题目讲解

K. 乘二

- 给定 n 个数 a_i，需要操作 k 次，每次操作将一个数乘二
- 要求操作 k 次之后，所有数的和最小。由于答案很大，只需输出对 998244353 取模的结果。
- $n \leq 2 \times 10^5, k \leq 10^9$

K. 乘二

- 显然每次操作会选择最小的一个数乘 2，这样模拟是 $O(k\log n)$ 的。但是我们其实可以只模拟到所有数都在 $[2^{29},2^{30})$ 之间的时候。这样的话一定就是每 n 轮一循环，每次循环从小到大乘所有数，快速模拟一下即可。

- 时间复杂度 $O(n\log a)$。

D. 树

- 给定一棵树，以及树上的 n 个点。
- 对于一趟行程 a，小 d 会从 a_1 出发，依次到达$a_2, \cdots, a_m$（从一个点到另一个点会依次经过链上所有点）。
- 给定序列 b，问最短的 a 的长度，使得：b 是行程 a 的子序列。

- 可以对 b 单点修改。

D. 树

- 性质：序列 a 一定是 b 的子序列。
- 基于此，我们可以设计一个贪心：每次找 b 的最远的下一个节点，使得方案合法。
- 容易发现，这个贪心结果只和多少相邻的三元组不共链有关。
- 因此我们只需要对每个相邻的三元组判定是否共链即可。

G. 后继

- 给定一个长度为 n 的序列 a，互不相同，还有一个长度为 n 的序列 b，满足 $b_i = a_i \operatorname{xor} x$，$x$ 是一个未知常量。
- 现在你可以向交互库若干次发问，每次询问一个整数 x，表示询问 b_x 的后继的下标。
- 在进行完所有询问后，你需要输出一个整数 x_0，满足对于长度为 n 的序列 c，$c_i = a_i \operatorname{xor} x_0$，$c$ 与 b 偏序关系相同，且 x_0 最小。
- m 组数据，每次的 n 和序列 a 不变。

G. 后继

- 我们把序列 a 放到 01 Trie 上后，我们考虑由低位至高位逐一确定 x。
- 假设我们需要确定 x 的第 p 位，容易发现，如果这一层 Trie 所有节点只有一个儿子，那么 x 这位是多少不影响答案。不然我们考虑其中一个有两个节点的儿子：
- 假设 x 第 p 位是 0，那么应当该节点左儿子元素在异或后均小于右子树，故可以拿出左儿子在异或后的最大值询问后继，来判断这位是 0 还是 1。
- 使用 01 Trie 优化求解，时间复杂度 $O(n\log a_i + m\log^2 a_i)$。

E. 骰子

- 有 n 枚 $m+1$ 面的骰子，$m+1$ 个面上分别写着 $0\sim m$ 这 $m+1$ 个数字，丢出第 i 枚骰子写着数字 j 的面朝上的概率永远是 $p_{i,j}$。
- 你打算投一些骰子，把每个骰子向上的面写的数字加起来，设总和是 sum，如果 $sum\le m$，你会得到 b_{sum} 的开心度。
- 你打算投 q 次骰子，第 i 次丢出编号在 $[l_i,r_i]$ 内的骰子，请问每一轮你得到的期望开心度是多少？答案对 998244353 取模。
- $n\le 1500, m\le 200, q\le 6\times 10^5$

E. 骰子

- 设多项式 $F_i = \sum_{j=0}^{m} p_{i,j}\, x^j$，一次询问 $[l,r]$ 的答案可以发现是算出 $\prod_{i=l}^{r} F_i$ 这个多项式，然后要求它和给定向量 b 点乘的结果，注意向量 b 只有 $0 \sim m$ 项有值。
- 先考虑所有 F_i 常数项在模意义下不为 0 的情况，那么任意一个 F_i 都可以求逆。因为向量 b 只有 $0 \sim m$ 项有值，计算 $(\prod_{i=l}^{r} F_i) \bmod x^{m+1}$ 和向量 b 点乘的结果和原来的答案一致。如果对所有前缀 p 计算出 $pre_p = \left(\prod_{i=1}^{p} F_i\right) \bmod x^{m+1}$ 以及 $ipre_p = pre_p^{-1} \bmod x^{m+1}$，所求相当于是 $pre_r ipre_{l-1}$ 与向量 b 点乘的结果。

E. 骰子

- 将 b reverse 设 $b' = \sum_{i=0}^{m} x^i b_{m-i}$，那么一个多项式与向量 b 点乘的结果与这个多项式与 b' 卷积的 $[x^m]$ 系数是一样的。相当于只要计算 $pre_r ipre_{l-1} b'$ 这三个多项式相乘的 $[x^m]$ 系数。
- 我们在 $O(nm^2)$ 时间内预处理出所有 $pre_p, ipre_p$ 以及 $pre_p b'$，查询时就只需要计算 $(pre_r b')ipre_{l-1}$ 这两个 m 次多项式卷积的一项系数，这可以 $O(m)$ 求。
- 现在考虑原问题，在 $p_{i,0} = 0$ 或 $10^9 + 7$ 时上述做法会失效，因为常数项为 0 无法求逆。对此我们考虑把所有常数项模意义下为 0 的多项式不断除以 x 并记录除了多少个，设查询时 $[l, r]$ 这些多项式一共除了 s 个 x，最后查询的相当于是 $pre_r ipre_{l-1} x^s$ 与向量 b 点乘的结果，即 $(pre_r b')pre_{l-1}$ 的 $[x^{m-s}]$ 项系数，这与刚才的形式是一致的。

H. BFS 序 0

- 给定一棵 n 个点的有根树
- 多次询问，每次询问给出一个序列，问该序列能否是这棵树某一个 BFS 序的子序列
- 这里 BFS 序的定义是：维护一个 queue，每次 pop 头上的元素，把它的儿子按照某种顺序 push 进去
- 范围：$n, \sum$序列长度 $\leq 3 \times 10^5$

BFS 序 0

- 考虑一个合法 BFS 序的子序列，首先它的节点深度必须单调不降。
- 然后设 BFS 序相邻两个点是 x, y，那么若 $dep_x < dep_y$ 则没有用（因为 x 必然在 y 前面）。
- 否则 $dep_x = dep_y$，设 $z = lca(x, y)$, a, b 分别是 z 到 x, y 路径上的第一个点，那么由于 BFS 序的性质，x 在 y 前面当且仅当 a 在 z 的儿子中的次序在 b 前面。于是可以连一条 a 到 b 的有向边，表示 a 在 z 的儿子中的次序在 b 前面。
- 那么这个序列可能是某个 BFS 序的子序列当且仅当连出来的有向图没有环，也即所有点可以排列儿子顺序使得存在合法 BFS 序。
- 时间复杂度 $O(n \log n + \sum k \log n)$。

B. 组合数

- t 次询问（$t \leq 10^5$），每次给出 l,r,n,m，问有多少个整数 $i \in [l,r]$ 满足存在 $0 \leq u \leq n, 0 \leq v \leq m$ 满足 $i = C(u,v)$。这里 C 表示组合数。
- $l,r,n,m \leq 10^9$

B. 组合数

- 首先，不妨假设 $m \le 20$。这是因为首先可以只考虑 $v \le u/2$ 的 v，而 $C(40,20) > 10^9$。
- 同时 v = 0,1 时，$C(u,v)$ 可以取到 $\le n$ 的所有正整数，下面不妨认为 $l > n$。

- 此时，$v \ge 2$，所以 $C(u,v) = \Omega(u\verb|^|2)$，所以 $u \le 50000$。
- 这样，可能产生贡献的 (u,v) 只有 $O(10^{\frac{9}{2}} + 10^{\frac{9}{3}} + 10^{\frac{9}{4}} + \cdots)$ 对，总共不超过 100000 对。可以全部预处理出来，每次询问二分答案。
- 对于 $v \le m$ 的限制，可以对每个 $2 \le m \le 20$ 分别预处理出所有可能的组合数，只会让预处理复杂度乘个 20，可以接受。

A. 不要玩弄字符串

初始有个 01 串 S，以及 k 个模板串 t_i，第 i 个模板串有价值 v_i。两个玩家轮流往 S 后面加一个 0 或 1，假如加完字符后有某个 t_i 第一次成为了 S 的子串，那么加字符的那个玩家收获 v_i。

两个玩家绝顶聪明，都希望最大化自己收获减去对面收获。

当所有串的价值都被拿到，或者两名玩家都觉得继续游戏自己收益不会更大时，游戏结束。

q 次询问给定 S_i，询问当初始 $S = S_i$ 时，先手收获减后手收获。

$k \le 12$，$|S_i|, |t_i| \le 100$，$q \le 1000$。

A. 不要玩弄字符串

建出全体 t_i 的 AC 自动机。记 $f_{i,st}$ 为目前到达 AC 自动机上节点 i，且未在 S 中出现的串集合为 st 时，回合玩家减去非回合玩家的最大值。

按 st 大小从小到大转移，假如节点 i 对应的串的集合 s_i 与 st 有交，那么 $f_{i,st} \leftarrow f_{i,st \setminus s_i} - \sum_{j \in s_i \cap st} v_j$ 已确定。否则设 $f'_{i,st}$ 为所有已确定的后继状态的负值的最大值。

若一个点所有后继状态都确定，那么它就确定了，即 $f_{i,st} = f'_{i,st}$。若剩下的状态都无法确定，那么拿出 f' 值最大的未确定状态，若 > 0，确定该状态；否则，把剩下所有状态确认为 0。

A. 不要玩弄字符串

考虑正确性，假如转移形成一个强连通块，那么我们拿出使 $f'_{i,st}$ 最大的 i，若 $f'_{i,st} > 0$，考虑它的一个前驱 j，知 $f_{j,st} \geq -f'_{i,st}$，所以 $|f_{j,st}| \leq f'_{i,st}$，同理，$j$ 的任一前驱 k 也满足 $|f_{k,st}| \leq f'_{i,st}$，因为转移强连通，所以 $\forall j, |f_{j,st}| \leq f'_{i,st}$，所以 $f'_{i,st}$ 不再会被更新，即 $f_{i,st} = f'_{i,st}$。然后考虑最后如果还有没被确认的状态，那么这些状态的 f' 都小于等于 0，且这些状态之间的转移还是强连通块，如果回合玩家决策为走出这个强连通块，那么非回合玩家的收益减回合玩家的收益就会变成非负的，故双方都会希望游戏直接结束，即这些状态的 f 为 0。

当转移不为强连通时，缩点后按拓扑序类似上面讨论可以获得一样的结论。

每次查询直接在 AC 自动机上找到初始状态对应的 dp 状态，用 f 回答即可。

复杂度 $O(2^k (\sum |t_i|) (\Sigma + \log(\sum |t_i|)) + \sum |S_i|)$。

F. 保区间最小值一次回归问题

- 有一个给定的数组 $[a_1, a_2, \dots, a_n]$ 和 m 个要求。每个要求用 (l_i, r_i, v_i) 给出，表示 $\min\{a_l, a_{l+1}, \dots, a_r\} = v$（区间 $[l, r]$ 最小值必须恰好等于 v）。
- 你可以对给定的数组执行任意次修改操作，使得它符合所有要求。
- 如果把 a_i 由 x 改为 y，代价是 $|x - y|$。
- 问所有操作代价和最小是多少，无解输出 -1。
- $n, m \le 200000, 1 \le a_i, v \le 10^9$

F. 保区间最小值一次回归问题

设 b_i 为最终 a_i 的下界。换句话说，

$$b_i = \max_{l_j \le i \le r_j} v_j$$

如果数组 $[b_i]$ 都不满足要求，肯定无解。

而且，一旦要修改 a_i，必定是修改成 b_i（否则不会新满足任何限制）。

如果 $a_i < b_i$，a_i 就必须改成 b_i。以下认为 $a_i \geq b_i$。

F. 保区间最小值一次回归问题

这样，$b_i = v$ 的 a_i 只会对 v 的限制可能有贡献，所以不同的 v 之间独立。

对于固定的 v，问题变为：有一些区间 $[l,r]$，你要在 $1,2,\ldots,n$ 中选一些 $b_i = v$ 的位置修改，修改 i 的代价为 $|a_i - v|$，问使得每个区间都至少有一个点被修改的代价至少是多少。

这显然可以线段树优化 dp 解决。

时间复杂度 $O((n+m)\log n)$。

C. 放苹果

- 有 n 个盘子和 m 个苹果，现在要把每个苹果都放到 n 个盘子里面的一个，总共 n^m 情况。
- 定义一次操作为将一个苹果移动到某个相邻的盘子里。对于一种放苹果的情况，代价为最少操作的次数，使得能将所有苹果放到同一个盘子里。
- 求所有 n^m 种情况的代价的和，对 998244353 取模。
- $n, m \leq 5 * 10^5$

C. 放苹果

- 对于一种放苹果的方案，记 s_i 表示前 i 个盘子放了多少个苹果，那么最优策略一定是将所有苹果移动到 s第一次大于等于 $n/2$ 的位置。即每个苹果放入盘子标号的中位数。
- 列出式子，即有代价是 $\sum \min(s_i, n - s_i)$。
- 所以所有方案的代价和就是枚举第 i 和第 $i+1$ 个盘子，再枚举前 i 个盘子有多少个苹果，即
- $\sum_{i,j} min(j, n-j)\, C(n,j) i^j (m-i)^{n-j}$
- 交换求和号，有

C. 放苹果

- $\sum_j min(j,n-j)\,C(n,j)\sum_i i^j(m-i)^{n-j}$
- 现在对于每个 j，希望求出
- $\sum_i i^j(m-i)^{n-j} = \sum_{i,k} C(n-j,k)(-1)^k i^{j+k} m^{n-j-k} = \sum_k C(n-j,k)(-1)^k v_{j+k}$
- 其中$v_t = m^{n-t}\sum_i i\hat{}t$，
- 注意到$\sum \frac{v_t\, x^t}{t!} = m^{n-t} \times \frac{e^{mx}-1}{e^x-1}$。之后，再做一遍卷积即可。时间复杂度 $O(n \log n)$。

I. 括号序列

- 对于一个长度为 $2n$ 的合法括号串 S，我们要对它进行 n 次操作。每次操作可以是：
- 1. 从 S 中删去一个连续子串 ()。位置不同视为不同的操作。例如，()() → ()。
- 2. 从 S 中删去一个连续子串)(。位置不同视为不同的操作。但是，这里删去的)(必须一开始就在 S 中相邻！
- 例如，下面操作是合法的 ()()() -> ()() -> ()
- 下面操作不合法 ()()() -> ()() -> ()
- 显然，n 次操作后 S 会被删空。设操作的方案数为 $f(S)$。
- 对所有合法括号串 S，求 $\sum f(S)$。保证 $n \leq 200000$。对 998244353 取模。

I. 括号序列

- 第一步：找到合适的组合转化
- 考虑从正常括号序列到有根无标号树的双射，其中儿子是有序的：将括号序列视为树的深度优先遍历顺序。在这里，树有n+1 个顶点，所以我们让 n 加一。
- 然后，这两个操作可以看作是从树中移除一个叶子，或者“压缩”两个相邻的儿子。
- 这里，压缩的意思是：把两个儿子合并为一个，并合并儿子的儿子，同时保持儿子的儿子的相对顺序。由于)(必须从一开始就是相邻的才能移除它们，我们只能压缩一开始就相邻的儿子。

I. 括号序列

- 现在，如何计算固定树的移除方案数？
- 如果我们只能移除叶子，不能压缩儿子，那么它等同于拓扑排序计数。
- 同时，对于一种固定的压缩儿子（i.e. 对于每个结点，已经把他的儿子分为了若干区间，强制恰好要把这些区间分别压缩成一个点）的方式，(移除叶子+压缩儿子)的方法数也可以看作是一棵新树上的拓扑排序：若 x 的 k 个儿子被压缩成一个儿子，则给 x 增加 $k-1$ 个叶子儿子。（删除它们分别代表把第 1,2， 2,3， ... 个儿子合并）
- 注意，压缩不会改变顶点的数量，因为 k 个儿子合并为 1 个儿子后会新出现 $k-1$ 个叶子。因此，最终的树（考虑到由于压缩操作而出现的顶点）始终有 n 个顶点。
- 根据树上拓扑序计数的结论，我们只需要找到所有（树，压缩）二元组的新树的 $\prod 1/(size_i)$ 之和，再乘以 $n!$ 输出。

I. 括号序列

- 第二步：列式
- 考虑以下两个生成函数：
- F：[x^n]F 是所有具有 n 个顶点的树的答案。
- G：[x^n]G 是所有具有 n 个顶点的“压缩森林”的所需总和。也即，有一些 已知要被压缩在一起的树，这些树的答案乘积之和。
- 此时有：

$$f(n) = \frac{1}{n} \sum_{x_1+\cdots+x_k=n-1} \prod g(x_i)$$

$$g(n) = \frac{1}{n} \sum_{x_1+\cdots+x_k=n} \prod x_i f(x_i)$$

$$g(0) = 0$$

I. 括号序列

- 第二步：列式
- 令 $h(n) = nf(n)$。
- 用生成函数的语言来说，就是：

$$H = \frac{x}{1-G}$$

$$xG' = \frac{1}{1-H} - 1$$

I. 括号序列

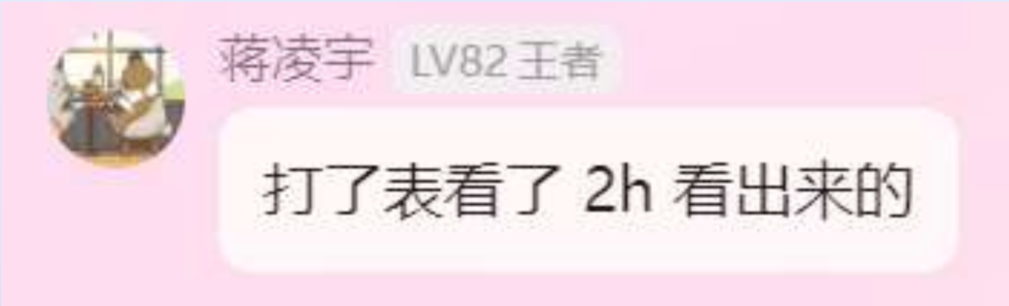
- 可以直接分治 FFT 算答案。
- 时间复杂度 $O(n\log^2 n)$（或更低）。

J.卡牌游戏

- A 和 B 在玩卡牌游戏，每个人有若干张牌，每张牌上标有一个数字。记它们拥有标有数字 $i(1 \leq i \leq n)$ 的牌的数量分别为 A_i 和 B_i。
- 游戏分为若干轮，每轮游戏中双方玩家轮流出牌，第一轮由 A 开始，之后由前一轮的胜者开始。每一轮中，开始的玩家任意出一张牌，之后必须出一样的牌或是放弃这一轮，放弃则对手赢得这一轮。
- 先出完所有牌的玩家获胜。
- 给定 A, B，判断谁会获胜。

J.卡牌游戏

- 设 $C_i = \min(A_i, B_i)$，当 $A_i > B_i, A_i = B_i, A_i < B_i$ 时分别称其为 $+, =, -$。我们认为有无穷多个值为 0 的 $=$。
- 注意到获胜的条件如下（可以用归纳证明）：
- 1. 若存在一个 $-$ 是最大值则 A 获胜。
- 2. 否则，如果最大值不唯一，则 B 获胜。
- 3. 否则，如果最大值是 $=$，则 A 获胜。
- 4. 否则，如果存在两个大于最大的 $-$，则 B 获胜。
- 5. 否则 A 获胜。



J.卡牌游戏

- 具体的证明如下（按照剩余牌数归纳）：
- 为了避免 Corner Case，认为 $\sum B = 0$ 的局面是 B 获胜，$\sum B \neq 0$ 但 $\sum A = 0$ 的局面是 A 获胜，可以验证这些局面符合条件（只有 Case 1/2），作为归纳的根基。
- 引理：A 可以在时刻保持先手的情况下，去掉所有的 +（变成 =），其 C_i 不变或变小（但最终的值 A 无法决定）。
- （注：这个引理导致 B 获得先手时不可能有 –。）

J.卡牌游戏

- Case 1：存在一个 – 是最大值。A 可以首先去掉所有的 +，然后不断出非最大值的牌，最后再出这个最大的 – 。如果 B 抢到先手，要么最大值不唯一，根据 Case 2 A 获胜，要么最大值唯一且对于 B 来说是 +（且对于 B 来说没有 –），根据 Case 4 A 获胜。
- Case 2：最大值不唯一，且不是 –。B 可以一直放弃，直到 A 出牌导致最大值变成唯一。然后 B 一直接牌一定可以抢到先手，根据 Case 1/3 B 获胜。

J.卡牌游戏

- Case 3：最大值唯一且为 =。A 可以首先去掉所有的 +，然后不断出不断出这个最大值，如果减到次大值则放弃，根据 Case 2 A 获胜，否则是 B 某一步放弃了，根据 Case 1 A 获胜。
- Case 4：最大值唯一且为 +，次大值不是 –。
  - 如果 A 出次大值，B 一定能抢到先手，根据 Case 1 B 获胜。
  - 如果 A 出最大值，B 在减到次大值时放弃，根据 Case 2 B 获胜。如果过程中 A 放弃，则仍然是 Case 4 B 获胜。
- Case 5：最大值唯一且是 +，存在一个 – 是次大值。A 可以首先去掉所有的 +，然后要么 – 变成最大值，根据 Case 1 A 获胜，要么最大值唯一且是 =，根据 Case 3 A 获胜。