2024 中山大学程序设计校赛（SYSUCPC2024）题目讲解

ACMM 协会

中山大学

2024-12-22

题目统计

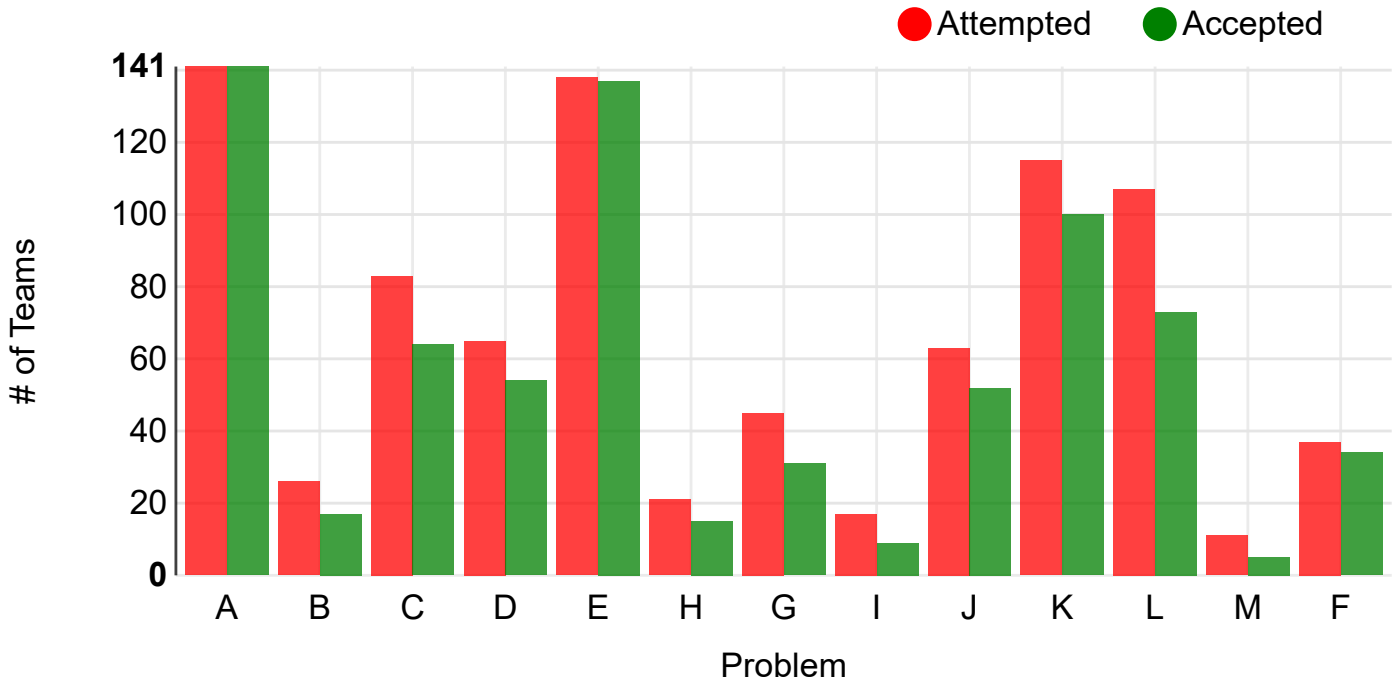

图 1　由于某些问题，这里的数据并没有按照题号排序，但是题号对应的数据是正确的。

A. Anniversary Celebration

题目简述

当前分别有 x 个字符 S，y 个字符 Y 和 z 个字符 U。每次可以将 SYSU 对应的四个字符打包为一组，求最多可以打包的组数。

题解：考虑到可以打包 k 组当且仅当 $x \geq 2k, y \geq k, z \geq k$ 即 $k \leq \min(\frac{x}{2}, y, z)$，故最终答案为：

$$\left\lfloor \min\left(\frac{x}{2}, y, z\right) \right\rfloor = \min\left(\left\lfloor \frac{x}{2} \right\rfloor, y, z\right)$$

输出这个答案即可。时间复杂度 $O(1)$。

E. Erasing Number

题目简述

初始有一个长度为 n 的 `01` 序列，Alice 和 Bob 轮流操作，Alice 可以选择删掉一个 `0` 或者同时删掉两个 `0` 以及中间的所有数字。Bob 的操作类似，但是他需要选择的数字为 `1`。如果无法删除数字的人获胜，那么在二人都足够聪明的前提下，给出最终的赢家。

题解：首先，如果整个序列不包含 `0`，或者序列的两端不全为 `0`，那么 Alice 要么无法操作，要么总能选择最左边和最右边的 `0` 并删除，使得剩余的序列只包含 `1`。此时 Bob 显然可以操作一步，而 Alice 无法继续操作，所以 Alice 获胜。

E. Erasing Number

如果序列两端全为 `0`，考虑 Alice 删完之后的序列，如果序列不包含 `1` 或者序列的两端不全为 `1`，那么 Bob 就会获胜。Alice 此时不能删除整个序列的所有数字，而在这一前提下，两端的 `0` 不会同时被删除，故删除后的序列两端必然不会全为 `1`，Bob 获胜。

综上，Alice 获胜当且仅当序列两端不全为 `0`，判断并输出即可。时间复杂度 $O(n)$。

K. King of Card Games

题目简述

对于一个长度为 m 的序列 $\{a_{1\ldots m}\}$ 使得 $1 \le a_i \le n$，定义这一序列的价值为极差的平方。求出所有 n^m 种序列的价值和。

题解：考虑求出极差为 k 的序列个数，其中 $1 \le k < n$。不妨假设序列最小值为 x，那么最大值为 $x + k$。这个序列需要满足两个条件：

- 序列的所有数字都在 x 和 $x + k$ 之间；
- x 和 $x + k$ 同时出现。

对于第二个条件，可以进行容斥，算出只符合第一个条件的情况，减去 x 和 $x + k$ 中钦定某一个数字不存在的情况，再加上 x 和 $x + k$ 都不存在的情况。

K. King of Card Games

根据简单的组合数学知识可以得到序列的个数应该为：

$$(k+1)^m - 2 \times k^m + (k-1)^m$$

而考虑到 $1 \le x, x+k \le n$ 需要同时成立，那么 x 对应的情况数为 $n-k$，最终即可得到极差为 k 的序列个数为：

$$(n-k)[(k+1)^m - 2 \times k^m + (k-1)^m]$$

枚举 k 并计算贡献即可，其中一些计算需要利用快速幂。时间复杂度 $O(n \log m)$。

C. Contest Reactions

题目简述

给出一场比赛对应的一些提交信息，在 0 号队伍通过了新的题目时，给出时间，通过的题目编号，以及排名的变化。

题解：对于每只队伍分别维护每道题目是否通过，以及每道题目对应的失败提交次数。在每支队伍通过一道新的题目时，将失败提交次数和当前时间通过公式累加到这支队伍的总罚时中。

对于排名变化，考虑到我们只会计算最多 52 次排名，故暴力枚举所有人并使用排序规则判断即可。

最终的时间复杂度为 $O(m + 26n)$。

L. LCM and GCD

题目简述

给定一个长度为 n 的序列 $\{a_i\}$ 和一个长度为 m 的序列 $\{b_j\}$，构造一个 $n \times m$ 的矩阵，使得：

- 对所有 $1 \le i \le n$，第 i 行所有数字的 lcm 为 a_i；
- 对所有 $1 \le j \le m$，第 j 列所有数字的 gcd 为 b_j。

题解：首先特判 $n = 1$ 和 $m = 1$ 的情况。对于 lcm 和 gcd，存在一个性质：一个数字 x 与其倍数的 gcd 必然等于 x，而一个数字 y 与其因数的 lcm 必然等于 y。我们据此可以导出一个较为简单的构造方式。

L. LCM and GCD

首先。第 i 行第 j 个元素既是 b_j 的倍数，也是 a_i 的因数，故 $b_j \mid a_i$ 是构造存在的一个必要条件。如果选择计算 b_j 的 lcm 进行判断，需要注意溢出问题。

其次，考虑在必要条件下进行构造。我们首先将第 j 列的所有元素设为 b_j，随后在每一行中，选择某一个元素并改为 a_i，并且保证每一行选择的位置不在同一列上即可。根据前面提到的性质：

- 对第 i 行，由于所有数字都是 a_i 的因子，而且存在 a_i，故 lcm 为 a_i；
- 对第 j 列，由于所有数字都是 b_j 的倍数，而且由于每行被选择数字不会出现在同一列上，这一列必然存在一个数字等于 b_j，故 gcd 为 b_j。

最后任意选择一个修改方案即可。时间复杂度 $O(nm + (n+m)\log V)$，其中 V 代表每个数字大小的上界，这里为 10^9。

D. Depths of Cities

题目简述

给出一棵包含 n 个点的树，对这棵树添加一条新边使其变为基环树后，将每个点的深度定义为这个点到环上任意点需要经过的最少边数。求出所有添加新边的情况下，每个点对应的深度和。

题解：首先需要注意到，一个包含 k 个点的树总共有 $\binom{k}{2}$ 个连接方案。这一点在题目中已经给出提示。接下来记 s_u 表示节点 u 的子树大小。

我们首先考虑对每个点计算其子树内的情况。记 f_u 代表环在节点 u 的子树内部时，节点 u 对应的深度和。对于节点 u，所有经过这个点的环对应的贡献都为 0（因为此时节点 u 的深度为 0），故贡献均通过子节点传递。对每个子节点 v 内部的所有环，v 对应的深度恰好为 u 对应的深度减 1，故有转移 $f_u \leftarrow f_v + \binom{s_v}{2}$。

D. Depths of Cities

接下来记 g_u 为环在节点 u 子树外部（也就是新边连接的两个点均不在 u 的子树中）时，节点 u 对应的深度和。此时的转移方向是从节点 u 的父节点 fa 转移到节点 u。类似的，对于所有可能的 $\binom{n-s_u}{2}$ 种环，fa 对应的深度恰好为 u 对应的深度减 1，故有转移 $g_u \leftarrow g_{fa} + \binom{n-s_u}{2}$。

根据树形 dp 的思想，只需要在树上进行 DFS 即可求出答案。时间复杂度 $O(n)$。

F. Finding Maxi-strings

题目简述

给出一个只包含小写字母的字符串 s，定义答案集合 S 为作为连续子串在 s 中出现次数最多的字符串集合。例如 $s =$ "ababab" 时 $S = \{$"a", "b", "ab"$\}$。你需要在一个输入框内输入所有的答案。输入框初始为空，而每次可以进行三次操作：

- 将任意一个小写字母添加到输入框内容的末尾；
- 在输入框的内容中删除最后一个字符；
- 确定当前输入框的内容为一个答案，在确认之后输入框内容不会改变。

给出确定所有答案所需的最少操作数。需要支持 m 次修改，每次修改会改动字符串 s 的某一个位置。

F. Finding Maxi-strings

题解：我们首先可以注意到，答案字符串是由一些“众字符”组成的，其中众字符代表在字符串 s 中出现次数最多的字符。我们还可以注意到另外一些性质，例如：每个答案内的字符不会出现重复（否则根据匹配规则，重复的众字符无法完全匹配），以及答案字符串的任意子串都是答案字符串。

接下来的策略可以通过简单的贪心思维得到，也可以通过 Trie 上游走的思维得到（这里不再展开）。我们对于每个众字符，找到这个众字符开头的最长答案，并假设这个长度为 k。那么接下来，我们需要输入这个字符串，并且每输入一个字符就按下一次回车。在输入完之后，我们可以直接删除这个字符串。这个流程总共需要 $3k$ 个步骤。而如果在最后一次回车后发现所有答案都已经确定，步骤数量可以减到 $2k$。

我们必然选择 k 最大的字符串在最后输入，那么答案就等于 $3 \times \sum k - \max k$。

F. Finding Maxi-strings

接下来考虑如何求出所有的 k。众字符的判定是简单的，而为了确定最长长度，我们需要获得另一个重要的性质：

- 在答案字符串中，众字符 a 可以出现在众字符 b 后，当且仅当在字符串 s 中，每一个字符 b 出现的位置都紧跟着一个字符 a。这也告诉我们：一个众字符后面要么无法跟随众字符，要么只能跟随一个固定的众字符。

据此维护字符串 s 中相邻两个字符对的出现次数，我们就可以使用 $O(26^2)$ 的复杂度确定众字符的后继，并利用 $O(26^2)$ 甚至更低的复杂度算出所有 k。

修改带来的差异是很容易维护的。最终的时间复杂度为 $O(n + 26^2 m)$。根据实际测试，常数较小的 $O(n + 26^3 m)$ 算法亦能通过。

G. General Checksum Calculation

题目简述

给定一个长度为 n 的序列 $\{a_i\}$，需要回答 m 次询问。每次询问给出 l, r, v，需要计算 $\oplus_{i=l}^{r} (a_i - v)$。

题解：对于异或和问题，不妨逐位考虑。对于答案的二进制第 k 位，某个位置 $l \le i \le r$ 会产生贡献当且仅当 $a_i - v$ 的二进制第 k 位为 1。这一条件实际上等价于：

$$\exists 2^k \le z < 2^{k+1} \text{ s.t. } a_i - v \equiv z \pmod{2^{k+1}}$$

G. General Checksum Calculation

不妨令 b_j 为满足 $a_i \bmod 2^{k+1} = j$ 的位置 i 的个数，那么上述条件对应的贡献等价于这个数列上的一个或两个区间的异或和。离散化询问后将对 $[l, r]$ 的询问拆解成 $[1, l-1]$ 的询问答案异或上 $[1, r]$ 的询问答案，随后枚举 k 并按照位置从小到大的顺序依次计算每一个询问的答案。对于 b_j 数组需要维护单点修改和区间异或和，使用线段树或者树状数组均可。

最终时间复杂度 $O(n \log^2 V)$。本题同样存在利用可持久化线段树实现的版本，复杂度相同。

J. Jazz Music from the Er-th

题目简述

给定一个长度为 n 的序列 $\{a_i\}$ 和一个数字 L，对于所有长度为 n 且满足 $L \le v_i \le a_i$ 的序列 $\{v_i\}$，求出其二进制或和的最大值。

题解：考虑对当前情况按位考虑并不断转换为子问题。不妨假设 $f(L, u, P)$ 为满足如下限制的 v 序列的二进制或和最大值：

- 对所有 $p \in P$，需要满足 $L \le v_p \le u_p$；
- 对所有 $1 \le q \le n$ 且 $q \notin P$，需要满足 $0 \le v_q \le u_q$。

定义全集 $U = \{1, 2, ..., n\}$，那么答案即为 $f(L, a, U)$。

J. Jazz Music from the Er-th

接下来计算 $f(L, u, P)$。我们从高到低枚举二进制位 k（编号 0 开始），并钦定 L 以及 u 序列的所有元素均小于 2^{k+1}。接下来考虑二进制第 k 位：

- 如果 L 的二进制第 k 位为 0：
  - 如果 u 序列中有至少两个位置 x 和 y 满足 u_x 和 u_y 的二进制位均为 1，那么构造 $v_x = 2^k, v_y = 2^k - 1$ 即可达到当前问题的理论上界 $2^{k+1} - 1$，返回即可。
  - 否则，如果 u 序列中只有一个位置 x 满足 u_x 的二进制位为 1，则为了保证二进制或和尽可能大，v_x 的二进制第 k 位也需要为 1，而此时 $L - 2^k < 0 \le v_x - 2^k \le u_x - 2^k$，那么 x 位置在子问题中不受 L 的限制。令 $u_x \leftarrow u_x - 2^k$，那么答案为 $2^k + f(L, u, P \setminus \{x\})$。
  - 否则，u 中所有元素均小于 2^k，枚举更小的二进制位 k 继续计算。

J. Jazz Music from the Er-th

- 如果 L 的二进制第 k 位为 1：
  - 如果 P 为空，此时所有数字都不受 L 限制，讨论和前面的分类完全一致；
  - 如果 P 非空，对所有 $p \in P$ 都有 $L \le v_p \le u_p$，故 u_p 和 v_p 的二进制第 k 位必然为 1，此时提前令 $u_p \leftarrow u_p - 2^k$。根据这个性质，此时问题答案的二进制第 k 位必然是 1。另外，考虑到 $0 \le L - 2^k \le v_p - 2^k \le u_p - 2^k$，$v_p - 2^k$ 仍然受到 $L - 2^k$ 的限制，即子问题的 P 不会减少元素。在这一前提下进一步讨论 $U \setminus P$ 中的位置：
    - 如果存在一个位置 $x \in U \setminus P$ 使得 u_x 的二进制第 k 位为 1，构造 $v_x = 2^k - 1$ 即可达到当前问题的理论上界 $2^{k+1} - 1$，返回即可。
    - 否则，答案为 $2^k + f\big(L - 2^k, u, P\big)$。

暴力维护每个元素是否在 P 中并按照上述方式讨论即可。时间复杂度 $O(n \log V)$。

B. Bruhcaea Simulator

题目简述

在一个二维坐标系上，有两条折线，分别按顺序连接下面的两组点：

$$(1,p_1),(3,p_2),...,(2m-1,p_m);(2,q_1),(4,q_2),...,(2m,q_m)$$

其中需要满足 p_i 和 q_i 均为不超过 n 的正整数。在两个折线不相交的前提下，定义两条折线的曲折度为 $\sum_{i=1}^{m-1}|p_i-p_{i+1}|+|q_i-q_{i+1}|$。求所有满足不交前提下的折线对应的曲折度总和。

题解：我们可以钦定第一条折线在第二条折线的下方，而只需要翻转一下就可以得到另外一半的情况。不妨称第一条折线为下折线，而第二条折线为上折线。

B. Bruhcaea Simulator

根据折线特性，对任意正整数 $a \in [2, 2m-2]$，两条折线在 $a \le x \le a+1$ 区域分别对应一条线段。我们可以得到如下推论：下折线保持在上折线的下方，当且仅当对任意正整数 $a \in [2, 2m-1]$，下折线和 $x = a$ 的交点对应的纵坐标比上折线交点小。

我们据此考虑转移形式。令状态 (a, u, v) 为当前考虑了横坐标不超过 a 部分的点，并且下折线当前纵坐标为 u，上折线当前纵坐标为 v 的状态。此时如果 a 是偶数，则下一步需要确定下折线的新坐标，否则需要确定上折线的新坐标。

我们讨论 a 为偶数的情况，另一个情况同理。此时下折线当前坐标为 $(a-1, u)$，上折线当前坐标为 (a, v)。如果要让下折线的当前坐标转移到 $(a+1, w)$，由于需要满足在 $x = a$ 处的交点的偏序，可以得到条件为 $w < 2 \times v - u$，恰好是一段前缀。

B. Bruhcaea Simulator

那么，定义 $f[a][u][v]$ 为转移到 (a,u,v) 对应的方案数，而 $g[a][u][v]$ 为转移到 (a,u,v) 对应的曲折度和，那么在 $1 \le w \le \min(n, 2 \times v - u - 1)$ 时，有转移：

$$\begin{cases} f[a+1][w][v] \leftarrow f[a][u][v] \\ g[a+1][w][v] \leftarrow g[a][u][v] + |w-u| f[a][u][v] \end{cases}$$

该转移的第一项均可以使用前缀和计算，而第二个转移的剩余项可以通过维护一阶差分数组计算，故上述转移对应的时间复杂度为 $O(n^2)$。

最后只需要用同样的思路计算出另外一部分的转移即可，不过为了方便起见，你也可以调整转移的终点（对后两个参数取反并交换），使得每次转移的式子相同。最终时间复杂度为 $O(mn^2)$。

H. Hoppers and Doors

题目简述

对于一个包含一个起始点，n 个宝箱点和 m 个被锁住的点组成的无向图中，如果每个被锁住的点对应的钥匙随机分配在 n 个宝箱点的一个，求出能够得到所有钥匙并打开所有门的钥匙放置方案数量。

题解：本题的 n 和 m 都相当小，足以设计如下状态：$f[S_1][S_2]$ 代表在打开 S_1 集合的宝箱，并拥有 S_2 集合的钥匙对应的情况数。我们考虑进行转移。根据钥匙的情况可以在图上搜索并算出所有可以到达的宝箱点，如果存在未前往的宝箱点，则可以进行转移。如果选择前往宝箱点 a，并开出了 S' 集合的新钥匙，那么可以得到转移：

H. Hoppers and Doors

$$f[S_1 \cup \{a\}][S_2 \cup S'] \leftarrow f[S_1][S_2] \times \frac{(n-|S_1|-1)^{m-|S_2|-|S'|}}{(n-|S_1|)^{m-|S_2|}}$$

右侧式子可以写为 $\left\{f[S_1][S_2] \times \left[\frac{n-|S_1|-1}{n-|S_1|}\right]^{m-|S_2|}\right\} \times (n-|S_1|-1)^{-|S'|}$，大括号内部的表达式可以提前计算，而剩余的项可以使用高维前缀和计算。具体而言，将大括号的项计算好后转移到 $\mathrm{pre}[S_1 \cup \{a\}][S_2]$ 中，然后对 pre 统一进行高维前缀和，每向高位转移就除以 $n-|S_1|$。这里少了一个 -1，这是因为转移之后 S_1 本身就多了一个元素。我们的计算流程如下：

从小到大枚举 S_1。首先计算 pre 的高维前缀和并转移到 f 上，然后枚举 S_2，并选择“恰当的”转移，将 $f[S_1][S_2]$ 转移到后续的 pre 数组中。需要注意，在 $S_1 = \emptyset$ 的时候，不应当进行 pre 计算的步骤。

H. Hoppers and Doors

前面的转移提到了“恰当的”这一前提，这是因为在不恰当的转移下，一个状态可能会重复转移到后面的状态。对于这个问题，我们可以对转移路径进行钦定，例如只选择当前可到达且编号最小的新宝箱点转移。

在提前计算每个钥匙集合对应可以到达的宝箱集合后，借助二进制处理技巧可以在 $O(m2^{n+m})$ 的时间复杂度内计算出最终答案。另外，本题的除法均可以整除，故无需介入浮点数。

上述方法足以解决本问题，不过你也可以更进一步，考虑通过“一次走完所有漏斗”的形式转移，可以发现宝箱集合可以进一步转换为宝箱的个数，转移也从高维前缀和变成子集转移。此时的时间复杂度为 $O(n3^m)$。

I. Isla Loves Christmas

题目简述

给定一个长度为 n 的颜色数组 a_i，另有一个长度为 n 且初始为 0 的数组 b_i。需要进行 m 次操作，每次操作使用 l, r, v 三个参数定义，并进行如下操作：

- 对所有满足 $l \le p \le q \le r \wedge a_p = a_q$ 的位置对 (p, q)，对满足 $p \le x \le q \wedge a_x = a_p$ 的所有位置 x，将 b_x 加上 v。

求出所有操作后 b_i 数组的每项值。

题解：操作对每种颜色都是独立的，故可以考虑对每种颜色分别计算。为了方便进行研究，不妨只考虑颜色全相同的问题，并通过重编号的形式套用到原题中。

I. Isla Loves Christmas

对每次询问分别计算每种颜色对应的询问区间是不现实的，而对区间的单次扩展和收缩只会影响一个颜色对应的问题，这启发我们考虑莫队算法。为了实现莫队算法，我们需要研究贡献变化。假设有一个长度为 $k+1$ 的区间，其内部对应颜色全部相同。我们进行一次询问 $(1,k,1)$，得到的 b 数组贡献形态如下：

$$1\times k,2\times(k-1),3\times(k-2),...,k\times 1,0$$

而将询问右端点移动到 $k+1$，对应的形态应当为

$$1\times(k+1),2\times k,3\times(k-1),...,k\times 2,k+1$$

差分后得到

$$1,2,3,...,k,k+1$$

I. Isla Loves Christmas

我们可以发现，在颜色相同的子问题中，移动区间的一个端点带来的贡献变化是一个等差数列。这启发我们在这个子问题中维护答案的一阶差分，随后就可以实现 $O(1)$ 更改贡献。

不过更改贡献值并不足以完成整个问题，因为我们还需要时刻将 v 值乘以贡献加到一阶差分数组中。对于这个问题，考虑在莫队排序结束后，将 v 参数重新赋值为后缀和，那么加入和撤销某次贡献的时机对应的贡献落差恰好对应了这段时间的 v 数组和。例如，某次贡献在 i 时刻加入，而在 j 时刻移除，那么对应的贡献为 $v_i + v_{i+1} + \ldots + v_{j-1} = \mathrm{sufv}_i - \mathrm{sufv}_j$，恰好符合贡献对应的落差。

最终的时间复杂度为 $O(n\sqrt{n} + m \log m)$。验题人给出了一种基于根号分治的算法，可以做到 $O(n\sqrt{n \log n})$，这个算法在常数较小的前提下亦可以通过。

M. Make SYSU Great Again 3

题目简述

给定一个整数 n $(n \geq 4)$，构造一个以圆环形式放置的排列 p，使得在这个圆环上取相邻三个数字时，在 n 个方案中的 $\lceil \frac{n}{2} \rceil$ 个方案满足三个数字的最大值等于剩余两个值的和。

例如，对于排列 $(1, 2, 3, 4, 5)$，五种方案为 $(1, 2, 3)$、$(2, 3, 4)$、$(3, 4, 5)$、$(4, 5, 1)$ 和 $(5, 1, 2)$，其中第一个和第四个方案满足要求。

M. Make SYSU Great Again 3

题解：不妨先脱离圆环问题，在序列上进行构造。一种使得满足要求的方案接近一半的方法如下：

- 如果 n 是奇数，取最大的 $\lceil \frac{n}{2} \rceil$ 个值，并按照“最小 $\rightarrow$ 最大 $\rightarrow$ 次小 $\rightarrow$ 次大 $\rightarrow$ ...” 的顺序排列，而其他 $\lfloor \frac{n}{2} \rfloor$ 个值插在中间。例如在 $n = 9$ 时，构造为 $\underline{5}, 4, \underline{9}, 3, \underline{6}, 2, \underline{8}, 1, \underline{7}$。
- 如果 n 是偶数，取最大的 $\frac{n}{2}$ 个值按类似顺序插入，而中间只能插入 1 到 $\frac{n}{2} - 1$ 的数字。例如在 $n = 8$ 时，构造为 $\underline{5}, 3, \underline{8}, 2, \underline{6}, 1, \underline{7}$，而 4 可以放在头和尾。

上述构造会提供恰好 $\lceil \frac{n}{2} \rceil - 1$ 个符合要求的方案，而最后一个方案需要通过调整得到。

M. Make SYSU Great Again 3

- 在 n 是奇数时，只需要将 1 向右移动一位，即可和序列的前两个数字构成新的方案。例如在 $n = 9$ 时，对应构造 $5, 4, 9, 3, 6, 2, 8, \underline{7}, \underline{1}$，此时最后三个数字原先形成的方案 $(8, 1, 7)$ 没有失效，而新增一个 $(1, 4, 5)$ 的方案。
- 在 n 是偶数时，同理将 $\frac{n}{2}$ 放在头部，并且将 1 向右移动一位，二者即可以和原先序列的第一个元素形成一个新的方案。例如在 $n = 8$ 时，对应构造 $\underline{4}, 5, 3, 8, 2, 6, \underline{7}, \underline{1}$。之前形成的方案 $(6, 1, 7)$ 没有失效，而新增一个 $(1, 4, 5)$ 的方案。

我们就得到了一个完备的构造。时间复杂度 $O(n)$。

致谢名单

感谢校外人士 wjy666 在出题工作和验题人招募上给予的帮助。

感谢校外人士 dXqwq，installb，tarjen，Nanani，DF_3Mz 和 SSerxhs 参与比赛的测试并提出宝贵的修改意见。

感谢 MikeMirzayanov 创立的 Polygon 平台，使出题流程变得更为流畅。

感谢 ACMM 协会的全体工作人员，以及辅助解决本次比赛各类问题的前辈。