交互、通信选讲

清华大学 刘海峰

February 7, 2026

讲题过程中可能讲题人无法严谨证明一些结论（或者一些题的做法不够优秀），如果大家更好的想法，欢迎和讲题人交流分享。

非特殊说明下我们不妨认为交互库是自适应的（尽管有时候自适应几乎无法实现），换而言之，在无特殊说明时，我们统一考虑最坏情况。

非特殊说明下序列下标从 1 开始。

传统的 OI 题都是给定输入，然后通过你写的代码计算得到输出。

而在交互题中，选手程序需要在运行过程中与评测器反复交互：根据评测器的实时反馈不断提出查询、调整策略，最终确定答案。其核心在于决策过程与查询策略设计，通常要求在有限交互次数内定位隐藏信息或完成判断。

当然，也有些交互题是为了限制某些量、函数的调用次数从而出成交互的形式。

这种题的评测方式也有两种，第一种是函数式交互，题目会准备好相应的头文件以及交互细节、编译方式，然后你需要按照要求引用头文件、调用函数从而达到交互效果，这种类型一般禁止访问 IO。

第二种类型是 IO 式交互，一般来说就是你可以通过访问 stdin 和 stdout 和交互库进行交互。注意每次输出之后需要 flush 一下来更新缓冲区。

IOI 2025 Practice

你需要猜一个数 x，已知 $1 \le x \le 10^{18}$，此外存在一个你不知道的阈值 K。

你每次可以询问交互库一个数 y，若当前询问次数 $\le K$，则交互库会返回 $x \le y$ 是否成立，否则会返回 $x > y$ 是否成立。

询问次数限制：67 次。

注意到我们一定可以在 $2\log V$ 次询问内得到答案，考虑将状态看成二元组 (x, K)，表示结果为 x 且阈值为 K 的情况。

每次询问我们可以通过适当手法找到一个 x 使得询问之后能分成尽量均匀的两份状态。

这样询问次数就是 $\log V + \log\log V + O(1)$。

AGC044D

交互库有一个长度不超过 L 的字符串 S，其字符集为所有大小写字母加数字。

你每次可以询问交互库一个长度不超过 L 的字符串 T，交互库会返回 S, T 的编辑距离。

定义两个字符串的编辑距离为：可以对第一个字符串做插入字符、删除字符以及修改字符使得其变成第二个字符串所需的最小操作数。

$L = 128$，询问次数限制：850。

编辑距离是很难刻画的，考虑通过编辑距离能获得什么信息。

首先如果 S, T 的编辑距离为 $|S| - |T|$，那说明 T 一定是 S 的子序列。

编辑距离是很难刻画的，考虑通过编辑距离能获得什么信息。

首先如果 S, T 的编辑距离为 $|S| - |T|$，那说明 T 一定是 S 的子序列。

同时注意到我们可以询问 L 个相同的字符组成的字符串得出这个字符的出现次数。

然后分治归并排序即可。

IOI2018 Highway Tolls

给定一张 n 个点 m 条边的无向连通图，交互库隐藏了 S, T 两个点，同时给定两个正整数 $A, B(A < B)$，你每次可以将一些边的权值标为 A、剩余的边权值标为 B，然后交互库会返回 S, T 之间的最短路。

你需要通过不超过 50 次询问确定 S, T（不关注顺序）。

$2 \le n \le 9 \times 10^4, 1 \le m \le 1.3 \times 10^5$。

首先我们可以问出 S, T 之间的最短距离。

同时，我们可以二分一个边集的前缀，满足删掉这个前缀之后 S, T 距离增加，那么前缀中最后一条边一定在最短路上，并且两个端点分别对应 S, T 两部分的连通块。

首先我们可以问出 S, T 之间的最短距离。

同时，我们可以二分一个边集的前缀，满足删掉这个前缀之后 S, T 距离增加，那么前缀中最后一条边一定在最短路上，并且两个端点分别对应 S, T 两部分的连通块。

这个时候我们可以求出一组 BFS 序然后二分即可。

复杂度为 $\log m + 2\log n + O(1)$。看起来可能超了一点，但其实 $2\log n$ 是卡不满的，因为显然最后 n 个点被分成了两个集合。

CF1764G3

交互库有一个长度为 $n(3 \le n \le 1024)$ 的 1 到 n 的排列 p，你可以做至多 M 次询问，每次询问给定 (l, r, k)，交互库会返回 $[\lfloor \frac{p_l}{k} \rfloor, \lfloor \frac{p_{l+1}}{k} \rfloor, \cdots, \lfloor \frac{p_r}{k} \rfloor]$ 中有多少个不同的数。

然后你需要确定一个 i 满足 $p_i = 1$。

满分限制：$M = 20$。

部分分：$M = 21, M = 30$。

注意到当询问 $k=2$ 的时候，大多数数 x 都存在 y 满足 $1 \le y \le n, x \ne y, \lfloor x/y \rfloor = \lfloor y/2 \rfloor$。

但 1 不存在，n 为偶数的时候 n 也不存在。

于是这给我们提供了解题思路。

考虑如果依次询问 $(1, x, 2), (x+1, n, 2)$，那么我们可以得出左边没有被匹配的数的数量和右边没有被匹配的数的数量。

如果如果左边不等于右边则区分了 1 可能所在的位置。

如果等于说明 1 和 n 一个在左边，一个在右边。

考虑判断 n 在哪边，其实很简单，只要全部除以 n，然后看不同的数的数量即可。

于是可以直接二分，这样算一下大概是 $2\log n + 1 = 21$ 次。

考虑如何优化掉最后一次，不妨假设 n 一定不在 $[l, r]$ 内，既然能确定到这个区间，那么显然我们已经询问过 $(1, l-1, 2)$ 和 $(1, r, 2)$ 了。

那么根据结果，我们能够判断除了 1 之外的那个数是和 $[1, l-1]$ 中的数匹配还是和 $[r+1, n]$ 中的数匹配。

于是就可以只一次询问了，于是就省掉了一次。

IOI2025 纪念品

有 n 种纪念品（下标从 0 开始），第 i 种纪念品的价格为 P_i，供应量为正无穷。保证 $P_0 > P_1 > \cdots > P_{n-1} > 0$。你现在只知道 n, P_0。

每次你可以给定一个正整数 x，然后交互库将执行以下流程：

- 对于 $i = 0 \sim n-1$，如果 $x \geq P_i$，则购买一次物品 i，然后将 x 减去 P_i。

要求每次操作至少购买一件商品。

之后，交互库会返回你购买了哪些物品，以及剩余了多少钱。

你需要调用至多 5000 次购买函数，使得第 i 种物品在所有调用中被购买了恰好 i 次（这意味着第 0 种物品不能够被购买）

$2 \leq n \leq 100$。

直观上来看，第一次我们肯定要询问 $P_0 - 1$。

假如我们现在询问出某 m 个商品的价值和为 k，那么接下来询问 $\lfloor \frac{k-1}{m} \rfloor$，可以发现，除非这次只够买到了第 n 件商品，否则接下来显然能购买至少一件商品，且购买到商品的最小编号肯定会变大。于是按照该做法我们可以获取第 n 个商品的价值。然后我们可以把所有询问出的结果减掉第 n 个商品的价值。

于是可以这么做，设 B_i 表示某次购买到的一组商品集合，满足编号最小的商品编号为 i，如果目前没询问到相应的组合，那么 B_i 为空。

第一次询问 $P_0 - 1$，然后接下来每次找到最大的非空的 B_i 并按照刚才的方式进行询问，稍微处理一下即可，最后达到的效果是通过不超过 i 次购买得出了商品 i 的价格。

QOJ 10182

给定一棵 n 个点的有根树，树上每个点都有一个正整数权值（但你不知道），保证所有节点的权值都小于等于 n 且互不相同，保证父亲节点的权值小于儿子节点的标号。

你现在需要对交互库做若干次询问，每次给定两个点的编号，交互库会返回这两个点的权值哪个更大。

你需要通过至多 $1.4\lfloor\log_2 P\rfloor$ 次操作确定答案，其中 P 为这棵树的所有可能的赋权方案数。

Finger search

我们知道交互题对交互次数的限制是比较严的，这点在普通的 OI 题中难以通过时间限制实现区分，所以需要格外小心。

假设交互库有一个长度为 n 的 01 序列，里面有恰好 m 个位置是 1。

你每次可以询问交互库一个区间，交互库会返回这个区间中是否有 1（或返回 1 的个数），我们希望用尽量少的次数确定所有 1 的位置。

如果直接二分下一个位置，复杂度是 $O(m \log n)$ 的，如果考虑直接扫一遍，复杂度是 $O(n)$ 的。都不太优美。

Finger search

我们知道交互题对交互次数的限制是比较严的，这点在普通的 OI 题中难以通过时间限制实现区分，所以需要格外小心。

假设交互库有一个长度为 n 的 01 序列，里面有恰好 m 个位置是 1。

你每次可以询问交互库一个区间，交互库会返回这个区间中是否有 1（或返回 1 的个数），我们希望用尽量少的次数确定所有 1 的位置。

如果直接二分下一个位置，复杂度是 $O(m \log n)$ 的，如果考虑直接扫一遍，复杂度是 $O(n)$ 的。都不太优美。

考虑从前往后一个一个找的过程中，如果当前位置和下一个位置的差为 d，我们希望有一个 $O(\log d)$ 的算法。

根据 Jensen 不等式，在所有点之间的距离都是 $\frac{n}{m}$ 的时候复杂度是最劣的，此时是 $O(m \log \frac{n}{m})$。

比如说我们找到一个位置之后可以倍增寻找下一个位置，枚举一下倍增上界。

还有常数更小的处理方法：我们每次先询问一下接下来 n/m 个位置中是否有 1，如果没有直接跳过，否则就递归进去做。不过在不知道 m 的情况下最好还是写枚举倍增上界的做法。

当然还有另外一种处理方法，就是考虑类似线段树一样递归去计算。

不过基本上表现都不如该算法。

考虑估计 $\log \binom{n+m}{m}$ 的大小，其中 $n \geq m$。

注意到 $\log n! = n \log n + O(n)$，所以最终结果大致为 $O((n+m)\log \frac{n}{m})$，与 Finger search 的复杂度差不多。

所以对于本题，直接自底向上维护当前权值的相对大小即可。使用 Finger search 进行合并即可。

UNR6 神隐

交互库有一个 n 个节点的树，树的边是有顺序的。你每次可以询问一个长度为 $n-1$ 的 01 串，然后如果第 i 个位置是 1 则交互库会保留树的第 i 条边，否则会删掉第 i 条边，然后交互库会返回当前情况下形成的连通块集合。你需要通过不超过 $limit$ 次询问确定这棵树。

满分：$n = 2^{17}, limit = 20$。

部分分：$n = 2000, limit = 14/22$ 以及 $n = 2^{17}, limit = 34$。

首先不难得出一种询问次数 $2\log n$ 的做法，对于 $i = 1 \sim \log n$，在第 i 次询问中，我们询问边的编号二进制下第 i 位为 1 的边，在第 $i + \log n$ 次询问中，我们询问边的标号二进制下第 i 位为 0 的边。

那么如果两个点 x, y 在恰好 $\log n$ 次中都处于一个连通块内，那么 x, y 之间存在连边。

证明是显然的。

接下来分成了两个部分，一是如何卡交互次数，二是如何优化复杂度。

先看怎么卡交互次数，注意到我们上述做法正确的原因是如果 x, y 之间的最短路径大于 1，那么 x, y 上的边的编号交起来的 popcount 不等于并起来的 popcount。

于是可以考虑构造若干个大小为 $[1, limit] \cap Z$ 的子集，满足所有集合大小相等且互不相同，那么只要有超过一个那么交起来的大小一定会改变。

注意到最多能构造 $\binom{20}{10} = 184756$ 个，所以可以接受。

接下来考虑如何优化复杂度。

考虑剥叶子，问题变成了如何判断一个点是不是叶子。

做法是简单的，考虑如果一个点在 10 个询问中都处于单点的状态，那么这个点就是叶子，然后我们就可以不断删叶子，从而得到树的一个拓扑序。

然后考虑如何用这个拓扑序来还原这棵树。

注意对于一个连通块，拓扑序最大的点一定是其余的点的祖先，而对于每个点，一定存在一个连通块满足该连通块中拓扑序最大的点是该点的父亲节点，所以直接做即可。

CF2164G

交互库有一个 n 个点的无根树，你每次可以询问交互库一个排列 p，交互库会返回一个长度为 n 的序列 a，其中 a_i 表示有多少对 (x, y) 满足 $1 \le x < y \le i$ 且 p_x, p_y 有边相连。

$3 \le n \le 5 \times 10^4$，要求询问次数不超过 31 次。

思考一下我们能通过这个问出些什么。

首先对得到的 a 序列做差分，我们可以问出 $p_{1\cdots i-1}$ 中有哪些点和 p_i 有边相连。

于是我们问一下 p，再问一下 $reverse(p)$，我们就可以得到树上每个节点的度数。

当然只有这个还不能做这题。

我们先考虑我们可以通过什么信息能确定一棵树。

一种想法是确定每条边，不过放到这题里很不现实。

实际上我们也可以剥叶子。

具体而言，我们可以维护每个点的度数以及每个点的所有邻居的编号和，然后每次找一个度数为 1 的点剥掉即可。

考虑如何求和一个点有连边的点的编号之和，一种显然的方法就是倍增，数二进制下每一位出现了多少个。

具体来说，枚举 i，然后把二进制下第 i 位为 1 的排成一个序列 A，为 0 的排成一个序列 B。

第一次询问交互库 AB，第二次询问交互库 BA，然后减一下就可以得到答案。

复杂度是 $2\log n+1$，因为问度数可以随便找一个然后 reverse 一下。

但交互次数多了一点。

这里有一个比较经典的 trick，就是 $f(x) = \frac{x}{\ln x}$ 在 $(1, +\infty)$ 上最小值在 $x = e$ 处取到，并且有 $f(2) > f(3)$。

也就是说这种情况下采取三进制其实是更优的。

用三进制复杂度是 $3\log_3 n + 1$ 次，可以通过。

IOI2020 mushroom

有 $n(n \leq 2 \times 10^4)$ 个蘑菇，每个蘑菇有 A,B 两种类型，保证第一个蘑菇是 A 类型。

现在有一台机器，每次可以将一些蘑菇排成一个序列放进去，机器会返回一个数 x，表示有多少对相邻的蘑菇类型不同，例如放入 ABBA 会返回 2。

你现在想知道 A 类型的蘑菇总共有多少个。你需要保证机器调用次数不超过 226 次，且放入机器的蘑菇总数不超过 10^5。

假如我们现在已经有一定量的 A,B 类型的蘑菇了。

假设我们当前有 x 个 A 类型的蘑菇，y 个 B 类型的蘑菇，不妨假设 $x \geq y$。

那么我们可以 $Ap_1 Ap_2 Ap_3, \cdots, Ap_x$ 这样去询问，可以得出 $p_{1 \cdots x-1}$ 中有多少个 A 类型的蘑菇，以及通过奇偶性来判断 p_x 的类型。

这样我们就可以做到 $O(\sqrt{n})$ 次调用了，虽然距离目标还是有点远。

当 A, B 数量比较少的时候，一次询问获取一个蘑菇类型性价比不如获取两个蘑菇类型，于是可以选择问 $ApAq$，这样就可以知道 p, q 两个蘑菇的类型了。

设一个阈值 B，询问次数小于等于 B 时采取第一种策略，大于 B 时采取第二种策略。

这样可以获得 92 分左右的高分。

询问次数小于等于 B 的部分有没有更好的做法。

首先每次询问通过奇偶性得到的那一个几乎没有优化的可能。考虑前面的部分能不能优化。

考虑第一次问 $AaAbA$，如果 a, b 两个蘑菇类型相等，那么就可以直接确定了。否则相当于是 a, b 两个蘑菇类型不同。

那么接下来问 $AaABbB$ 结果只有可能是 1 或 5。于是这里就有操作的空间了。

下一次直接询问 $AaABbBcB$，那么就可以获得 a, b, c 的具体类型。

于是这样相当于实现了两个问三个。

应用于本题再精细卡常可以获得满分。

实际上这个 2 问 3 有大量的优化空间。

其实这个问题本质上就是说，交互库有 n 个 01 变量，你每次给定 n 个 $\pm 1, 0$ 的权值，交互库会返回对应的 01 变量乘这一个值得到的答案。

实际上这个问题询问次数可以做到 $O(\frac{n}{\log n})$。

具体而言，假设我们现在存在一个 $a \times b$ 的矩阵 X，其中 $X_{i,j} \in \{0, \pm 1\}$，满足按照这个矩阵询问可以获取前 b 个 01 变量的具体值。

那么考虑每次迭代令

$$X \to \begin{pmatrix} X & -X & I_a \\ X & X & 0 \end{pmatrix}, b \to 2b + a, a \to 2a$$

首先我们把第 i 行和第 $i + a$ 行相加通过奇偶性可以确定最后 a 个数，然后确定完之后就可以直接做了。

询问次数大致为 $O(\frac{n}{\log n})$。

实际上还可以进一步拓展，考虑每次我们询问给定的权值只有可能是 01 的情况。

首先可以将 X 拆成两个矩阵相减，这样就转化成了第一个问题，但常数翻了一倍。

考虑令 Y 表示该解法的询问矩阵，设其规模为 $a \times c$。

那么每次迭代令

$$Y \to \begin{pmatrix} Y & X_1 \\ Y & X_2 \end{pmatrix}, c \to c + b$$

其中 X_1, X_2 是 $a \times b$ 规模的 01 矩阵，满足 $X_1 - X_2 = X$。

这样就可以获得一个常数更小的做法。

上述内容参考自 https://www.luogu.com/article/1dzympve

通信题的题目会将算法拆分为多个相互通信的角色，每个进程只能看到局部信息，只能通过受限的通信协议交换消息完成整体任务。

其核心在于信息分配与通信复杂度控制，强调如何用尽量少、尽量高效的信息传递达成全局最优。

通信题目前国内 OI 基本没有出现过，不过 JOI 每年都会有，感兴趣的选手可以去刷一下近些年 JOI 的通信题。

通信题的评测一般也分两种。

一种是像函数式交互那样，注意到扮演的两个角色直接只能通过指定的方式通信，所以你需要写两个程序。然后像交互题那样通过指定的函数接口进行操作。

还有一种是出成 Run twice 的类型。第一次运行你需要将第一个角色发送的东西输出，然后第二次运行你将接收到第一次运行所发送的内容，然后再进行解密。

（当然好像也有极少数题是 IO 通信的）

IOI 2025 Practice

有一个长度为 256 的排列，Alice 需要将其中至少 16 个数变成 -1，然后 Alice 会把变完之后的序列发给 Bob，然后 Bob 需要复原这个排列。

你需要安排 Alice 和 Bob 的策略使得最后排列一定能被复原。

首先注意到一个排列一定满足 $LIS \times LDS \geq n$，所以一定存在一个长度大于等于 16 的上升或者下降子序列。所以一种想法是干掉一个上升或下降的子序列。

于是问题变成了如何区分 LIS 和 LDS，于是就会出现大量的做法。

首先注意到一个排列一定满足 $LIS \times LDS \geq n$，所以一定存在一个长度大于等于 16 的上升或者下降子序列。所以一种想法是干掉一个上升或下降的子序列。

于是问题变成了如何区分 LIS 和 LDS，于是就会出现大量的做法。

这里给一个比较简单的做法。

如果 LIS 小于 16，那么 LDS 一定大于等于 17。所以如果选定的是 LIS 则选一个长度为 16 的，否则选一个长度为 17 的即可进行区分。

ICPC2025 沈阳站 C

给定 $n, m(n \le m)$，保证 $1 \le n \times m \le 10^6$。

第一次运行你将会收到 n 个互不相同的 $1 \sim m$ 中的数，你需要对每个数构造一个到长度为 $3m$ 且 $1 \sim m$ 恰好出现三次的序列。

随后，交互库会对于每个序列，构造一个 $\{1, 2, \cdots, m\} \to \{0, 1\}$ 的映射并能将这个序列中的每个数作用于这个映射，保证作用完之后序列不是全 0 或者全 1 序列。

第二次运行你将会收到 n, m 以及这 n 个 01 序列，你需要确定这些 01 序列对应的原来的数是多少。

首先如果前面放 $1, 2, \cdots, m$，那么我们就可以确定哪些映射到了 0、哪些映射到了 1。

设 p 为小于等于 n 的最大的质数。
考虑 $x \leq p$ 怎么构造。

设 p 为小于等于 n 的最大的质数。

考虑 $x \le p$ 怎么构造。

构造两个长度为 p 的序列，第一个是 $1 \sim p$，第二个是 $n \sim n-p+1$，然后对第 i 组构造再对序列做 i 次循环位移。

注意对于任意长度为质数的非全相等的字符串，其任意循环位移都不等于原串。于是我们可以确定循环位移的长度。

然后我们再把没填够次数的数依次补充到最后面。

考虑剩余的部分怎么构造。

考虑剩余的部分怎么构造。

做法是简单的，有很多解决方法，这里提供一种我的构造方法。

先按照 $\bmod p$ 下的方法进行构造，然后对前 n 个位置做一个右移 1 的循环位移即可。

JOIST 2024 Spy 3

有一个 n 个点（编号从 0 开始）m 条边的无向图，每条边都有一个正整数权值。

Alice 有这张图的完整信息，而 Bob 手里的图中有 k 条边的权值缺失了（剩余的信息 Bob 也全知道），Alice 知道 Bob 手里缺失哪些边的信息。

Bob 现在想要知道 0 到 $T_1, T_2, \cdots, T_Q$ 的最短路（只需要找出一条最短路即可，不需要求出权值），并且 Alice 也知道 $Q, T_{1\cdots Q}$。

Alice 需要给 Bob 传输尽量少的比特，使得 Bob 能成功找出相应的最短路。

$2 \le n \le 10^4, 1 \le m \le 2 \times 10^4, 1 \le Q \le 16, 1 \le K \le 300$，边权 $\le 10^{12}$。

满分限制：1560 bits。

首先最粗暴的想法是把这 k 条边的权值传过去，然后 Bob 自己找最短路。显然是没前途的。

首先最粗暴的想法是把这 k 条边的权值传过去，然后 Bob 自己找最短路。显然是没前途的。

一个聪明点的想法是如果一条边在 $0 \to T_i$ 的最短路中，那么可以把这条边的权值设置为 0，否则设置为 $+\infty$，这样只需要传 $KQ = 4800$ bits 即可。

接下来是两种优化方式。

注意到最短路会构成一个树形结构，我们只需要把树的信息想办法发送过去即可。

注意到最短路会构成一个树形结构，我们只需要把树的信息想办法发送过去即可。

具体来说，我们先把树可能的形态压缩成 01 串并发送，然后再发送每个点属于树上哪一条边。

这样就是 $1500 + K$ bits，其中 K 为把树传过去所需要的 bits。

处理优秀的话即可通过。

刚才的方法是 Bob 在不借助任何已知信息的情况下直接收到 Alice 的树的做法。

但实际上 Alice 不需要把整棵树的完整信息，因为 Bob 这里也有图上剩余的信息。

就是直接像建树那样来做。先处理出 $0 \to T_1$ 的树上有哪些边，然后再插入 $0 \to T_2$，这两个有一些公共边，然后还有一些独属于 $0 \to T_2$ 的部分，然后再插入 $0 \to T_3 \cdots$ 以此类推。

对于每一条信息丢失的边，我们发送第一次加入是在什么时候，发送这个需要 $K \log(Q+1)$ bits。

然后我们还要处理出每条链最后会拼接到什么位置，这里需要 $(Q-1)\log(K+1)$ bits。

所以总共大概需要 1350 bits，可以通过。

IOI2025 恐龙大迁徙

有一个 n 个点的有根树，保证根节点为 1 且所有点的父亲编号都小于该节点的编号。

第一次运行时你将获取 n，然后依次收到每个节点的父亲，在每收到一次之后需要发送一个 $0 \sim Z$ 之间的数。你需要保证发送的非 0 数的数量不超过 M。

第二次运行的时候你将会获取 n 和你发送的数的序列，你需要确定这棵树的一对直径的端点（即找到一组 x, y 使得 x, y 的树上距离最大）。

满分限制：$Z = 4, M = 8$。

部分分：$Z = 5, M = 50$，$Z = 4, M = 30$，$Z = 4, M = 11$。

如果当前直径端点为 (x, y)，加入 z 之后直径只可能是 $(x, y), (x, z), (y, z)$ 中的一种。

我们倒着考虑，首先最后一次肯定要想办法发送一下变成了什么，但如果只把这三种情况发出去，那么显然是没救的。

考虑困难一点的情况，假如最后一次操作能传出最后一次和倒数第二次的那该怎么传。

如果当前直径端点为 (x, y)，那么结果只可能是 $(x, n-1/n), (n-1/n, y), (n-1, n), (x, y)$，一共六种可能，所以 $Z = 5$ 其实是好做的。

考虑 $Z=4$ 怎么做，这个时候就要拆信息了。

考虑如何两次传三个信息，比如当前端点是 (x,y)，接下来分别加入了 (a,b,c)，该如何传递信息。

首先第一次我们只知道 (a,b) 和 (x,y) 之间的关系，注意到我们可以把 $(x,a/b)$ 看成一种。

然后如果这次传的是 $(x,a/b)$，那么下一次只可能是 $(x,a/b),(x/c),(a/b,c)$，这两 $Z=4$ 就可以传出来了。

于是我们就实现了两次传三个。

然后对于前面的情况，考虑分块传递信息，用 dp 构建决策即可。

考虑更高分的做法。

对于前面的分块传递信息，我们维护一下左右端点可能位于的集合，设左右端点分别处于 A, B 集合。

如果我们拿一个长度为 x 的段来传递信息，那么总共能传递 $4x+1$ 种不同的信息。

于是考虑分成 $[140, 20, 6, 2, 2]$（这个分段方法可以通过 dp 计算出来），注意到第一次的时候由于 $(A, B), (B, A)$ 是等价的，所以可以将 A, B 大小除以 33，这是因为 $\frac{33\times 34}{2} = 561 = 140 \times 4 + 1$，之后每次操作分别除以 $[9, 5, 3, 3]$。

我们来计算一下最后会剩多少（接下来除法默认上取整）。
本来有 $10^4 - 3 - 140 - 20 - 6 - 2 - 2 = 9827$ 个元素。
第一次 $9827/33 + 140 = 438$。
第二次 $438/9 + 20 = 69$。
第三次 $69/5 + 6 = 20$。
第四次 $20/3 + 2 = 9$。
第五次 $9/3 + 2 = 5$。
可以通过计算得出最坏情况最后剩余的 A, B 集合大小都为 5。

上述方法并不优秀，这是因为每一段的第一个位置也可以算到之前划分的状态里。

通过计算发现这样的话每次都可以少一个，于是接下来只需要考虑 A, B 集合大小都是 4 的情况。

相当于是一个 5×5 的网格图，挖掉左上角，然后你需要将网格图划分成五个部分。你希望划分到的面积为 5 的部分周长尽量地小。

可以手玩得出最终结果。

日常生活中的通信更多关注的不是效率，而是信息的安全性，即信息不能泄露。

比如你和你的同学考完试了，想比较一下谁的分高，但是双方都不想泄露自己的分数。你需要在不泄露任何信息的情况下完成这次对话。

实际上完全不泄露是很困难的，我们的目标是对方无法在短时间内（比如至少要用计算机算一百年）获取额外的信息。

你现在在一个 4 个人的小组，你很想知道这个组的考试成绩之和，但你不希望任何人知晓其余人的考试分数。并且你除了知道考试成绩总和之外也不知道任何信息。

不妨假设考试分数是 $[0, 10]$ 之间的整数。

设计一个通信方案使得信息不泄露。

注意到总分一定是 $0 \sim 40$ 之间，取 $P = 41$，第一个人先随机生成一个 $[0, P)$ 之间的数 x，告诉第二个人，第二个人加上自己的分数告诉第三个人，第三个人加上自己的分数告诉第四个人，第四个人加上自己的分数再告诉第一个人。这样第一个人就可以得到其余三个人的成绩总和。

注意到每个人接收到的数据本质上都是一个 $[0, P)$ 之间等概率随机的数，所以不会泄露信息。

在实际的通信中，为了保证信息不泄露，于是就出现了大量的加密算法。

大多数公钥密码学算法基于计算困难性假设，这样即便知道加密的逻辑也无法解密。

在实际的通信中，为了保证信息不泄露，于是就出现了大量的加密算法。

大多数公钥密码学算法基于计算困难性假设，这样即便知道加密的逻辑也无法解密。

数密码学问题的构造大多都是基于数论中的难题，比如分解质因数或者求离散对数。这些问题目前无法在输入数的位数的多项式时间复杂度内解决。

对称加密与非对称加密

假如现在 Alice 和 Bob 要进行交流，并且希望他们交流的信息别人无法获取。

那么一种最好的方法是他们先约定一组密钥，然后用这组密钥进行加密和解密。

对称加密与非对称加密

假如现在 Alice 和 Bob 要进行交流，并且希望他们交流的信息别人无法获取。

那么一种最好的方法是他们先约定一组密钥，然后用这组密钥进行加密和解密。

但是有个问题，就是 Alice 和 Bob 需要提前约定一组密钥，但有可能约定密钥时的通信也会泄露。

上述的加密方法是对称加密，即可以用一个密钥进行加密解密，缺点刚才也提到了。

于是就有了非对称加密，非对称加密同时拥有公钥和私钥，公钥用于加密，私钥用于解密，即便知道了公钥也无法在有效时间内得出私钥。

一种想法是 Alice 直接生成一对公钥和私钥，把公钥告诉 Bob，这样 Bob 就可以用公钥加密信息并发送给 Alice。

下面介绍一下非常经典的非对称加密算法：RSA 加密。

假设我们现在要对 x 进行加密，随机选取大质数 p, q，令 $P = p \times q$，然后选取 k_1, k_2 满足 $k_1 \times k_2 \bmod \varphi(P) = 1$，其中 $\varphi(P) = (p-1) \times (q-1)$，然后计算 x^{k_1}。

注意到分解质因数是难以高效计算的，所以即便知道 P, k_1, x^{k_1} 也无法还原出 x。

虽然 RSA 加密破解难度高，但是 RSA 效率较低且加密文本长度受限。

而对称加密则存在一些高效且文本长度无限制的算法，目前最常用的是 AES 算法。

如果需要进行大量的文本加密，可以先选取一种对称加密，然后用 RSA 对对称加密的密钥进行加密，这样也能实现非对称加密的效果。

半诚信模型

是一种对参与者行为的假设。在该模型中，节点被假定会：

1. 诚实地遵循协议规则：它们不会主动违反协议的规定步骤，例如不会伪造消息、不会丢弃应转发的信息。
2. 好奇并可能进行被动分析：它们会完整地记录下协议执行过程中所有经过它们的公开和私有信息，并试图从这些信息中推断出其他节点的私有数据（如交易金额、投票意向、身份等）。

通信方面，除了半诚信模型之外还有恶意模型等。不过今天我们接下来将在半诚信模型下对几种通信问题进行分析，我们认为通信过程中没有可信任的第三方。

不经意传输（OT）

Alice 现在有 n 个信封，Bob 想要查看第 k 个信封。通信过程中 Alice 不能知道 k，Bob 也不能知道除第 k 个信封之外的内容。

你可以认为 n 在一个可接受的范围。

选取大质数 p，和两个数 g, h。Alice 和 Bob 都知道 p, g, h，不妨假设接下来的运算都是 mod$p$ 意义下的。

Bob 随机一个数 b 并计算 $g^b h^k$ 发送给 Alice。

设 Alice 收到的数为 s，Alice 再随机 $a_{1\cdots n}$，计算 $x_i = (s \times h^{-i})^{a_i}$，并以 x_i 为密钥加密第 i 个信封的内容。

然后 Alice 把加密后的信封内容发给 Bob，然后再发送每个 g^{a_i} 给 Bob。

那么我们有 $x_k = (g^b h^k \times h^{-k})^{a_k} = g^{a_k \times b}$，于是 Bob 就知道了第 k 个信封的密钥，直接解密即可。

不过实际上只有 $n=2$ 是有意义的，注意到更大的情况可以设第 i 封信的密钥为 $C+\sum_{j\in S_i} q_j$，其中 S_i 表示 i 二进制为 1 的位对应的集合，$C, q_{1\cdots\log n}$ 均为随机数。然后就可以通过若干次 $n=2$ 的 OT 得出。

Alice 和 Bob 需要合作计算一个指定函数 $f(X, Y)$，其中 X 由 Alice 输入，Y 由 Bob 输入。Alice 需要知道最后的 $f(X, Y)$，要求不能泄露信息，这里不妨认为 $X, Y, f(X, Y)$ 范围都是 $1 \sim M$，其中 $M \leq 100$。

一种思路是 Alice 把加密后的数据和计算方法发给 Bob，然后 Bob 通过加密后的计算得到结果再发给 Alice，Alice 最后再解密。

因为信息被加密了，所以 Bob 无法得到 Alice 的相关信息。

首先 Alice 随机 $3m$ 个大整数 $A_{1\cdots m}, B_{1\cdots m}, C_{1\cdots m}$，然后将 A_X 发给 Bob。

之后 Bob 再通过 OT 得到 B_Y 的值，然后 Alice 对于所有的 X, Y，求出 $C_{f(X,Y)}$ 按照 A_X, B_Y 进行对称加密得到的结果，然后打乱并发送给 Bob。

然后 Bob 再根据手里的 A_X, B_Y 对所有的结果进行解密。

为了能让 Bob 自行验证结果是否正确，可以考虑让 $C_{1\cdots m}$ 的 16 进制下最低的若干位结果是 0，这样解密错误就大概率能够发现。然后 Bob 把解密成功的数发回给 Alice，最后 Alice 就可以得到 $f(X, Y)$。

混淆电路

考虑现在有一个数字电路，有些门是 Alice 进行输入，有些门是 Bob 进行输入，有些门是对某两个门进行逻辑运算，Alice 和 Bob 都知道整个电路的形态，Alice 需要知道电路中某个指定门的输出结果（Bob 也知道这个指定门）。

实际上跟刚才的做法是几乎相同的。

对于每个门的 01 状态随机相应的数，然后按照刚才问题的方式进行加密处理并发送给 Bob，然后将 Alice 的输入门的数发给 Bob，Bob 输入门的数可以通过 OT 获取。

然后 Bob 直接在本地计算出结果并将指定门的标签发送给 Alice。

我们都知道，数字电路的功能是强大的，可以实现加法器、乘法器以及各种复杂的运算，因此很多协同计算相应函数且不泄露信息的问题都可以迎刃而解。

谢谢大家