + *b_2$，如果 $a_1 < a_2$ 那么 $G_1 \leq G_2$，如果 $a_1 > a_2$ 那么 $G_1 \geq G_2$，否则两个游戏无法比较。在非空的 G^L 中去除所有对左边不优的游戏之后，会剩下一系列 a 相同的游戏，记这些游戏的 a 为 L_a，b 的集合为 L_b，类似地对 G_R 有 R_a 和 R_b。如果 $L_a < R_a$ 那么 G 仍然是数，并且对于双方来说，选择 b 为 0 的游戏都会比不为 0 的游戏略优一个先手，可以分析 G 的值有如下四种情况：

D. Toe-Tac-Tics

- 如果 $0 \in L_b$ 且 $0 \in R_b$，那么 G 为区间 (L_a, R_a) 内“最简单的数”；
- 如果 $0 \notin L_b$ 且 $0 \in R_b$，那么 G 为区间 $[L_a, R_a)$ 内“最简单的数”；
- 如果 $0 \in L_b$ 且 $0 \notin R_b$，那么 G 为区间 $(L_a, R_a]$ 内“最简单的数”；
- 如果 $0 \notin L_b$ 且 $0 \notin R_b$，那么 G 为区间 $[L_a, R_a]$ 内“最简单的数”。

D. Toe-Tac-Tics

- 否则 $L_a = R_a$，如果此时也有 $R_a = R_b$，那么 $G - L_a$ 是一个 nim 游戏，其值为 $*mex(R_a)$，这里 $mex(R_a)$ 是不属于 R_a 的最小非负整数，于是有 $G = L_a + *mex(R_a)$。否则 $R_a \neq R_b$，情况会变得更为复杂，但是在搜索一个棋盘的所有局面之后，可以发现所有局面的值都已经能在上述的计算规则内得到，因此不需要再继续扩展计算规则。
- 现在只需要将所有棋盘的值 $a_i + *b_i$ 求和得到 $\sum_i a_i + *(\oplus_i b_i)$，如果 $\sum_i a_i > 0$ 则左边 (这里是 Alice) 必胜，如果 $\sum_i a_i < 0$ 则右边 (这里是 Bob) 必胜，否则 $\sum_i a_i = 0$，如果 $\oplus_i b_i = 0$ 则后手 (这里是 Bob) 必胜，否则是先手 (这里是 Alice) 必胜。

L. $P \oplus Q = R$

题意

- 构造三个 $0, 1, \ldots, n-1$ 的排列 P, Q, R
- 使得 $\forall 0 \le i < n, P_i \oplus Q_i = R_i$。($\oplus$ 为按位异或操作)

L. $P \oplus Q = R$

- 首先，两边同时全部异或起来，
  $0 = \oplus_{i=0}^{n-1}(P_i \oplus Q_i) = \oplus_{i=0}^{n-1}(R_i) = \oplus_{i=0}^{n-1}(i)$。
  会发现 n 必须满足 $\bmod 4$ 余 0 或者 1 。
- $n = 4k + 1$
  当 $n = 0$ 的时候显然有唯一解。
  当 $n > 0$ 时无解，因为观察最高位的 01，异或出来的序列必定会有偶数个 1，这与目标不符。

L. $P \oplus Q = R$

- 记原问题为问题 A，这边可以规约到问题 B:
- 定义 n 的双排列为 $0, 0, 1, 1, 2, 2, ..., n-1, n-1$ 的 permutation
- 构造两个双排列 p_i, q_i ，使得对应位置异或之后 $r_i = p_i \oplus q_i$ 仍然是双排列
- (可以把 p_i 固定成 $0, 0, 1, 1, 2, 2, ..., n-1, n-1$)

下面举例说明规约流程，假设问题 B 的 $n' = 2k$ 的解如下：

P	0	0	1	1	2	2	3	3	4	4
Q	0	4	2	3	3	1	1	2	0	4
R	0	4	3	2	1	3	2	1	4	0

我们把它倍增复制一下（这样变成 $n = 4k$ ）

P	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
Q	0	0	4	4	2	2	3	3	3	3	1	1	1	1	2	2	0	0	4	4
R	0	0	4	4	3	3	2	2	1	1	3	3	2	2	1	1	4	4	0	0

第一行每个节点内部的两个数连边，这里可以发现有 0-4-0，2-3-1-2 两个环

将节点内部的两个数（形成的边）按环上的顺序排好

P	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
Q	0	0	4	4	2	2	3	3	3	3	1	1	1	1	2	2	4	4	0	0
R	0	0	4	4	3	3	2	2	1	1	3	3	2	2	1	1	0	0	4	4

将所有数都乘上四倍，然后在最后两个 bit 上操作
第一行依次放上 $[0, 1, 2, 3, 0, 1, 2, 3, \cdots]$
第二行依次放上 $[0, 2, 1, 3, 0, 2, 1, 3, \cdots]$（只不过同一组的 0 和 2 之后有机会交换，1 和 3 之后有机会交换）

P	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Q	0	2	17	19	8	10	13	15	12	14	5	7	4	6	9	11	16	18	1	3
R	0	3	19	16	12	15	11	8	4	7	15	12	8	11	7	4	0	3	19	16

由于

$(0,1) \oplus (0,2) = (0,3), (0,1) \oplus (2,0) = (2,1)$

$(2,3) \oplus (1,3) = (3,0), (2,3) \oplus (3,1) = (1,2)$

我们可以反转一些 $(0,2)$ 对或者 $(1,3)$ 对，使得第三行相同的数错开

P	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Q	0	2	17	19	8	10	13	15	12	14	7	5	6	4	11	9	18	16	3	1
R	0	3	19	16	12	15	11	8	4	7	13	14	10	9	5	6	2	1	17	18

这样就成功用问题 B 的解，构造出了对应问题 A 的解。

L. $P \oplus Q = R$

- 加强一下问题 B 变成问题 C:
- 构造两个长度为 n 的排列 p_i, q_i ， 使得 $i \oplus p_i$ 和 $i \oplus q_i$ 这 $2n$ 个数构成双排列
- (加强的限制是一组 $0, 1, \ldots, n-1$ 必须对应一组 $0, 1, \ldots, n-1$)

假设 $n = 2^l + k$，其中 $0 \le k < 2^l$，那么需要按照如下表格来进行组合：
（其中 block $2, 4, 5$ 凑出 2^l 的双排列，block $1, 3$ 凑出 k 的双排列再将结果异或上 2^l）

$0 \sim k-1$	$k \sim 2^l - 1$	$2^l \sim 2^l + k - 1$	|	$0 \sim 2^l - 1$	$2^l \sim 2^l + k - 1$
$2^l \sim 2^l + k - 1$	$k \sim 2^l - 1$	$0 \sim k-1$	|	$0 \sim 2^l - 1$	$2^l \sim 2^l + k - 1$
subproblem $C(k)$	subproblem $D(2^l, k)$ 2nd block of p	subproblem $C(k)$	|	subproblem $D(2^l, k)$ whole q	subproblem $D(2^l, k)$ 1st block of p

L. $P \oplus Q = R$

$0 \sim k-1$	$k \sim 2^l-1$	$2^l \sim 2^l+k-1$	|	$0 \sim 2^l-1$	$2^l \sim 2^l+k-1$
$2^l \sim 2^l+k-1$	$k \sim 2^l-1$	$0 \sim k-1$	|	$0 \sim 2^l-1$	$2^l \sim 2^l+k-1$
subproblem $C(k)$	subproblem $D(2^l,k)$ 2nd block of p	subproblem $C(k)$	|	subproblem $D(2^l,k)$ whole q	subproblem $D(2^l,k)$ 1st block of p

- 问题 $D(2^l, k)$ 定义为：
- 构造两个长度为 $n = 2^l$ 排列 p_i, q_i ，使得 $i \oplus p_i$ 和 $i \oplus q_i$ 这 $2n$ 个数构成双排列
- 并且 $\{p_0, p_1, \ldots, p_{k-1}\} = \{0, 1, \ldots, k-1\}$，$\{p_k, p_{k+1}, \ldots, p_{2^l-1}\} = \{k, k+1, \ldots, 2^l-1\}$
- 也就是 p 数组的位置 k 上有个挡板阻止排列的流动

构造方法：

- 当 $2^l = 1$ 时 $p = [0]$ $q = [0]$
- 当 $2^l = 2$ 时 $p = [0, 1]$ $q = [1, 0]$（挡板在任何地方都成立）
- 其余情况：
  设 $B = 2^{l-1}$ ，先构造 q 把 $i \oplus q_i$ 弄成
  $[B, B, B+1, B+1, \ldots, 2B-1, 2B-1]$（这一步简单手玩）
  从子问题 $D(B, k \bmod B)$ 得到 $[p'_{first}, p'_{second}, q']$
  假如 $k < B$ 就是 $p = [p'_{first}, p'_{second}, q' \oplus B]$
  否则就是 $p = [q, p'_{first} \oplus B, p'_{second} \oplus B]$

以 $2^l = 16, B = 8, k = 11$ 举例：

构造 q 使得 $i \oplus q_i$ 弄成 $[8, 8, 9, 9, \ldots, 15, 15]$

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Q	8	9	11	10	14	15	13	12	4	5	7	6	2	3	1	0
R	8	8	9	9	10	10	11	11	12	12	13	13	14	14	15	15

得到子问题 $D(8, 3)$ 的解（这个示例也符合所说的递归构造）：

i	0	1	2	3	4	5	6	7	i	0	1	2	3	4	5	6	7
P	1	0	2	3	6	7	5	4	Q	4	5	7	6	2	3	1	0
R	1	1	0	0	2	2	3	3	R	4	4	5	5	6	6	7	7

扩展出 $D(16, 11)$ 的解：

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P	4	5	7	6	2	3	1	0	9	8	10	11	14	15	13	12
R	4	4	5	5	6	6	7	7	1	1	0	0	2	2	3	3
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Q	8	9	11	10	14	15	13	12	4	5	7	6	2	3	1	0
R	8	8	9	9	10	10	11	11	12	12	13	13	14	14	15	15

思路脚手架：

- 手动构造 $n = 2^l$ 是比较容易的，但还是需要打表得知所有的 $n = 4k$ 都有解
- 可以发现问题 A 的 $n = 8k$ 能规约到问题 A 的 $n = 4k$
- 从 $n = 12$ 的情况可以进一步做出 $n = 3 \times 2^l$ 的情况，也能找到问题 A 归约到问题 B 的思路。
- 到这一步可以写出 $n \leq 24$ 的打表，但打表已经没有用了，需要果断放弃掉。
- 按照位运算的传统步骤一位一位做，从最低位做比较困难。需要从高往低做并加强约束使得能递归构造

Thank you!