

CCF CAT 2025 决赛题解

哈尔滨工程大学

2025/08/07

Problem A. 备用账号

按题意，将相似的用户名归为一组，大小为 S 的组对答案产生的贡献为 $\binom{S}{2}$ ，使用 `std::map` 或类似的数据结构维护即可。时间复杂度为 $O((N+L)\log N)$ 或 $O(N+L)$ ，其中 L 为用户名的总长度。

Problem B. “进制”变换

idea: Fresca ; solution: Fresca ; prepared by: Fresca

题意概述

- 要求不超过 10 次操作把 a 变成 b ，每次操作可以选择一个正整数 k ，通过 $f(n) = \sum_{i=0}^{n-1} x_i \times k^{i+1}$ 把 $\overline{x_{n-1}x_{n-2}\cdots x_0}$ 变成新的整数。
- $1 \leq a, b \leq 10^9, a \neq b$ 。

Problem B. “进制” 变换

- 观察操作过程发现，可以实现两种操作。
- 第一种操作，可以把一个一位数变成这个数的任意倍数，特别的，可以把 1 变成任意正整数。
- 第二种操作，可以通过 $k = 1$ ，把一个数变成它的数位和。
- 一种构造方式是通过小于等于 9 次的操作把 a 变成 1，再利用第一种操作变成 b 。
- 利用 $k = 9$ 可以把一个数字变成 9 的倍数，而 9 的倍数持续求数位和均为 9 的倍数，最终会变为 9，对于题目的数据范围，这个过程可以在小于等于 3 次的操作解决。
- 通过手玩或打表发现 $9 \xrightarrow{k=6} 54 \xrightarrow{k=2} 28 \xrightarrow{k=1} 10 \xrightarrow{k=1} 1$ 。
- 总共只需 $3 + 4 + 1 = 8$ 次操作。

Problem B. “进制” 变换

- 另外一种构造方式是，我们可以先通过取数位和，将 a 变化成一个 ≤ 9 的数字。此时次数一定是 ≤ 3 的。随后，我们总是有方式将它变成 1。具体的：
- $2 \xrightarrow{k=5} 10 \xrightarrow{k=1} 1$
- $3 \xrightarrow{k=6} 18 \xrightarrow{k=2} 20 \xrightarrow{k=1} 2 \xrightarrow{k=5} 10 \xrightarrow{k=1} 1$
- $4 \xrightarrow{k=5} 20 \xrightarrow{k=1} 2 \xrightarrow{k=5} 10 \xrightarrow{k=1} 1$
- $5 \xrightarrow{k=2} 10 \xrightarrow{k=1} 1$
- $6 \xrightarrow{k=3} 18 \xrightarrow{k=2} 20 \xrightarrow{k=1} 2 \xrightarrow{k=5} 10 \xrightarrow{k=1} 1$
- $7 \xrightarrow{k=2} 14 \xrightarrow{k=1} 5 \xrightarrow{k=2} 10 \xrightarrow{k=1} 1$
- $8 \xrightarrow{k=5} 40 \xrightarrow{k=1} 4 \xrightarrow{k=5} 20 \xrightarrow{k=1} 2 \xrightarrow{k=5} 10 \xrightarrow{k=1} 1$
- $9 \xrightarrow{k=2} 18 \xrightarrow{k=2} 20 \xrightarrow{k=1} 2 \xrightarrow{k=5} 10 \xrightarrow{k=1} 1$

Problem C. 或非

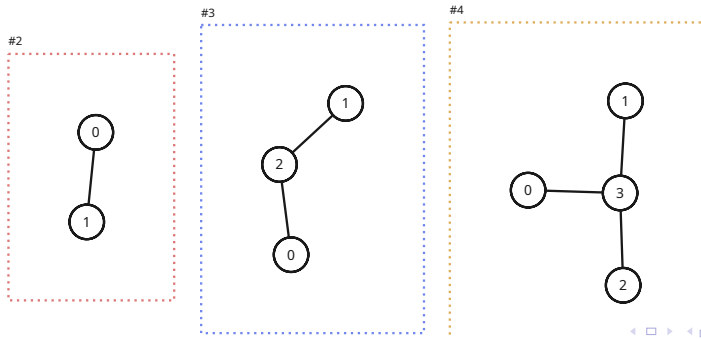
idea: frost_ice ; solution: frost_ice ; prepared by: frost_ice

题意概述

- 给定一个整数 n ($2 \leq n \leq 2 \times 10^5$)。
- n 个节点的无向图，节点编号为 0 到 $n-1$ ，节点 u 和 v 之间 ($0 \leq u < v < n$) 连有一条边权为 $\text{nor}(u, v)$ 的边。
- 这里 $\text{nor}(x, y)$ 定义为“按位或非”运算，即对第 0 位到第 29 位依次采取「有 1 为 0 ，否则为 1 」的逻辑或非运算。
- 要求找到一棵生成树，使得边权的按位异或结果尽可能小。

Problem C. 或非

- $n = 2$ 时答案为 $\sum_{i=1}^{29} 2^i$ 。
- $n = 3$ 时答案为 1。
- $n = 4$ 时答案为 $\sum_{i=2}^{29} 2^i$ 。
- 以下是这三种情况下，一种可行的构造方案。



Problem C. 或非

- 当 n 更大时, 我们考虑每个二进制位对答案的贡献。
- 如果 $0 \sim n-1$ 的所有数, 第 i 个二进制位都为 0, 那么根据或非运算的性质, 全部的 $n \times (n-1)/2$ 条边的边权, 第 i 位都一定是 1。而我们需要选出 $n-1$ 条边, 那么它一定会对答案产生 $2^i \times ((n-1) \bmod 2)$ 的贡献。
- 而如果存在 $0 \sim n-1$ 中的某个数, 它的第 i 个二进制位为 1, 那么这 $n \times (n-1)/2$ 条边当中, 有些边的第 i 位为 0 而有些为 1。为了答案最小, 我们希望选出偶数条第 i 位为 1 的边。这样按位异或的结果中, 该位的贡献就一定是 0。
- 下面我们通过构造证明这种方案一定是存在的, 从而答案能够取到理论最小值:

$$((n-1) \bmod 2) \times \sum_{i=\lfloor \log_2(n-1) \rfloor + 1}^{29} 2^i$$

Problem C. 或非

- 一种方案如下：
- 我们取最大的满足 $2^c \leq n-1$ 的整数 c 。令 $x = 2^c$ 。
- 连接 u 和 v 的边，只要 u 和 v 中有一个数第 i 位为 1，那么边权的第 i 位就为 0。
- 对于 $x+1 \leq v < n$ 的这些点 v ，它们不是很好处理。因此考虑将它们全部连到 $x-1$ 上。由于 $x-1$ 这个数，第 0 位到第 $c-1$ 位数全是 1，而 v 的第 c 位一定是 1，因此这些边的第 0 位到第 c 位就全是 0。

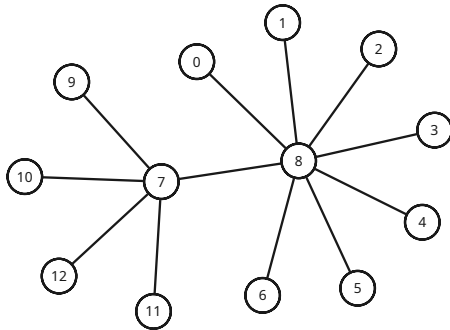
```
1 000111110
2 000111111 x-1
3 001000000 x
4 001000001 x+1
5 001000010
6 .....
7 001xxxxxx n-1
```

Problem C. 或非

- 现在 $x+1 \sim n-1$ 的点都跟 $x-1$ 联通了。接下来处理剩余的数。
- 我们可以直接将 $0 \sim x-1$ 的所有节点，连到 x 上去。
- 考虑 $2i$ 和 $2i+1$ 这两个点，它们跟 x 的边的边权一定只有最低位是不同的，因此这两条边的按位异或结果一定是 1。而当 $x \geq 4$ 时候，这样的 $2i$ 和 $2i+1$ 数字对，一定有偶数个，因此这部分按位异或的结果就一定是 0。
- 至此我们完成了构造。

Problem C. 或非

- 树的形态大致如下图所示：



Problem D. Fever Dash

idea: jiangdu; solution: jiangdu; prepared by: jiangdu, frost_ice

题意概述

模拟音乐游戏中的一个加分机制：每个音符有固定的得分和 Fever 值，可以通过击打音符积攒 Fever，达到上限后可开启持续 k 时刻的 Fever 状态，在该状态下击打音符分数翻倍但不再获得 Fever 值。你可以任意时刻开启 Fever（前提是 Fever 值已满），但不能主动中断。给定所有音符的落下时间、分数和 Fever 值，求一个最优的 Fever 开启方案，使得总得分最大。

Problem D. Fever Dash

- 首先所有音符的总分 $\sum p_i$ 必定会累加在最终答案里，因此我们只需要计算，通过 fever 最多能额外获得多少分即可。
- 设只考虑前 i 个音符，且 fever 内最后一个音符为 i 时，最多能额外获得的分数为 dp_i
- 当次 fever 的结束时间 T_e 需要属于 $[t_i, \min\{t_{i+1} - 1, t_i + k - 1\}]$
- 开始时间 T_s 需要属于 $[t_i - k + 1, \min\{t_{i+1} - k, t_i\}]$ （即往前推 $k - 1$ 个时刻）
- 这里的开启和结束的意思是，所有满足 $T_s \leq t_x \leq T_e$ 的音符 x 都能被覆盖到。
- 在本次 fever 开始前的最后一个音符 j 需要满足 $t_j < T_s$ 且 $T_s \leq t_{j+1} \leq T_e$ 。不难看出，如果 T_s 确定为某个值，则 j 是满足 $t_j < T_s$ 的最后一个音符。
- 因为 T_s 实际可能在一个区间范围内，因此满足条件的 j 也会构成一个区间。
- 我们记 l 是最后一个满足 $t_l < t_i - k + 1$ 的音符， r 是最后一个满足 $t_r < \min\{t_{i+1} - k, t_i\}$ 的音符。那么满足条件的 j 构成的区间就是 $[l, r]$ 。

Problem D. Fever Dash

- 令 last_x 表示: 如果需要打完第 x 个音符后, fever 值积累满, 则最晚需要从哪个音符开始积攒。如果没法积攒满, 则 last_x 为 0。
- sum_x 表示: 前 x 个音符的分数的前缀和, 即 $\sum_{i=1}^x p_i$ 。特别的, sum_0 为 0。
- 令 maxdp 表示 dp 的前缀最大值, 即 $\text{maxdp}_x = \max_{i=1}^x \text{dp}_i$
- 则有转移方程:

$$\begin{aligned} dp_i &= \max_{\substack{l \leq j \leq r \\ \text{last}_j \neq 0}} (\text{maxdp}_{\text{last}_j-1} + \text{sum}_i - \text{sum}_j) \\ &= \text{sum}_i + \max_{\substack{l \leq j \leq r \\ \text{last}_j \neq 0}} (\text{maxdp}_{\text{last}_j-1} - \text{sum}_j) \end{aligned}$$

- 题目所求答案就是 $\text{maxdp}_N + \text{sum}_N$

Problem D. Fever Dash

Solution1:

- 由于上述 $\max$ 中的表达式 $\max dp_{\text{last}_j-1} - \text{sum}_j$ 只跟 j 有关, 因此我们可以在 DP 的过程中使用单调队列、大根堆、线段树等方式进行维护。
- 时间复杂度 $O(N)$ 或 $O(N \log N)$

Problem D. Fever Dash

Solution2:

- 在枚举到音符 i 时满足条件的音符 j 的范围 $[l, r]$ 中, l 是最后一个满足 $t_l < t_i - k + 1$ 的音符, r 是最后一个满足 $t_r < \min\{t_{i+1} - k, t_i\}$ 的音符
- 忽略左端点 l , 剩下的 j 满足: $t_j \in [t_i - k + 1, \min(t_{i+1} - k, t_i)]$
- 由于 t_i 单增, $[t_i - k + 1, t_{i+1} - k]$ 互相无交集
- 因此所有音符 i 对应的 $(l, r]$ 区间互相不交, 因此所有 i 对应的 $(l, r]$ 范围内总计最多有 N 个音符。
- 而所有的 l 总计最多对应 N 个音符。
- 因此直接枚举 $[l, r]$ 总计至多枚举到 $2N$ 个音符, 暴力计算该 DP, 时间复杂度为 $O(N)$

Problem E. 简单的数据结构题

idea: frost_ice ; solution: frost_ice ; prepared by: frost_ice

题意概述

- 要求维护一个长度为 n 的序列，支持三种操作：
- 操作 1： $[l, r]$ 区间所有数模 16 意义下 $+1$ 。
- 操作 2： $[l, r]$ 区间所有数在 4 位二进制数意义下循环左位移 1 位。
- 操作 3： 查询 $[l, r]$ 区间内 $0 \sim 15$ 的数字个数。
- $n, q \leq 5 \times 10^5$

Problem E. 简单的数据结构题

- 我们考虑维护区间内 $0 \sim 15$ 出现的次数, 记为 $cnt[0] \sim cnt[15]$ 。
- 我们将前两种操作都看成置换。
- 具体的, 我们令 $A = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0]$, 同时 $B = [0, 2, 4, 6, 8, 10, 12, 14, 1, 3, 5, 7, 9, 11, 13, 15]$ (下标从 0 开始)
- 对一个区间施加操作 1 就等效于将 $cnt[0], cnt[1], cnt[2], \dots, cnt[15]$ 变为 $cnt[A[0]], cnt[A[1]], cnt[A[2]], \dots, cnt[A[15]]$ 。操作 2 同理。
- 这样的置换可以复合。并且对于两个区间, 先合并再施加操作, 与先施加操作再合并是等效的。
- 因此用线段树维护即可。时间复杂度 $O(16n \log n)$

Problem F. 乱序法杖

idea: frost_ice; solution: jiangdu; prepared by: jiangdu, frost_ice

题意概述

给定一个整数数组 a ，和两种操作 AB：

- 操作 A： a 中所有正数 $+1$ ，负数 -1 ，0 不变；
- 操作 B： a 中所有正数 -1 ，负数 $+1$ ，0 不变。

给出 x 个操作 A 和 y 个操作 B，将这 $x+y$ 个操作以等概率打乱顺序后依次施加在数组上，求操作完毕后数组中每个元素的期望值是多少？

Problem F. 乱序法杖

- 数组中每个元素是独立的，考虑单个元素 a_i 的期望值
- 操作 A 和 B 的情况是对称的，我们以操作 A 进行分析
- 对于一个初始为正数的 a_i ，在操作之后要么血量归零，要么最后值为 $a_i + x - y$
- 考虑计算 a_i 未被归零的方案数，即要求任意前缀有 $a_i + cnt_A - cnt_B > 0$ ，其中 cnt_A 代表操作 A 的个数。
- 注意到该式类似于卡特兰数的定义，即要求任意前缀有 $cnt_A - cnt_B \geq 0$ ，因此可以使用卡特兰数的思想来计算方案数。

Problem F. 乱序法杖

定理

a 个 A 类元素和 b 个 B 类元素，任意前缀 $cnt_b - cnt_a \leq k$ 的序列的方案数：

$$F(a, b, k) = \begin{cases} 0, & b - a > k \\ \binom{a+b}{a} - \binom{a+b}{a+k+1}, & \text{else} \end{cases}$$

证明.

- 考虑计算在网格图上从 $(0, 0)$ 走到 (a, b) 不能碰到 $L: y = x + k + 1$ 的方案数。
- 若 $b - a > k$ 则必须穿过 L ，因此方案数为 0。
- 考虑用总方案数 $\binom{a+b}{a}$ 减去会碰到 L 的方案数，后者等于 $(0, 0)$ 走到 (a, b) 关于 L 的对称点的方案数。横向需要走 $b - (k + 1)$ ，纵向需要走 $a + k + 1$ ，等于 $\binom{a+b}{a+k+1}$ 。



Problem F. 乱序法杖

- 满足条件 $a_i + cnt_A - cnt_B > 0$ 的方案数即为 $F(x, y, a_i - 1)$
- 对于所有方案的值之和为 $S = (a_i + x - y) \times F(x, y, a_i - 1)$
- 所求期望还需要再除以总方案数 $C = \binom{x+y}{x}$

Problem G. Good Number

idea: frost_ice ; solution: frost_ice ; prepared by: frost_ice

题意概述

- 我们定义一个数的美观程度 $G(x)$ 为：

$$G(x) = \sum_{\substack{0 \leq i \leq 9 \\ F(x,i) > 0}} F(x,i)^{F(x,i)}$$

- 其中 $F(x,i)$ 表示 x 的十进制表示下数字 i 的出现次数。
- 给定区间 $[L, R]$ 以及整数 K ，求区间 $[L, R]$ 内所有 $G(x) \geq K$ 的数字的数量以及它们的 $G(x)$ 的总和。

Problem G. Good Number

- 首先差分一下，将问题改为对 $[1, R]$ 内的数进行求解。
- 假设 R 一共有 s 位。对于位数与 R 相同的数字，按照数位 dp 的套路，我们枚举它们的最长公共前缀的长度。假设长度是 i ($0 \leq i < s$)。剩下的 $s - i$ 个位置，我们可以直接枚举 $0 \sim 9$ 的数字各出现多少次。这样的枚举量是 $\binom{s-i+9}{9}$ 的。
- 由于相等的前缀是已知的，后续各个数字出现多少次是我们枚举到的，此时我们可以去计算这个数字的 $G(x)$ ，判断它是否 $\geq K$ 并累加到答案上。
- 而它对两个答案的贡献就分别是：方案数，以及 $G(x)$ 乘上方案数。

Problem G. Good Number

- 我们假设当前枚举的出现次数分别是 $cnt_0 \sim cnt_9$ 。
- 如果第 $i+1$ 位可以填的数字构成的区间是 $[l, r]$ (具体的, $i=0$ 时, l 为 1, 因为不能有前导 0, 其余情况 l 都是 0; 而 r 始终为 R 的第 $i+1$ 位的数减一), 我们可以先枚举 $[l, r]$ 以外的数字并用组合数计算它们填在后 $s-i-1$ 个位置的方案, 之后再枚举 $[l, r]$ 当中的数字计算填进去的方案。
- 具体来讲, 我们用一个变量 rem 记录还剩多少个位置没有填, rem 初始为 $s-i-1$ 。我们先枚举 $l \sim r$ 以外的的数 j , 将 $\binom{rem}{cnt_j}$ 乘到方案数上, 随后将 rem 减去 cnt_j 。枚举完 $l \sim r$ 以外的数字后, 我们将 rem 加上 1 (因为在此之后, 第 $i+1$ 位就可以被填了), 重复这一步骤计算 $l \sim r$ 中的数的答案, 即可 $O(10)$ 的计算得到方案数。

Problem G. Good Number

- 至于位数 $< s$ 的那些数，计算方法也是类似的。重复上述过程即可。
- 总计算量大约是 $2 \times 2 \times 10 \times \sum_{i=1}^{15} \binom{i+9}{9} = 20 \times \binom{25}{10} \approx 1.3 \times 10^8$
- 实际常数较小，可以通过。

Problem H. 快速排列置换

idea: jiangdu ; solution: frost_ice ; prepared by: frost_ice

题意概述

- 给定一个长为 n 的置换 a 。接下来重复 k 次，每次将 a 变为自身与自身的复合。求最终结果。
- $1 \leq n \leq 10^5$, $0 \leq k \leq 10^{10^5}$ 。

Problem H. 快速排列置换

- 题目等效于求：

$$\underbrace{a \circ a \circ a \circ \dots \circ a}_{2^k \uparrow a}$$

- 其中 $\circ$ 符号表示置换的复合。
- 即，我们取一排列 c ，初始等于 a ，随后进行 $2^k - 1$ 次操作，每次对每个位置 i ，将该位置上的数（记为 c'_i ）变为 $a_{c'_i}$ 。
- 我们考虑一个 n 个点 n 条边的有向图， i 到 a_i 连有有向边。
- 根据置换的性质，该有向图每个点的入度和出度都为 1，并且由若干个有向环组成。
- 最终答案的第 i 个位置，一定是从 i 出发，沿着出边走 $2^k - 1$ 步之后的结果。

Problem H. 快速排列置换

- 对于一个长度为 l 的有向环，我们只需要知道 $(2^k - 1) \bmod l$ 的结果，即可知道该环中的每一个点对应的位置的答案是多少。
- 由于这些环的总长度是 n ，因此这些环至多只有 $\sqrt{n}$ 种不同的长度。
- 对这 $\sqrt{n}$ 种不同的长度分别求出 $(2^k - 1) \bmod l$ 的结果即可。
- 当 k 较小时 ($\leq 10^6$)，我们可以直接计算结果。
- 否则，由于 $k > \varphi(l)$ ，所求表达式等于 $(2^{(k \bmod \varphi(l)) + \varphi(l)} - 1 + l) \bmod l$
- 预处理出所有的 $\varphi(l)$ 之后即可进行计算。
- 时间复杂度 $O(n + \sqrt{n} \log_{10} k)$

Problem I. Emotional Flutter

idea: SkyRainWind ; solution: SkyRainWind ; prepared by: SkyRainWind, frost_ice

题意概述

- 给定三个凸多边形，第 i 个多边形有 n_i 个点，且质量密度已知。一共有 q 次询问，每次询问给出一个坐标 (X, Y) ，问是否可以在三个多边形中各选择一个点，使得这三个点的质心位于 (X, Y) ，强制在线。
- $1 \leq n_i, q \leq 5 \times 10^5$ 。

Problem I. Emotional Flutter

- 首先我们根据质心的定义（下式），联想到可以将每个凸包的顶点乘以对应的质量密度，使用新凸包的点进行查询以去掉质量密度这个限制。
- $$\frac{m_1 x_1 + m_2 x_2 + m_3 x_3}{m_1 + m_2 + m_3} = X \Leftrightarrow m_1 x_1 + m_2 x_2 + m_3 x_3 = (m_1 + m_2 + m_3) X$$
- 具体来说，我们通过对凸包的顶点应用变换： $x_i \rightarrow m_i x_i$ ，来获得新凸包的点。现在，我们将问题转化成了：在三个新凸包中各自任取一点，将这三个点的坐标直接相加，是否存在一种结果恰好为 $((m_1 + m_2 + m_3)X, (m_1 + m_2 + m_3)Y)$ 的方案。
- 接着，我们需要求出在两个凸包中任取一点并相加，得到的点构成的集合的形状。这是经典的凸包合并问题，可以使用闵可夫斯基和解决。由于两个凸包合并之后仍然是凸包，因此可以轻松扩展到三个凸包的情况中。
- 对于每次查询，我们只需要知道 $((m_1 + m_2 + m_3)X, (m_1 + m_2 + m_3)Y)$ 是否在合并后的凸包中，这个可以通过极角排序 + 二分轻松实现。
- 时间复杂度 $O((n + q) \log n)$

Problem J. Eternal Feather II

idea: SkyRainWind ; solution: SkyRainWind ; prepared by: SkyRainWind

题意概述

- 有一个长度为 n 的，由 $\{y, u, k, o\}$ 为字符集构成的字符串。求出这样的串的个数，使得这个串不包含一个子串，这个子串经过重新排列之后可以构成 *yuuko*。
- $1 \leq n \leq 10^{18}$ 。

Problem J. Eternal Feather II

- 考虑一个长度为 4 的串，我们需要保证加入下一个字符之后经过重排也不构成 *yuuko*。
- 首先，如何构建每个串的状态？我们可以将这个串正则化一下。具体来说，将 *u* 固定设成 5，将其他字符按照最早出现次数设置成 0, 1, 2. 这样，我们就得到了每个长度为 4 的串的状态（注意按照这样的做法，可能出现不同的串的状态一样，如 *yuok* 和 *ouyk* 都是 0512。但由于这两种状态本质相同，因此无所谓）
- 得到每个串的状态之后，我们可以开始构建 DFA，由于串长很短，每个位置的可能性也很少，可以直接暴力构建，构建之后发现可能的状态只有 ~ 50 种。
- 我们可以根据 DFA 构建转移矩阵，只需要保证加入下一个字符并排列之后不构成 *yuuko* 即可。例如，如果不构成的话，可以将两个状态对应的点连边，对应转移矩阵（也就是邻接矩阵）元素 +1。
- 构建出转移矩阵后，直接矩阵快速幂解决。时间复杂度 $O(50^3 \log n)$