2023「钉耙编程」中国大学生算法设计超级联赛（8）题解

A. Chord

题目描述

求 $\sum_i \prod_j a_{i+b_j} \mod p$。

tag：离散对数，MTT

题解

注意到 a_i 不为 0，且都是乘法，考虑利用原根转化为加法。

设原根为 g，$a_i = g^{d_i}$，所求即为

$$\sum_i g^{\sum_j d_{i+b_j}} \mod p$$

观察该式，如果把长度为 m 的 b 转化成长度为 n 的 01 序列 c, c_i 表示 i 是否在 b 中出现，该式即为

$$\sum_i g^{\sum_j d_{i+j} \cdot c_j} \mod p$$

上式为卷积形式，我们将 c 翻转得到 c'，即 $c'_i = c_{b[m]-i}$，所求即为

$$\sum_i g^{\sum_j d_{i+j} \cdot c'_{b[m]-j}} \mod p$$

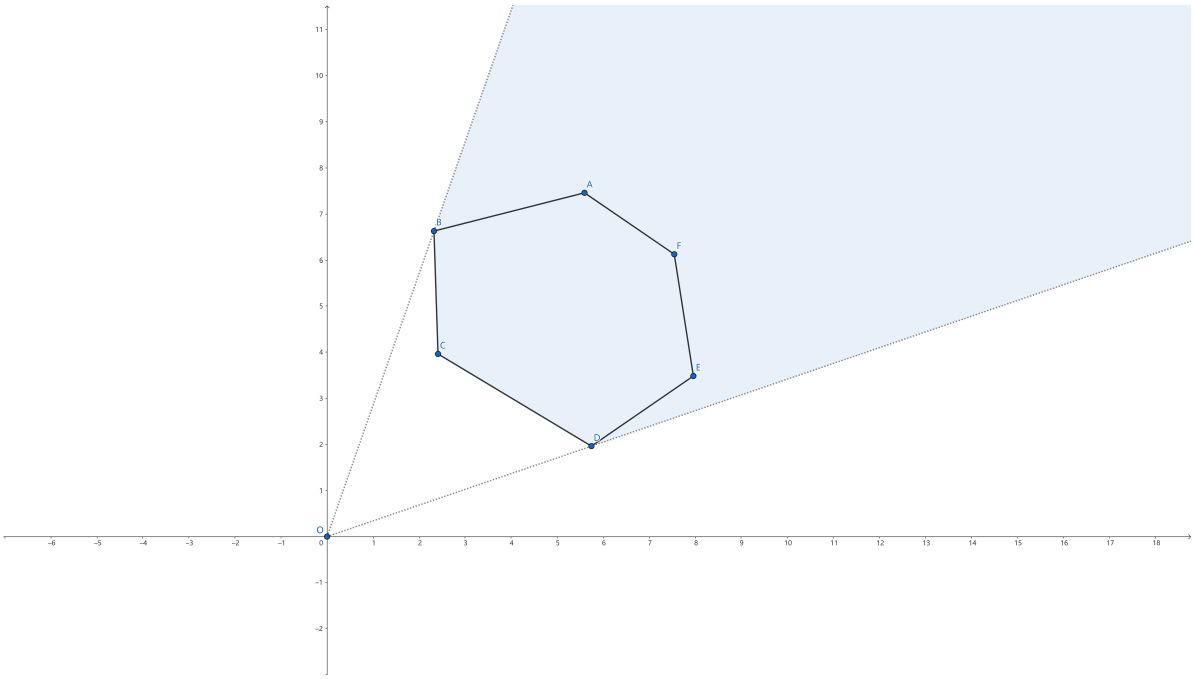
该式可以 FFT 加速，卷出第 $i + b[m]$ 位就是第 i 项贡献的指数部分，然后做快速幂再加起来即可。

关于实现问题和如何让这道题更毒瘤：

1. 求离散对数 d_i 时，每次 BSGS 会 T 飞，明显的优化是预处理大步的部分，每次只走小步，这个做法期望是被卡掉的，但是为了防止卡常时限开了标算 4 倍左右，不排除优良的实现（比如手写哈希表）可以过。
2. 标算的做法是观察到 $p - 1$ 光滑，所以可以用 Pohlig Hellman。做 Pohlig Hellman 的时候，因为会重复求很多次，没有必要每次对每个 p 做 BSGS，预处理时直接存下全部结果就可以单次 log 查询，否则有几倍常数，测题时能过但是很极限。
3. 用上一场 F 题的做法求离散对数也是可以的，虽然可以把 p 开大把除了 Pohlig Hellman 的做法统统卡掉，但是出题人觉得还是应该做个人，所以没刻意卡。这个做法的效率和标算基本相同。
4. 虽然没有把模数开得太大，但是指数部分大小为 $O(np)$，FFT 的精度显然是不够的，所以标算使用了 MTT。另外也可以写 二模数 NTT（因为两个模数 CRT 就足够求出真实值，所以不需要三模数），但是 NTT 在杭电 OJ 常数会略大一些。
5. 为了提示求离散对数，题目保证没有 $a_i = 0$ 的情况，但这个实际上是可以做的：因为最后对每个位置的贡献是单独求的贡献，只要判断到 $x = i$ 时会按到不可选的键就不算贡献即可，显然这又是个卷。如果出题人再毒瘤一点，就可以只放 Pohlig Hellman + MTT + FFT 过。

B. Delivery Robot

考虑会在 $\Delta t = 1$ 内会发生碰撞的 $\mathbf{v}$ 的范围，发生碰撞的条件可以写为 $\exists \mathbf{a} \in A, \mathbf{b} \in B, k \in [0, 1]$，使得 $\mathbf{a} + k\mathbf{v} = \mathbf{b}$，即 $k\mathbf{v} = \mathbf{b} - \mathbf{a}$，等式右侧形成的集合是凸集 B 与 $-A$ 的闵可夫斯基和，而如果 $\mathbf{v}$ 对应的有向线段与该闵可夫斯基和有交，则存在 $k \in [0, 1]$ 使得 $k\mathbf{v} = \mathbf{b} - \mathbf{a}$。研究 $\mathbf{v}$ 对应的有向线段与该闵可夫斯基和有交的所有情况，可以发现从原点出发作出闵可夫斯基和对应凸多边形的两条切线，$\mathbf{v}$ 应当落在两条切线与凸多边形连成的区域中（下图蓝色区域）。



对于每个询问，如果 $\mathbf{v}$ 没有落在上述区域中，则 $\Delta t = 1$ 内不会发生碰撞，无需对速度进行调整；如果落在上述区域中，那么需要找到将 $\mathbf{v}$ 移出该区域的最短距离，即 $\mathbf{v}$ 与边界的最短距离，枚举边界上的所有线段或射线求距离最小值即可。使用近期另一场比赛某题的方法可以实现 $O(q \log(n + m))$ 的复杂度，本题对此不作要求，实现正确的朴素算法即可，复杂度 $O(q(n + m))$。

C. Congruences

题目描述

解同余方程组 $x^{p_i} \equiv q_i \mod n$，其中 $n = \prod p_i$, 且 p_i 为两两不同的质数。

tag：费马小定理，中国剩余定理

题解

假设有解，显然有

$$\begin{aligned} & x^{p_i} \equiv q_i \mod n \\ \Rightarrow \quad & x^{p_i} \equiv q_i \mod p_i \\ \Leftrightarrow \quad & x \equiv q_i \mod p_i \end{aligned}$$

由于 p_i 为两两不同的质数，可以 CRT 合并得到 $x = kn + r$，这一步也是充要的。

所以在有解的情况下，可以直接 CRT 求出 r，并特判 $r = 0$ 的情况，即可得到最小正整数解。

然而注意到第一步两边并不互为充要条件，所以 CRT 求出的解不一定是原方程的解，因此需要验证这些解是否满足原方程。由于原方程对 n 取模，因此只需验证 r 一个数即可。若 r 满足原方程则有解，否则无解。

D. GG and YY's Binary Number

题目描述

给定一组 250 维的线性向量 V ($|V| \le n$) 和 m 个向量区间 $[l_j, r_j]$，令 $f(x) = \sum_{\{u_1,\dots,u_k\}\subseteq V}[u_1 \bigoplus u_2 \dots \bigoplus u_k = x])$，对每个区间求 $(\sum_{i\in[l_j,r_j]} f(i)) \mod (10^9+7)$。

数据范围：$1 \le n, m \le 250$。

题解

不难发现如果 x 能被 V 线性表出，则 $f(x) = 2^{n-k}$，其中 k 是 V 的线性基大小，令 $g(x) = \sum_{i\in[0,x]}[i \text{ 能被} V \text{线性表出}]$，则 $\sum_{i\in[l_j,r_j]} f(i) = (g(r_j) - g(l_j-1)) \cdot 2^{n-k}$，因此，只需要求解 $g(x)$ 即可。

做法 1：套用做过异或数是第几大和范围内线性表出这两类做法，首先求 $[0,x]$ 中可被表出的最大数 $largest(x)$，然后求该数是第几大的 $rank(largest(x))$，则 $g(x) = rank(largest(x))$。

做法 2：由做法 1，不难发现 $g(x)$ 可以用类似做法一遍从高位到低位的枚举+维护得出。

时间复杂度：$\mathcal{O}(250 \cdot n^2 + 250 \cdot nm)$，可以使用压位的方式来进行常数优化。

E. 0 vs 1

题目描述

有两名玩家分别叫做 0 和 1，他们轮流在一个给定的 01 串 S 中从左端或右端取字符。0 取到 1 输，1 取到 0 输，如果 S 取完也没有决出胜负就是平局，0 先手。问如果双方均采取最优策略，最终是否平局，以及如果不是平局，谁会获胜。

题解

如果当前字符串两端一个是 0 一个是 1，那么只有一种选法，直接选。这样一直推到两端是相同字符的情况，设当前两端的字符为 x。若当前的操作者不是 x，则该操作者输了，否则他需要从两端选择一端取字符。可以发现，如果有一端有连续两个 x，那么 x 直接取这边，x 的对手就会面临两端都是 x 的情况，于是 x 会赢。否则，两端的第二个字符都是 $!x$，因此不论 x 选那边，x 的对手在下一回合都会只有一种选法，也就是说，x 选哪边，下一回合 x 的对手也会跟着选哪边。可以理解为 x 获得了主动权。此外，我们发现，能决出胜负的方式仅有通过连续的两个相同字符来使得一方落败。于是我们可以分类讨论：当 x 获得主动权之后，只要距离两端最近的连续两个相同字符其中有一端是 xx，那么 x 一定会获胜，否则 $!x$ 一定会先获得主动权，然后让 x 落败。但要特判整个序列只有一个 $!x!x$ 的情况，这样是平局。

时间复杂度 $O(n)$。

F. Nested String

题目描述

定义在 T_1 中任意一个位置插入 T_2 得到的字符串为 T-嵌套串，问 S 串中有多少个 T-嵌套串。两个嵌套串不同，当且仅当它们在 S 中的位置不同或 T_2 插入的位置不同。

题解

正解做法是求 S 中以每个位置 l 起始的后缀与 T_1 的最长公共前缀（LCP）长度 p_l，以及以每个位置 r 结尾的前缀与 T_2 的最长公共后缀长度 q_r。后者相当于反着求 LCP，所以这个东西可以用 Z 函数（即 exKMP）预处理出来。假设 $r-l+1=|T_1|+|T_2|$，我们求 $S_{l\dots r}$ 对应的合法嵌套串数量。我们考虑 T_2 在 $S_{l\dots r}$ 中的嵌套位置，于是 p_l 将嵌套位置限制在 l 到 r 范围的一个前缀内，而 q_r 则将嵌套位置限制在 l 到 r 的一个后缀范围内，只需要对其求交集就可以知道 T_2 能够出现在哪些地方。于是我们需要求一个区间内与 T_2 匹配的字符串数量，可以先对 S 预处理一遍与 T_2 的 KMP 匹配，然后用前缀和统计。时间复杂度 $O(n)$。

本题还存在其它的理论线性做法，但实际上很难通过。对于哈希的做法（用哈希表作为桶统计所有满足条件的嵌套串），单哈希无法满足本题的冲突率（需要满足 $O(n)$ 个字符串不发生哈希碰撞），而双哈希本身常数很大，再加上数据范围较大，难以开出足够内存大小的内存池，这就导致哈希做法的常数过大（不论是时间上还是空间上），理论上很难通过本题。

G. Solubility

我们可以看出实际上题目所求的是给定的点集是否都在同一联通块中，采取并查集维护即可。

H. Expectation of Rank

题意：$A\in\mathbb{F}_p^{n\times n}$ 中每个元素均为取值范围为 $\mathbb{F}_p$ 的均匀随机变量，求 $\mathbb{E}[\operatorname{rank}A]$。

题解：令 $\mathrm{dp}_{i,j}$ 为满足 $\operatorname{rank}\{a_1\dots,a_i\}=j$ 的有序向量组数量。因为秩为 j 的 n 维向量组张成的空间内恰好有 p^j 个元素，考虑新加入的向量是否属于该空间，易得

$$\mathrm{dp}_{i,j}=\begin{cases}1 & i=0,j=0\\ 0 & i=0,j>0\\ (p^n-p^{j-1})\mathrm{dp}_{i-1,j-1}+p^j\mathrm{dp}_{i-1,j} & i>0\end{cases}$$

答案即

$$\mathbb{E}[\operatorname{rank}A]=\frac{1}{p^{n^2}}\sum_{k=0}^{n}k\mathrm{dp}_{n,k}$$

时空复杂度均为 $O(n^2)$。

I. Diagonal Fancy

题目描述

给定一个 $n\times m$ 矩阵，求其中有多少个连续子正方形，满足该子正方形内每个从左上到右下的对角线内部的数字相同，而不同对角线之间数字互不相同。

题解

定义 $dp[i][j]$ 为以 (i,j) 为左上角的满足第一个条件（即同一个对角线内数字相同）的最大子正方形的边长。我们可以采用经典的 dp 方式，从 $dp[i+1][j]$、$dp[i][j+1]$、$dp[i+1][j+1]$ 转移到 $dp[i][j]$，转移式如下：

$$dp[i][j]=\begin{cases}1, & a[i][j]\neq a[i+1][j+1]\\ \min(dp[i+1][j],dp[i][j+1],dp[i+1][j+1])+1, & a[i][j]=a[i+1][j+1]\end{cases}$$

接下来，对于条件二的保证，提供两种时间复杂度均为关于 nm 线性且常数小的做法。

双指针做法

可以发现，以 (i,j) 为左上角的最大正方形的右边界不大于以 $(i,j+1)$ 为左上角的最大正方形的右边界，即 $j+dp[i][j]-1 \le (j+1)+dp[i][j+1]-1$。如果我们把 j 看作左端点，$j+dp[i][j]-1$ 看作右端点，会发现右端点随着左端点的增加而单调不减。这满足双指针算法所要求的单调性。

于是我们可以对每一行单独进行双指针，这一次我们对第二个条件（不同对角线之间数字不同）进行保证，设双指针的左右端点分别为 l 和 r，我们需要保证在 $r \le l+dp[i][l]-1$ 的前提下，找出最大的 r 满足以 l 为左边界、r 为右边界的子正方形是满足条件二的。由于已经保证了 $r \le l+dp[i][l]-1$，也就是已经保证了条件一一定满足，设 $w=r-l+1$，我们只需要检查这 $2w-1$ 条对角线内的数字均不同即可。我们可以用一个桶来维护所有出现过的数字，当右端点从 $r-1$ 变换到 r 时，我们只需要尝试将 $a[i][r]$ 和 $a[i+r-l][l]$ 加入桶即可，若发生冲突则说明已到达最大的合法右端点；当左端点从 l 变换到 $l+1$ 时，只需要将 $a[i+r-l][l]$ 和 $a[i+r-l-1][l]$ 从桶中删除即可。

纯 DP 做法

在上述转移式的基础上，我们继续增加对第二个条件的保证。现在我们令 $dp[i][j]$ 表示以 (i,j) 为左上角的同时满足两个条件的最大的子正方形的边长。假设 $a[i][j] \ne a[i+1][j+1]$，我们继续考虑类似的转移方式。设 $k=\min(dp[i+1][j],dp[i][j+1],dp[i+1][j+1])$，我们用下图表示以 (i,j) 为左上角的边长为 $k+1$ 的子正方形的每一条对角线：

$$\begin{bmatrix} l+k & l+k+1 & \cdots & r-1 & r \\ l+k-1 & l+k & \cdots & r-2 & r-1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ l+1 & l+2 & \cdots & l+k & l+k+1 \\ l & l+1 & \cdots & l+k-1 & l+k \end{bmatrix}$$

其中，标记为相同数字的位置属于同一个对角线，我们称这些对角线分别为 $l, l+1, \cdots, r$。

从而，$dp[i+1][j]$ 保证了第 $l \sim r-2$ 条对角线互不相同；$dp[i][j+1]$ 保证了第 $l+2 \sim r$ 条对角线互不相同；$dp[i+1][j+1]$ 保证了第 $l+1 \sim r-1$ 条对角线互不相同。于是，我们只需要检查第 l，$l+1$，$r-1$，r 是否互不相同，若是，则 $dp[i][j]=k+1$，否则 $dp[i][j]=k$。

J. Rikka with Square Numbers

题目描述

给两个整数 a,b，每次可以加或减一个完全平方数，问最少几次才能把 a 变成 b。

数据范围：$1 \le a,b \le 10^9, a \ne b$。

题解

不妨令 $a<b$，$a>b$ 可以交换 a,b，情况类似。

令 $d=b-a$，若 d 是完全平方数，则只需一步完成。

考虑 d 不是完全平方数。当 d 为奇数时，存在两整数 p,q，满足 $p+q=d$ 且 $|p-q|=1$，不妨设 $p>q$，则先加 p^2 再减 q^2 即可，且 $p^2-q^2=(p+q)(p-q)=d$。这样是满足要求的变换中次数最少的情况，只需两步。

考虑 d 为偶数，当 $d \bmod 4=0$ 时，则 $d=4p=(p+1)^2-(p-1)^2$，此时 $p \ne 1$，因此只需两步。

当 $d \bmod 4 = 2$ 时，如果能用两步完成，则 d 可能是两平方数之和或差；如果 d 无法分解成两平方数之和或差，则只能加 1 转化为差为奇数的情况，因此需要两步或三步。对于判断 d 是否可以分解成两平方数之和，可以枚举 $\sqrt{d}$ 内的整数 i，判断 $d - i^2$ 是否为平方数即可。对于判断 d 是否可以分解成两平方数之差，考虑 $d = (p + q)(p - q)$，而此时 d 中一定只有一个质因子 2，因此分配乘积时 $p + q$ 和 $p - q$ 一定存在一个奇数，无法得到整数 p, q，因此 d 无法分解为两平方数之差。综上，仅需判断 d 是否能分解成两数平方和即可，若可以则需两步，否则需三步。

时间复杂度：$\mathcal{O}(\sqrt{|b - a|})$，空间复杂度：$\mathcal{O}(1)$。