2025“钉耙编程”中国大学生算法设计暑期联赛（2）题解

2025 年 7 月 21 日

概况

- Easy：B、F、H
- Easy-Medium：I、L、K
- Medium：C、G
- Medium-Hard：J、A
- Hard：E、D

现场情况

通过人数

A	B	C	D	E	F
36	1072	22	2	5	942
G	H	I	J	K	L
90	1013	349	5	52	172

首次通过

A	B	C	D	E	F
0:44	0:02	0:54	3:13	1:02	0:04
G	H	I	J	K	L
0:07	0:03	0:13	1:40	1:10	0:17

B. 数上的图

题目大意

- 给定正整数 n 和区间 $[1,n]$ 内的起点 x 和终点 y，你需要通过若干次操作将 x 变成 y。
- 每次操作可以任选一个 $i(1 \le i \le n)$，只要满足以下条件之一即可将当前值 x 变成 i：
  - $\mathrm{count}(x) = \mathrm{count}(i)$，即二进制表示下 1 的个数相同。
  - $\mathrm{lowbit}(x) = \mathrm{lowbit}(i)$，即二进制表示下最低位 1 及其后面所有 0 构成的数值相同。
- 请求出将 x 转化为 y 的最小操作次数。
- 数据范围：$1 \le x, y \le n \le 10^{15}$。

B. 数上的图

可以发现，答案不超过 2。不难得出：

- 当 $\mathrm{count}(x) < \mathrm{count}(y)$ 时。先使用操作一、再使用操作二，即可从 x 到 y。
- 当 $\mathrm{count}(x) \geq \mathrm{count}(y)$ 时。先使用操作二、再使用操作一，即可从 x 到 y。

特判一下答案为 $0, 1$ 的情况，其余情况答案必为 2。

F. 半

题目大意

- 给出两个关于 n 的排列 a, b，分别表示在两场个人比赛中，同一批 n 名选手从高到低的排名编号。
- 你不仅是 n 名选手中的其中一个，也是一个管理员。你可以随意禁赛其他选手。被禁赛后其余选手在两场比赛中的相对名次不变。
- 对于所有的 $1 \leq i \leq n$，你需要求出你是选手 i 时，你想要在两场比赛中均获得第一名，最少需要禁赛多少个选手。
- 数据范围：$1 \leq n \leq 10^6$。

F. 半

设选手 i 在两场比赛中的排名分别是 p_i, q_i。则对于选手 i，需要将所有满足 $p_j < p_i$ 或 $q_j < q_i$ 的所有选手 j 禁赛。

可以用“两场比赛排名分别比 i 高的人数”减去“两场比赛排名都比 i 高的人数”得到答案，即

$$(p_i - 1) + (q_i - 1) - \sum_{i=1}^{n} [p_j < p_i \land q_j < q_i]$$

F. 半

求 $\sum_{i=1}^{n}[p_j < p_i \land q_j < q_i]$ 是一个二维偏序问题。

将所有选手以 p_i 为第一关键字，q_i 为第二关键字从小到大排序。排序过后即可消除 $p_j < p_i$ 的偏序关系，使用树状数组维护已经扫过选手的 q_i 值。设当前扫到的是第 i 位选手：

- 查询树状数组中小于 q_i 的值的个数，得到 $\sum_{i=1}^{n}[p_j < p_i \land q_j < q_i]$ 的值。
- 将 q_i 插入树状数组。

时间复杂度 $O(n \log n)$。

H. 井

题目大意

- 有一张 $n \times n$ 的网格图，网格里的数为 0 或 1。恰有一行或一列的数为 1，其余的数均为 0。
- 这一张网格图会在 $2n$ 种可能的状态中，均匀随机地选择一种状态出现。
- 每个格子在开始时都是盖上的，你需要按照你的决策依次翻开网格里的数（由你决定翻开网格的位置）。当所有 1 都被翻出时，翻数结束。你需要使用最优策略，使得翻开所有 1 的期望次数最小。
- 数据范围：$1 \le n \le 10^9$。

H. 井

记全是 1 的行或列为“特殊线”。

如果翻到 (i,j) 为 0，则可以排除第 i 行或第 j 列是特殊线的可能。

如果翻到 (i,j) 为 1，则可以确定特殊线为第 i 行或第 j 列的其中一个。

故最优策略即为：

- 翻开第一个 1 之前，不选择重复的行或列；
- 翻开第一个 1 之后，直接判定特殊线是行还是列，然后将特殊线上的 1 全部翻开。

故答案为

$$\frac{1+\cdots+n}{n}+\frac{1}{2}(n+(n-1))=\frac{3n}{2}$$

I. 苹果树

题目大意

- 给出一棵包含 n 个点的树，每个节点 i 都有一个权值 a_i。有 m 次操作，每次操作都形如以下的两种：
  - `1 x y`：查询 x 到 y 的路径上，最大的点权权值。
  - `2 x z`：对于所有与 x 直接相连的点 i，令 $a_i \leftarrow a_i + z$。
- 数据范围：$1 \le n, m \le 10^5$。

I. 苹果树

该做法来自某验题队伍，在此致谢！

考虑以 1 为根，进行重链剖分，并使用线段树维护 dfs 序。

若暴力修改，每次都需要在线段树上进行 $\deg_x$ 次单点修改，难以接受。

注意到，对于一条重链上，任意一个非链顶节点，其父亲的重儿子必为该节点。

这启发我们，每次修改就只需要对父亲 Fa_x 与重儿子 son_x 进行单点修改。所有非链顶节点都可以正确更新，**唯独链顶节点不能正确更新。**

不难发现，单点查询可以轻松 $O(1)$ 维护。故我们只需要在路径查询的过程中，对链顶节点额外单点查询即可。

时间复杂度 $O(n\log^2 n)$。

注：本题还存在一些 $O(n\sqrt{n\log n})$ 的分块思想类算法，但在本题时限下难以通过。

L. 子集

题目大意

- **请注意本题特殊的空间限制：2 MB。**
- 给出一个长度为 n 的非负整数序列 a，你可以在这些数中选取任意个数（可以是零个），但不能选取相邻的数，求选出来的数的异或和最大值。
- 数据范围：$1 \leq n \leq 50$，$0 \leq a_i \leq 10^{18}$。

L. 子集 – 解法一：搜索 + 线性基

考虑从左往右，贪心地将数字插入线性基中。

首先 a_1, a_2 必须要选其中一个插入线性基。由于多插入一个数字，一定不会使得线性基能表示的集合减少，因此插入的相邻两个数的位置间隔必为 1 或 2（若间隔超过 2，将中间的数字插入线性基一定不劣）。

这样做会算重，但本题只是求异或最大值，故没有影响。

设 $g(i)$ 表示：考虑到了 $a_1 \sim a_i$，且 a_i 必须被插入时，这样的线性基个数。则有

$$g(i) = g(i-2) + g(i-3)$$

直接进行搜索，要枚举的线性基个数为 $g(n) + g(n-1)$。

当 $n = 50$ 时，$g(n) + g(n-1) = 1221537$，可以接受。

L. 子集 – 解法二：折半搜索 $+$ 0/1 trie

考虑折半搜索，对前半段搜索的结果插入 0/1 trie，对后半段搜索的结果放入 0/1 trie 中查询。

设 $f(i)$ 表示：从 $a_1 \sim a_i$ 中选取任意个数，但不能选取相邻数时的方案数。则有

$$f(i) = f(i-1) + f(i-2)$$

但难以通过本题 2 MB 的空间限制。实现较好的调段长 $+$ 卡空间做法或许可以通过。

K. 10010

题目大意

- 对于一个二进制数 $x(x>0)$，设 $2^y = \mathrm{lowbit}(x)$，设 $z = \lfloor x/2^{y+1} \rfloor$。
  定义 f 函数：$f(0)=0$，当 $x>0$ 时

$$f(x) = \begin{cases} y & z = 0 \\ f(z)+2 & z \neq 0 \wedge \mathsf{lowbit}(z) = \mathsf{lowbit}(x) \times 2 \\ y & z \neq 0 \wedge \mathsf{lowbit}(z) \neq \mathsf{lowbit}(x) \times 2 \end{cases}$$

- 给出一个长度为 n 的 01 序列 a。有 m 次操作，每次操作都形如以下的两种：
  - `1 l r`：查询序列 a 的区间 $[l,r]$ 组成的二进制数 x 的 $f(x)$ 值。
  - `2 x`：令 a_x 反转（0 变成 1，1 变成 0）。
- 数据范围：$1 \le n \le 5.1 \times 10^5$，$1 \le m \le 5 \times 10^5$。

K. 10010 – 解法一：线段树

相当于是要求：从右到左，最远能找出多少个 1，使得相邻 1 之间的间隔，构成一个公差为 1 的等差数列。

(假设查询的是区间 $[l, r]$，则要在位置 $r + 1$ 上补一个 1)

注意到信息可以合并。具体地，对一个区间维护：

- `len`：区间长度。
- `le`：区间前缀 0 的个数。
- `rg`：区间后缀 0 的个数。
- `cnt`：区间 1 的个数。
- `over`：区间答案是否完结。
- `st`，`ed`：区间答案能被表示成一个左端 st 右端 ed 公差为 1 的等差数列。

合并时，先考虑右区间答案是否完结；再考虑横跨中点的间隔能否把左右区间接起来；再考虑左区间答案是否完结。

时间复杂度 $O(n \log n)$。

K. 10010 – 解法二：树状数组倍增 + 哈希

将原序列翻转。假设已经对序列“$10100100010000\cdots$”进行哈希的预处理，则回答询问时只需二分哈希即可。在有单点修改的情况下，也只需使用树状数组倍增 or 线段树上二分即可。

该序列的长度是 n^2 级别的，但我们只需要对于其中 $O(n)$ 个值为 1 的位置进行预处理即可。

在进行比较哈希值的时候，可能需要处理基数 base 的若干次幂。n^2 级别的指数都与哈希预处理时 1 的下标有关，也可以预处理。

时间复杂度 $O(n \log n)$。

注：本题数据并没有对哈希进行 hack，请自行在以后的问题中处理哈希冲突。

C. 图上的数

题目大意

- 给出一个 n 个点 m 条边的 DAG，第 i 条边有一个权重 $p_i \in [0,1]$。
- 初始时，你有一个空的二进制数字（长度 $\text{len} = 0$），每当经过第 i 条边时：
  - 你有 p_i 的概率获得 1，有 $1 - p_i$ 的概率获得 0。
  - 获得的数字将会被均匀随机地插入到 $\text{len} + 1$ 个空隙位置的其中一个上，然后令 $\text{len} \leftarrow \text{len} + 1$。
- 你可以任取一个起点和一个终点，自己确定一条从起点到终点的路径。求经过这条路径后，收获数字的最大期望（答案对 998244353 取模）。
- 需要注意的是，你需要预先选择路径后再行动，而不能在行动时决策下一步的行动。
- 数据范围：$1 \le n \le 10^5$，$1 \le m \le 4 \times 10^5$。

C. 图上的数

当确定一条长度为 k 的路径，经过边的概率依次为 $p_1, \cdots, p_k$。不难发现，随机插入的过程等价于：先按照概率随机生成 k 个值（0 或 1），再将这些值均匀随机地插入 k 个位置。故该路径最终生成的二进制数的期望为

$$\frac{\left(\sum_{i=1}^{k} p_i\right)(2^k - 1)}{k}$$

朴素的 dp 做法，就是对每个点 u 维护所有深度的最长链。然后再对不同深度的最大期望进行比较，取最优值。

进一步，可以想到对每个点 u 维护前 30 大深度的最长链。但这样忽略了前 30 大深度最长链出现 $\sum p_i = 0$ 的情况，此时可能会出现更短的路径具有更大的期望。

再进一步，可以想到对每个点 u 维护最大深度、最大非零深度、最大非零深度 $\sim$ 最大非零深度 -29 的最长链。

C. 图上的数

对于比较函数：先将分母移项，设当前要比较的是

$$A = x \cdot (2^a - 1), \qquad B = y \cdot (2^b - 1)$$

不妨设 $a > b$，则有

$$A - B = (x \cdot 2^{a-b} - y) \cdot 2^b - (x - y)$$

令 $z = x \cdot 2^{a-b} - y$：

- 当 $z \geq 0$ 时，可以证明 $A > B$。
- 当 $z < 0$ 时，可以进一步判断 $z \cdot 2^b - (x - y)$ 的正负。

整个判断过程需要使用 `__int128` 来防止溢出。
时间复杂度 $O((n + m) \log V)$，其中 V 代表最长链的值域。
注：本题数据较弱，比较函数写得不正确也有可能通过；
如果使用 `std::map` 维护 dp 值有可能会 TLE。

G. 计数

题目大意

- 给定一个长度为 n 的正整数序列 $a_1, a_2, \cdots, a_n$，以及一个正整数 R。请你求出，有多少种长度为 n 的正整数序列 $b_1, b_2, \cdots, b_n$，满足：
  - 对于任意 $1 \leq i \leq n$，有 $a_i \leq b_i \leq R$。
  - 对于任意 $1 \leq i < n$，有 $b_i \geq b_{i+1}$。

  答案对 $10^9 + 7$ 取模。
- 数据范围：$1 \leq n \leq 5 \times 10^3$，$1 \leq a_i \leq R \leq 10^9$。

G. 计数

设 M_i 表示 $a_i \sim a_n$ 的最大值。不难发现，可以将 a_i 替换成 M_i。

设 $F_i(x)$ 表示：已经考虑了 $b_i \sim b_n$，且 $b_i = x$ 时的方案数。显然有递推

$$F_i(x) = \sum_{y=M_{i+1}}^{x} F_{i+1}(y)$$

注意到 $F_i(x)$ 相当于是 $F_{i+1}(x)$ 的前缀和。不难发现 $F_i(x)$ 是关于 x 的 $n-i$ 次多项式。

考虑将 $F_i(x)$ 写成“二项式系数基”的形式

$$F_i(x) = \sum_{k=0}^{n-i} c_{i,k} \binom{x}{k}$$

G. 计数

代入递推式，得

$$
\begin{aligned}
F_i(x) &= \sum_{y=M_{i+1}}^{x} \sum_{k=0}^{n-i-1} c_{i+1,k} \binom{y}{k} \\
&= \sum_{k=0}^{n-i-1} c_{i+1,k} \sum_{y=M_{i+1}}^{x} \binom{y}{k} \\
&= \sum_{k=0}^{n-i-1} c_{i+1,k} \cdot \left[\binom{x+1}{k+1} - \binom{M_{i+1}}{k+1} \right] \\
&= \sum_{k=0}^{n-i-1} c_{i+1,k} \cdot \left[\binom{x}{k+1} + \binom{x}{k} - \binom{M_{i+1}}{k+1} \right]
\end{aligned}
$$

故可以 $O(n)$ 从 $F_{i+1}(x)$ 的系数得到 $F_i(x)$ 的系数。
在计算答案时，可以令 $a_0 = R$。最后 $F_0(R)$ 即为答案。
时间复杂度 $O(n^2)$。

J. 子序列

题目大意

- 给出一个长度为 k 的模式序列 t，以及 m 种数字 $1 \sim m$ 以及它们各自必须出现的次数 $c_1, c_2, \cdots, c_m$。
- 你需要构造一个长度为 $n = \sum c_i$ 的序列 s，满足每个数字 i 的出现次数为 c_i。你需要使得序列 s 含有子序列 t 的数量最多，在此基础上使得 s 的字典序最小。
- 数据范围：$1 \leq m, k \leq 10^5, 1 \leq \sum c_i \leq 10^6$。

J. 子序列

首先，若子序列 t 的最大数量为 0，则只需从小到大排序 s 即可。

否则，先考虑如何分配 t 中出现过的数字，使得子序列 t 的出现次数最大。对于 t 的每一个相同数字极长连续段，我们也对 s 进行相同的分段，每一段初始先分配 t 中该段的长度个数字。

然后考虑一个数字一个数字地加入。设某一段数字在 s,t 中的长度分别为 p,q，现在要将 $p \leftarrow p+1$，对子序列 t 出现次数的影响为

$$\begin{aligned}\frac{\binom{p+1}{q}}{\binom{p}{q}} &= \frac{p+1}{p+1-q} \\ &= 1 + \frac{q}{p+1-q}\end{aligned}$$

发现该比值随着 p 的增大而减小。

J. 子序列

由于 n 较小，可以使用优先队列，每次贪心地取出最大比值加入。n 较大时，可以二分比值。

到最后会剩余一些可扩增段，这些段加入一个数带来的影响相同，此时我们就需要考虑字典序的问题。

首先先将 t 中没有出现过的部分，与已加入部分进行合并(类似一个归并的过程，每次选择开头字典序较小的部分加入)。

此时发现，对于所有的连续段，每一段的数字已经确定，长度不确定。

从左到右，依次考虑每一个可扩增段，将该段数字与下一段数字进行比较。若更小，则应该尽量扩增；若更大，则应该尽量不扩增。

时间复杂度 $\mathcal{O}(n \log m)$。

A. 骰子

题目大意

- 求有多少种给两个正 n 面体骰子 A, B 的每个面分配正整数的方案，使得掷出的点数之和与两个标准骰子（各个面依次标上 $1 \sim n$）掷出的点数之和得到的分布列相同。形如：

点数之和	2	3	$\dots$	$2n-1$	$2n$
概率	$1/n^2$	$2/n^2$	$\dots$	$2/n^2$	$1/n^2$

- 不要求点数不同，也不要求不超过 n。骰子 A, B 是有序的，不需要考虑具体把每个点数分配到哪个面上。答案对 998244353 取模。
- 数据范围：$1 \le n < 120$。

A. 骰子

前置知识：分圆多项式。

问题可以转化成，将 $\left(\frac{x^n-1}{x-1}\right)^2$ 写成两个多项式 f,g 的乘积，使得 f,g 中系数之和为 n 且所有系数非负。

有 $x^n-1=\prod_{d|n}\Phi_d(x)$，而 $\Phi_n(x)=\prod_{d|n}(x^d-1)^{\mu(\frac{n}{d})}$。

其中 $\Phi_n(x)$ 表示第 n 个分圆多项式，它是一个不可约的首一多项式。

每个分圆多项式在多项式环 $Z[x]$ 上都是不可约的。

因此将 x^n-1 分解成若干个分圆多项式后（特别地，需要扣除掉 $\Phi_1(x)=x-1$），只需要考虑每个分圆多项式在 f 中出现 $0/1/2$ 次。

理论上只需要朴素的多项式乘除，时间复杂度 $O(3^{d(n)}n^2)$。

结合一些剪枝（系数非负、系数之和为 n 即 $f(1)=g(1)=n$）可以跑的比较快。

E. 循环位移

题目大意

- 给出一个长度为 n 的随机排列 a，你需要将这个排列排序为 $1 \sim n$ 的升序排列。
- 在操作开始前，你可以指定一个操作参数 $x(2 \leq x \leq n)$，同时 x 不能超过 1.9×10^3。
- 你可以进行不超过 1.9×10^6 次操作，每次操作你都可以选择一个长度为 x 的子区间，并将该区间**向左循环位移一位**。
- 数据范围：$1 \leq n \leq 1.9 \times 10^4$。

E. 循环位移

观察操作的性质，不难发现一次操作会将区间内首个数字向后移动 $x-1$ 位，并将剩下的数字向前平移一位。

可以想到一个策略：不断将一个数右移 $x-1$ 位，最后剩余部分再一位一位移过去即可，但是这样会剩下 $1\sim x-1$ 无法处理。

考虑引入一个调整的位置，在长度为 $x+1$ 的序列中考虑问题。

先尝试交换开头的两个数。当 x 为偶数时，可以用 $x+1$ 次操作交换开头两个数：先以 2 为左端点操作一次，然后以 $2,1$ 为左端点交替操作共 x 次。

E. 循环位移

正确性证明：

- 对于 1，其会被操作 $x-1$ 次（前两次操作都不会涉及到），最后移动到第二个位置。
- 对于 2，第一次操作后会移动到 $x+1$ 处，此后 x 次操作都会涉及，最后移动到开头。
- 对于 $3\sim x+1$ 中的奇数，其会被操作 x 次。而其不会被移动到 1，因此位置不变。
- 对于 $3\sim x+1$ 中的偶数，其会被操作 x 次。而其不会被移动到 $x+1$，因此位置不变。

接下来尝试交换 p_1,p_i 且不打乱其他数，只要将 i 先循环位移到 2 处，交换完开头两个数后再将后面的复原即可。

由此可以按顺序归位 $1,2,\cdots,x-1$，每次归位 i 时在 $[i,i+x]$ 区间内操作即可。

期望操作次数估计为 $\frac{1}{2}\left(\sum_{i=x}^{n}(\frac{i}{x}+x)\right)+(x-2)(2x+1)$，枚举 $x=2,4,\cdots$，取最优的 x 即可。

注：无脑取 $x=\sqrt{n}$ 将难以通过此题。

D. 子串的故事（2）

题目大意

- 给出一个长度为 n 的字符串 S，有一个初始为空的字符串可重集 T。
- 有 m 次操作，每次操作给出两个正整数 $l, r (1 \le l \le r \le n)$，你需要将字符串 $S[l:r]$ 的**所有前缀**都加入集合 T。每次操作过后，设 T 包含字符串 $s_1, s_2, \cdots, s_k$，你需要求出

$$\sum_{1 \le i < j \le k} \mathrm{LCP}(s_i, s_j)$$

  答案对 $10^9 + 7$ 取模。
- 数据范围：$1 \le n, m \le 10^5$。

D. 子串的故事（2）

一个比较暴力的想法是，将字符串 S 的所有后缀都插入一个 trie 中（即后缀 trie）。此时对于 S 的两个子串，设它们在 trie 上的节点分别是 p, q，则它们的 LCP 长度为 $\mathrm{dep}[\mathrm{LCA}(p, q)]$，即为“根到 p”与“根到 q”两条路径的交集大小。

建立一个计数数组 a，考虑统计两个串的 LCP：

- 令“根到 p”上每一个点 x 的 $a_x \leftarrow a_x + 1$。
- 查询“根到 q”上每一个点 x 构成的 $\sum a_x$。

但本题是要将一个子串 $S[l:r]$ 的所有前缀插入，也可以直接在 a 数组上进行讨论：

- 令“根到 u”上每一个点 x 的 $a_x \leftarrow a_x + (\mathrm{dep}[u] + 1 - \mathrm{dep}[x])$。
- 查询“根到 u”上每一个点 x 构成的 $\sum a_x \cdot (\mathrm{dep}[u] + 1 - \mathrm{dep}[x])$。
  (子串 $S[l:r]$ 内部两两 LCP 的贡献还要特殊计算)

D. 子串的故事（2）

上述统计过程，显然可以拆贡献后使用重链剖分加速。大概要维护区间 $0/1/2$ 次项之和。

但后缀 trie 的节点个数为 $O(n^2)$ 级别。可以使用反串 SAM 的 parent 树的结构来代替后缀 trie。**相当于是将后缀 trie 进行高度压缩。**

要找到 $S[l:r]$ 在 SAM 上的状态（子串定位），可以在 parent 树上进行树上倍增。

仍然沿用上述思路。对于 SAM 上的每个节点，维护深度在 $\mathrm{minl} \sim \mathrm{maxl}$ 的 a 信息。但注意到，询问串在 SAM 上的状态有可能不完全覆盖 $\mathrm{minl} \sim \mathrm{maxl}$（为该区间的一个前缀）。

如何处理这一点，就延伸出了离线做法与在线做法。

D. 子串的故事（2）– 离线做法

离线将所有询问的子串在 SAM 上的状态找出来，然后新建关于询问串的虚拟节点。

此时，所有询问串一定完整地覆盖从当前状态到根的路径。沿用上述做法即可。

总节点数不超过 $2n+m$。

设 n, m 同阶，时间复杂度 $O(n\log^2 n)$。

D. 子串的故事（2）– 在线做法

仍然沿用上述做法，只不过不完全覆盖时多算 or 少算的贡献，还要进行打补丁。

进一步地大分讨、拆贡献即可...

设 n, m 同阶，时间复杂度 $O(n \log^2 n)$。

当然，你可以选择使用 LCT 新建节点，以达成在线的目的。

Thank you!