杂题选讲（数据结构）

RpUtl

简介

在题单中随机选择了 12 道题目中讲一讲。
选择了一些我认为比较经典的，常见的，或者小清新的 DS。
课件制作感谢 To_Carpe_Diem 和豆包的指导。

题目 1 - 题面

给定一个 2^n 个集合（用二进制位给出）和每个集合的初始值 v_S，若干次操作：

1. 求某个集合的子集和。
2. 改变某个集合的值。

数据范围：$1 \le n \le 20, q \le 3 \times 10^5$。

1000 ms，512 MB。

题目 1 - 解法

我们有两种做法：

1. 直接用数组存每个集合的值，修改 $O(1)$，查询为 $O(2^n)$。
2. 直接求高维前缀和，每次修改就重新求，修改 $O(2^n)$，查询 $O(1)$。

题目 1 - 解法

我们有两种做法：

1. 直接用数组存每个集合的值，修改 $O(1)$，查询为 $O(2^n)$。
2. 直接求高维前缀和，每次修改就重新求，修改 $O(2^n)$，查询 $O(1)$。

可以综合一下？设 $s_{i,j}$ 表示满足集合前 $n/2$ 位为 i，后 $n/2$ 位为 j 子集的所有集合的和。修改只需要枚举 j，查询只需要枚举 i，预处理出来 $s_{i,j}$。复杂度为 $O(q2^{n/2})$。

题目 1 - 解法

我们有两种做法：

1. 直接用数组存每个集合的值，修改 $O(1)$，查询为 $O(2^n)$。
2. 直接求高维前缀和，每次修改就重新求，修改 $O(2^n)$，查询 $O(1)$。

可以综合一下？设 $s_{i,j}$ 表示满足集合前 $n/2$ 位为 i，后 $n/2$ 位为 j 子集的所有集合的和。修改只需要枚举 j，查询只需要枚举 i，预处理出来 $s_{i,j}$。复杂度为 $O(q2^{n/2})$。

对于神秘位运算的数据结构，预处理后几位枚举前几位也是一种比较好的处理方式。

题目 2 - 题面

给你 n 个区间 $[l_i, r_i], 1 \le i \le n$，定义函数 $f_i(x)$ 为：

$$f_i(x) = \begin{cases} f_{i-1}(x) + [l_i \le f_{i-1}(x) \le r_i] & i \neq 0 \\ x & i = 0 \end{cases}$$

q 次询问，查询 $f_n(v)$ 的值，其中 v 给出。

数据范围：$1 \le l_i \le r_i \le 5 \times 10^5, 1 \le n, q, v \le 5 \times 10^5$。

4000 ms,1 GB。

题目 2 - 解法

这应该是这一类题中最简单的一个了。

考虑维护一个数组 $g_{i,j} = f_i(j)$，我们希望通过一些操作使得 $g_{i,j}$ 变成 $g_{i+1,j}$，手玩和感性理解可以发现，$g_{i,j}$ 关于第二维 j 一定是单调的，知道第一个 $g_{i,p} \geq l_i$ 和最后一个 $g_{i,q} \leq r_i$，只需要进行 $[p, q]$ 的区间加操作即可推到 $g_{i+1,j}$。

题目 2 - 解法

这应该是这一类题中最简单的一个了。

考虑维护一个数组 $g_{i,j} = f_i(j)$，我们希望通过一些操作使得 $g_{i,j}$ 变成 $g_{i+1,j}$，手玩和感性理解可以发现，$g_{i,j}$ 关于第二维 j 一定是单调的，知道第一个 $g_{i,p} \geq l_i$ 和最后一个 $g_{i,q} \leq r_i$，只需要进行 $[p, q]$ 的区间加操作即可推到 $g_{i+1,j}$。

实际上我们不需要真的维护一个二维数组，而是像滚动数组一样滚掉第一维，拿线段树维护第二维，需要支持区间加和全局找第一个 $\geq x$ 的位置，懒标记和线段树二分可以解决。

最后单点查询即可，复杂度可以 $O(n \log n)$。

题目 2 - 解法

这应该是这一类题中最简单的一个了。

考虑维护一个数组 $g_{i,j} = f_i(j)$，我们希望通过一些操作使得 $g_{i,j}$ 变成 $g_{i+1,j}$，手玩和感性理解可以发现，$g_{i,j}$ 关于第二维 j 一定是单调的，知道第一个 $g_{i,p} \geq l_i$ 和最后一个 $g_{i,q} \leq r_i$，只需要进行 $[p, q]$ 的区间加操作即可推到 $g_{i+1,j}$。

实际上我们不需要真的维护一个二维数组，而是像滚动数组一样滚掉第一维，拿线段树维护第二维，需要支持区间加和全局找第一个 $\geq x$ 的位置，懒标记和线段树二分可以解决。

最后单点查询即可，复杂度可以 $O(n \log n)$。

这是一类很经典的问题，拓展一下：如果每次询问都是在一个区间上进行该如何做。

题目 3 - 题面

给出一个 1 到 n 的排列，现在对这个排列序列进行 m 次局部排序，排序分为两种：对下标在区间 $[l, r]$ 内的数升序或者降序排序。

最后单次询问第 q 位置上的数字。

$1 \leq q \leq n \leq 10^5$。

4000 ms,256 MB。

题目 3 - 解法

这个题真的是线段树必做题啊，太经典了。

区间排序是几乎做不到的，但是当值的种类很少的时候，可以用赋值代替排序。

题目 3 - 解法

这个题真的是线段树必做题啊，太经典了。

区间排序是几乎做不到的，但是当值的种类很少的时候，可以用赋值代替排序。

注意到只有一次询问，考虑非常经典的二分答案，二分 p_q 的值 v，将初始排列中 $\geq v$ 的值赋值为 1，剩下的赋值为 0，然后做 m 次赋值，最终查询 q 的位置为 1 还是 0 来判断二分的值是大了还是小了。

题目 3 - 解法

这个题真的是线段树必做题啊，太经典了。

区间排序是几乎做不到的，但是当值的种类很少的时候，可以用赋值代替排序。

注意到只有一次询问，考虑非常经典的二分答案，二分 p_q 的值 v，将初始排列中 $\geq v$ 的值赋值为 1，剩下的赋值为 0，然后做 m 次赋值，最终查询 q 的位置为 1 还是 0 来判断二分的值是大了还是小了。

只需要简单的二分和普通线段树即可，复杂度为 $O(n\log^2 n)$。

对于一些答案具有单调性的问题，考虑把权值转化为 01 序列可能会更好处理。

题目 4 - 题面

给出一个长度为 n 的非负数列 a，q 次询问 b, x, l, r，求

$$\max_{j \in [l,r]} b \oplus (a_j + x)$$

其中 $\oplus$ 表示按位异或运算。

$1 \le n, q \le 10^5, 0 \le a_i < 2^{20}$。

3000 ms,256 MB。

题目 4 - 解法

从高到底去依次确定答案的每一位，假设目前已经确定了答案的前 k 位，要确定第 $k+1$ 位，我们假设这一位是 1。

因为知道了答案和 b 的前 $k+1$ 位，就可以得出 a_j+x 的前 $k+1$ 位了，设为 v。

题目 4 - 解法

从高到底去依次确定答案的每一位，假设目前已经确定了答案的前 k 位，要确定第 $k+1$ 位，我们假设这一位是 1。

因为知道了答案和 b 的前 $k+1$ 位，就可以得出 a_j+x 的前 $k+1$ 位了，设为 v。

实际上，因为只关心假设是否成立，即区间内是否存在 a_j 满足条件，而 a_j+x 的前 $k+1$ 位为 v 这个条件可以转化为对于 a_j 在值域上的区间限制，问题转化为了是否存在下标在 $[l,r]$，值域在 $[u,v]$ 的 a_j 了。

这是一个简单的二维数点问题，主席树即可解决，如果假设成立则答案第 $k+1$ 为 1，反之为 0，从高往低去推，复杂度为 $O(n\log n\log V)$。

题目 4 - 解法

从高到底去依次确定答案的每一位，假设目前已经确定了答案的前 k 位，要确定第 $k+1$ 位，我们假设这一位是 1。

因为知道了答案和 b 的前 $k+1$ 位，就可以得出 a_j+x 的前 $k+1$ 位了，设为 v。

实际上，因为只关心假设是否成立，即区间内是否存在 a_j 满足条件，而 a_j+x 的前 $k+1$ 位为 v 这个条件可以转化为对于 a_j 在值域上的区间限制，问题转化为了是否存在下标在 $[l,r]$，值域在 $[u,v]$ 的 a_j 了。

这是一个简单的二维数点问题，主席树即可解决，如果假设成立则答案第 $k+1$ 为 1，反之为 0，从高往低去推，复杂度为 $O(n\log n\log V)$。

对于这种位运算最优化，可以拆位考虑，转化到连续值域上的限制。

题目 5 - 题面

给你 n 个物品，每个物品有体积 v_i 和收益 w_i，q 次询问，每次给出 $[l, r]$ 和 m，问在第 $l \sim r$ 个物品上做总体积为 m 的背包，收获的最大收益是。

$1 \leq n \leq 4 \times 10^4, q \leq 1 \times 10^5, 0 \leq w_i, v_i, m \leq 200$。

2000 ms,128 MB。

题目 5 - 解法

一些有关背包的基础知识：支持 $O(m)$ 增加一个物品，可以 $O(m^2)$ 合并两个物品。

题目 5 - 解法

一些有关背包的基础知识：支持 $O(m)$ 增加一个物品，可以 $O(m^2)$ 合并两个物品。

实际上，对于两组物品最终得到两个背包数组 F, G，想要把两个背包合并后，查询体积为 x 的最大收益不必合并背包，直接计算 $\max(F_i + G_{x-i})$ 即可。因为背包的本质是卷积。

题目 5 - 解法

一些有关背包的基础知识：支持 $O(m)$ 增加一个物品，可以 $O(m^2)$ 合并两个物品。

实际上，对于两组物品最终得到两个背包数组 F, G，想要把两个背包合并后，查询体积为 x 的最大收益不必合并背包，直接计算 $\max(F_i + G_{x-i})$ 即可。因为背包的本质是卷积。

如果询问的区间必然经过序列中点，其实可以在前一半序列求一个后缀背包，后一半序列求一个前缀背包，可以 $O(nm)$ 预处理，$O(m)$ 完成一个查询，

题目 5 - 解法

一些有关背包的基础知识：支持 $O(m)$ 增加一个物品，可以 $O(m^2)$ 合并两个物品。

实际上，对于两组物品最终得到两个背包数组 F, G，想要把两个背包合并后，查询体积为 x 的最大收益不必合并背包，直接计算 $\max(F_i + G_{x-i})$ 即可。因为背包的本质是卷积。

如果询问的区间必然经过序列中点，其实可以在前一半序列求一个后缀背包，后一半序列求一个前缀背包，可以 $O(nm)$ 预处理，$O(m)$ 完成一个查询，

对于不经过区间中点的询问，考虑分治，把他们扔到两边重复上述操作，知道遇到某个分治区间符合其条件，至多分治 $\log n$ 次。

最终，将询问离线，可以做到 $O(nm \log n + qm)$，实际上，可以通过猫树技巧做到 $O(nm \log n + qm)$。

题目 5 - 解法

一些有关背包的基础知识：支持 $O(m)$ 增加一个物品，可以 $O(m^2)$ 合并两个物品。

实际上，对于两组物品最终得到两个背包数组 F, G，想要把两个背包合并后，查询体积为 x 的最大收益不必合并背包，直接计算 $\max(F_i + G_{x-i})$ 即可。因为背包的本质是卷积。

如果询问的区间必然经过序列中点，其实可以在前一半序列求一个后缀背包，后一半序列求一个前缀背包，可以 $O(nm)$ 预处理，$O(m)$ 完成一个查询，

对于不经过区间中点的询问，考虑分治，把他们扔到两边重复上述操作，知道遇到某个分治区间符合其条件，至多分治 $\log n$ 次。

最终，将询问离线，可以做到 $O(nm \log n + qm)$，实际上，可以通过猫树技巧做到 $O(nm \log n + qm)$。

当一个简单的问题略微复杂，扔到区间上难以直接解决时，如果信息是可以合并的，可以考虑分治算法。

题目 6 - 题面

给你一个长度为 n 的序列 a，q 次查询，每次给出 l, r, p，求 $[l, r]$ 中所有子序列去重后的和的和 $\bmod p$ 的值。

$1 \leq n, q, a \leq 10^5, 2 \leq p \leq 10^9 + 7$。

3000 ms,512 MB。

题目 6 - 解法

这题啊，非常的精彩。

这个答案的形式不好计算，考虑把贡献拆到每个元素上，设子区间长度为 x，元素 v 出现次数为 y，则元素 v 对答案的贡献为 $(2^x - 2^{x-y})v$。

题目 6 - 解法

这题啊，非常的精彩。

这个答案的形式不好计算，考虑把贡献拆到每个元素上，设子区间长度为 x，元素 v 出现次数为 y，则元素 v 对答案的贡献为 $(2^x - 2^{x-y})v$。

这个形式还是不容易统计答案，考虑莫队，并求出区间中出现次数为 y 的元素之和 $\sum v$，并将 $(2^x - 2^{x-y}) \sum v$ 贡献到答案。

题目 6 - 解法

这题啊，非常的精彩。

这个答案的形式不好计算，考虑把贡献拆到每个元素上，设子区间长度为 x，元素 v 出现次数为 y，则元素 v 对答案的贡献为 $(2^x - 2^{x-y})v$。

这个形式还是不容易统计答案，考虑莫队，并求出区间中出现次数为 y 的元素之和 $\sum v$，并将 $(2^x - 2^{x-y}) \sum v$ 贡献到答案。

一个巧妙的观察是所有元素出现次数之和不超过 n，即出现次数不同的元素的个数为 $\sqrt{n}$ 级别，可以用链表来维护。

题目 6 - 解法

这样处理实际上就比较复杂了，实现比较复杂，常数比较大。

一个更简单的处理方法，不难发现整个序列出现次数 $> \sqrt{n}$ 的元素不超过 $\sqrt{n}$ 种，只有这些元素才有可能在区间中的出现次数 $> \sqrt{n}$。

把这些元素预处理出来，在莫队的过程中直接维护出现次数为 x 的 v 之和 c_x，计算答案时只计算 $c_{1\sim\sqrt{n}}$，超过 $\sqrt{n}$ 的部分找预处理出来的元素。

题目 6 - 解法

这样处理实际上就比较复杂了，实现比较复杂，常数比较大。

一个更简单的处理方法，不难发现整个序列出现次数 $> \sqrt{n}$ 的元素不超过 $\sqrt{n}$ 种，只有这些元素才有可能在区间中的出现次数 $> \sqrt{n}$。

把这些元素预处理出来，在莫队的过程中直接维护出现次数为 x 的 v 之和 c_x，计算答案时只计算 $c_{1\sim\sqrt{n}}$，超过 $\sqrt{n}$ 的部分找预处理出来的元素。

注意两种做法都需要 $\sqrt{n}$ 次 2^y 查询，而模数不固定，需要 $O(\sqrt{n})$ 预处理，$O(1)$ 查询的快速幂技巧。

最终离线可以做到严格的 $O(n\sqrt{n})$ 复杂度。

题目 7 - 题面

给你一个长度为 n 的序列 v，q 次查询，每次给出 l, r，求满足 $l \le a < b < c \le r, b - a \le c - b$ 的所有三元组中，最大的 $v_a + v_b + v_c$。

$1 \le n, q \le 5 \times 10^5, 1 \le v_i \le 10^8$，每次询问保证存在合法的三元组。

2000 ms,512 MB。

题目 7 - 解法

感觉是十分 Ad-hoc 的支配对题目。

若固定了 a, b，则合法的 c 是一个后缀，可以得出的第一个结论就是，最优的 a, b 一定满足 b 是 $[a+1, b]$ 中最大的。否则可以将 b 移动到更大的，一定更优。

题目 7 - 解法

感觉是十分 Ad-hoc 的支配对题目。

若固定了 a, b，则合法的 c 是一个后缀，可以得出的第一个结论就是，最优的 a, b 一定满足 b 是 $[a+1, b]$ 中最大的。否则可以将 b 移动到更大的，一定更优。

进一步，发现 a 一定是 $[a, b-1]$ 中最大的，理由相同。发现可能成为最优答案的 $[a, b]$ 一定满足边界严格大于中间，除掉 $(i, i+1)$ 这种二元组，记 $[a+1, b-1]$ 区间最大值为 p，一个 p 至多产生一个二元组 a, b，所以可能成为最优答案的 a, b 只有 $O(n)$ 对，且可以找出来。

题目 7 - 解法

感觉是十分 Ad-hoc 的支配对题目。

若固定了 a, b，则合法的 c 是一个后缀，可以得出的第一个结论就是，最优的 a, b 一定满足 b 是 $[a+1, b]$ 中最大的。否则可以将 b 移动到更大的，一定更优。

进一步，发现 a 一定是 $[a, b-1]$ 中最大的，理由相同。发现可能成为最优答案的 $[a, b]$ 一定满足边界严格大于中间，除掉 $(i, i+1)$ 这种二元组，记 $[a+1, b-1]$ 区间最大值为 p，一个 p 至多产生一个二元组 a, b，所以可能成为最优答案的 a, b 只有 $O(n)$ 对，且可以找出来。

把所有询问按照左端点排序，并以 a 为第一关键字加入可能成为最优答案的 a, b，更新可能的后缀，需要一个懒标记线段树即可。

最终的时间复杂度为 $O(n \log n)$。

题目 8 - 题面

给定一个长度为 n 的序列 a，求多少对 $1 \le i \le j \le n$ 满足 $a_i a_j \le \max(a_{i\sim j})$。

$1 \le n \le 10^5, 1 \le a_i \le 10^9$。

1000 ms,128 MB

题目 8 - 解法

考虑分治算法，设当前区间为 $[l, r]$，统计所有满足 $l \le i \le j \le r$ 且合法的对数。设 $[l, r]$ 内的区间最大值下标为 x。

若满足 $i \le x \le j$，则 $\max(a_{i\sim j})$ 一定是 x，反之若 $i \sim j$ 没有经过 x，则扔到 $[l, x-1], [x-1, r]$ 去分治。

题目 8 - 解法

考虑分治算法，设当前区间为 $[l,r]$，统计所有满足 $l\le i\le j\le r$ 且合法的对数。设 $[l,r]$ 内的区间最大值下标为 x。

若满足 $i\le x\le j$，则 $\max(a_{i\sim j})$ 一定是 x，反之若 $i\sim j$ 没有经过 x，则扔到 $[l,x-1],[x-1,r]$ 去分治。

对于当前区间 $[l,r]$，枚举 $i\in[l,x-1]$，查询有多少个 $j\in[x+1,r]$ 满足条件，实际上是一个二维偏序问题。主席树即可解决。

题目 8 - 解法

考虑分治算法，设当前区间为 $[l,r]$，统计所有满足 $l\le i\le j\le r$ 且合法的对数。设 $[l,r]$ 内的区间最大值下标为 x。

若满足 $i\le x\le j$，则 $\max(a_{i\sim j})$ 一定是 x，反之若 $i\sim j$ 没有经过 x，则扔到 $[l,x-1],[x-1,r]$ 去分治。

对于当前区间 $[l,r]$，枚举 $i\in[l,x-1]$，查询有多少个 $j\in[x+1,r]$ 满足条件，实际上是一个二维偏序问题。主席树即可解决。

这样的复杂度还是 $O(n^2\log n)$ 的，因为可能分治的两个区间大小规模不平衡。考虑启发式分治，每次枚举较小的一侧，计算另外一侧符合条件的答案。

分析时间复杂度，一个点如果被枚举了一次，说明他下一层的分治区间长度不到父亲的一半，所以枚举总次数不超过 $n\log n$，加上主席树，最终的时间复杂度为 $O(n\log^2 n)$。

题目 9 - 题面

给定一个长度为 n 的序列 v，q 次询问给出 l, r，求最大的 $v_a + v_b + v_c$ 且满足 $a < b < c, v_b - v_a \leq v_c - v_b$。

$1 \leq n \leq 10^6, 1 \leq q \leq 5 \times 10^4, 0 \leq v_i \leq 10^9$。

4000 ms,1 GB

题目 9 - 解法

转化条件为 $2B_y \le B_x + B_z$，枚举 y，则 x, z 分别是 $[l, pos_y), (pos_y, r]$ 内的最大值，判断是否符合条件来更新，复杂度 $O(nq)$。

题目 9 - 解法

转化条件为 $2B_y \le B_x + B_z$，枚举 y，则 x, z 分别是 $[l, pos_y), (pos_y, r]$ 内的最大值，判断是否符合条件来更新，复杂度 $O(nq)$。

不难发现，若 y 为 $[l, r]$ 最大值，找出的解一定不符合条件。其余情况可以分是 x 为 $[l, r]$ 最大值和 z 为 $[l, r]$ 的最大值。以前者来讨论。

接下来，要在 $(p_x, r]$ 里找 y, z，设 $(p_x, r]$ 内最大值为 m，分三种情况讨论：

题目 9 - 解法

转化条件为 $2B_y \le B_x + B_z$，枚举 y，则 x, z 分别是 $[l, pos_y), (pos_y, r]$ 内的最大值，判断是否符合条件来更新，复杂度 $O(nq)$。

不难发现，若 y 为 $[l, r]$ 最大值，找出的解一定不符合条件。其余情况可以分是 x 为 $[l, r]$ 最大值和 z 为 $[l, r]$ 的最大值。以前者来讨论。

接下来，要在 $(p_x, r]$ 里找 y, z，设 $(p_x, r]$ 内最大值为 m，分三种情况讨论：

1. $y \in (p_x, m)$：此时 y 选择 (p_x, m) 内最大值，z 选择 m 一定最优，且一定合法。

题目 9 - 解法

转化条件为 $2B_y \le B_x + B_z$，枚举 y，则 x, z 分别是 $[l, pos_y), (pos_y, r]$ 内的最大值，判断是否符合条件来更新，复杂度 $O(nq)$。

不难发现，若 y 为 $[l, r]$ 最大值，找出的解一定不符合条件。其余情况可以分是 x 为 $[l, r]$ 最大值和 z 为 $[l, r]$ 的最大值。以前者来讨论。

接下来，要在 $(p_x, r]$ 里找 y, z，设 $(p_x, r]$ 内最大值为 m，分三种情况讨论：

1. $y \in (p_x, m)$：此时 y 选择 (p_x, m) 内最大值，z 选择 m 一定最优，且一定合法。

2. $y = m$：此时一定选择 $(m, r]$ 内的最大值为 z 才是最优解，不一定合法。

题目 9 - 解法

3. $y \in (m, r]$：发现和现在解决的问题一样，递归解决子问题即可。注意到当情况二合法就不处理情况三。因为情况二合法的情况下选择了 $[l, r]$ 最大的两个数，$(m, r]$ 是 $[l, r]$ 的子集，选出的答案必然不优于 $[l, r]$ 最大的两个数。

题目 9 - 解法

3. $y \in (m, r]$：发现和现在解决的问题一样，递归解决子问题即可。注意到当情况二合法就不处理情况三。因为情况二合法的情况下选择了 $[l, r]$ 最大的两个数，$(m, r]$ 是 $[l, r]$ 的子集，选出的答案必然不优于 $[l, r]$ 最大的两个数。

考虑证明递归的复杂度：第一次选出的最大值为 x，$(p_x, r]$ 最大值位置为 m，$(m, r]$ 最大值为 z，递归的条件即为：

$$x + z < 2m \Rightarrow z < 2m - x$$

发现 z 就是下一轮递归的 m，带入进去得并继续递归：

$$z' < 2(2m-x)-x \Rightarrow z' < 4m-3x \Rightarrow z'' < 8m-7x \Rightarrow \cdots \Rightarrow z'' < 2^k(m-x)+x$$

不难发现对区间最大值的限制是呈现指数级减小的，所以递归此时为 $O(\log V)$。

维护区间最大值及其位置即可完成查询，区间加法用线段数维护即可，复杂度为 $O(n \log n \log V)$。

题目 10 - 题面

给一个字符串集合 S 和 m 次操作，初始 S 为空。

每次操作有三种类型，插入一个字符串 s，删除一个字符串 s，查询集合 S 中所有字符串在 t 中的出现次数，不同位置出现算多次，强制在线。

$\sum |s|, \sum |t|, m \leq 3 \times 10^5$。
3000 ms,750 MB

题目 10 - 解法

不知道是否学过 AC 自动机，但大概可以抽象为这个问题：有一个只支持静态查询的在线数据结构，预处理 $O(X)$ 的复杂度，查询需要 $O(Y)$ 的复杂度。

题目 10 - 解法

不知道是否学过 AC 自动机，但大概可以抽象为这个问题：有一个只支持静态查询的在线数据结构，预处理 $O(X)$ 的复杂度，查询需要 $O(Y)$ 的复杂度。
这类静态转动态的问题有两个非常经典的解决方法，根号重构或者二进制拆分。

题目 10 - 解法

不知道是否学过 AC 自动机，但大概可以抽象为这个问题：有一个只支持静态查询的在线数据结构，预处理 $O(X)$ 的复杂度，查询需要 $O(Y)$ 的复杂度。

这类静态转动态的问题有两个非常经典的解决方法，根号重构或者二进制拆分。

根号重构：每次要插入一个串 s，先不处理，等积攒到 $\sqrt{n}$ 个字符串再插入，预处理总复杂度为 $O(X\sqrt{n})$。对于询问，只需要额外处理 $\sqrt{n}$ 个串的对答案的贡献即可，复杂度为 $O(Y\sqrt{n})$。

题目 10 - 解法

不知道是否学过 AC 自动机，但大概可以抽象为这个问题：有一个只支持静态查询的在线数据结构，预处理 $O(X)$ 的复杂度，查询需要 $O(Y)$ 的复杂度。

这类静态转动态的问题有两个非常经典的解决方法，根号重构或者二进制拆分。

根号重构：每次要插入一个串 s，先不处理，等积攒到 $\sqrt{n}$ 个字符串再插入，预处理总复杂度为 $O(X\sqrt{n})$。对于询问，只需要额外处理 $\sqrt{n}$ 个串的对答案的贡献即可，复杂度为 $O(Y\sqrt{n})$。

二进制分组：考虑维护若干个大小为 2 的次幂的数据结构，每次插入相当于多了一个大小为 2^0 的数据结构，如果出现两个大小相同的数据结构就合并，可以证明预处理复杂度为 $O(X\log n)$，查询只需要在 $\log n$ 个数据结构上查询 $O(Y\log n)$。

题目 11 - 题面

给一个以 1 为根的有根树，对于 $i \geq 2$，节点 i 的父亲节点为 f_i，支持两种操作：

1. 给定 l, r, x，执行 $\forall i \in [l, r], f_i \leftarrow \max(1, f_i - x)$。
2. 给定 u, v，查询 u, v 的最近公共祖先。

$n, q \leq 10^5, x \geq 1$。
2000 ms,256 MB

题目 11 - 解法

首先这个题基本是分块无疑了，因为区间平移父亲的操作很难有什么 polylog 的数据结构能够做到。

首先有一个暴力求 LCA 的做法，每次让两个点深度较小的跳 LCA，最终跳到的节点就是答案，本题就是让编号大的跳。

题目 11 - 解法

首先这个题基本是分块无疑了，因为区间平移父亲的操作很难有什么 polylog 的数据结构能够做到。

首先有一个暴力求 LCA 的做法，每次让两个点深度较小的跳 LCA，最终跳到的节点就是答案，本题就是让编号大的跳。

这样做太慢了，考虑分块，设 nxt_i 表示从 i 出发第一次跳出块后会跳到那个位置，若 $nxt_a \neq nxt_b$，则直接让编号大的点跳 nxt_i 而不是 f_i，这样复杂度查询的复杂度是 $O(n\sqrt{n})$。

题目 11 - 解法

首先这个题基本是分块无疑了，因为区间平移父亲的操作很难有什么 polylog 的数据结构能够做到。

首先有一个暴力求 LCA 的做法，每次让两个点深度较小的跳 LCA，最终跳到的节点就是答案，本题就是让编号大的跳。

这样做太慢了，考虑分块，设 nxt_i 表示从 i 出发第一次跳出块后会跳到那个位置，若 $nxt_a \neq nxt_b$，则直接让编号大的点跳 nxt_i 而不是 f_i，这样复杂度查询的复杂度是 $O(n\sqrt{n})$。

对于修改操作，散块暴力更新重构 nxt_i，对于整块，如果块内存在 i, f_i 在同一块，直接暴力更新然后重构。

散块复杂度显然是 $O(n\sqrt{n})$，整块一共 $\sqrt{n}$ 块，每块至多重构 $\sqrt{n}$ 次，每次重构的代价是 $\sqrt{n}$。最终的总复杂度是 $O(n\sqrt{n})$。

题目 12 - 题面

给一个只有根节点的数，有 n 次插入一个叶子，对于叶子 $f_i = 0$，非叶节点 f_i 为所有子节点 f_j 的 mex 值。每次插入叶子后，计算根节点的 f_i 值。

$n \leq 3 \times 10^5$。
5000 ms,512 MB

题目 12 - 解法

肯定先把原树建出来，然后在这上面处理。首先，不难注意到每个点的值域为 $O(\log n)$ 的。

题目 12 - 解法

肯定先把原树建出来，然后在这上面处理。首先，不难注意到每个点的值域为 $O(\log n)$ 的。

对于这种根据儿子信息得出父亲信息层层递推的，可以考虑划分轻重儿子。不难维护出对于节点 x，只考虑他的轻儿子得到了 $\operatorname{mex} = w$，但如果重儿子权值为 w，这个节点 mex 还要改变，所以在每个点上维护仅考虑轻儿子，最小的没有出现的两个自然数。

题目 12 - 解法

肯定先把原树建出来，然后在这上面处理。首先，不难注意到每个点的值域为 $O(\log n)$ 的。

对于这种根据儿子信息得出父亲信息层层递推的，可以考虑划分轻重儿子。不难维护出对于节点 x，只考虑他的轻儿子得到了 $\operatorname{mex} = w$，但如果重儿子权值为 w，这个节点 mex 还要改变，所以在每个点上维护仅考虑轻儿子，最小的没有出现的两个自然数。

好的，对于一个节点 x，他只会对 x 到根的路径上的所有点的信息产生影响。我们只会维护每个节点轻儿子的信息，根据树链剖分理论，x 到根路径上的轻边为 $\log n$，这说明只需要修改 log 个地方就能处理好维护的信息，求出一个节点轻儿子的 mex 可以直接用桶来维护，会因为值域最多是 $\log n$，所以这部分的复杂度为 $O(n \log^2 n)$。

题目 12 - 解法

如何维护答案？对于每一条重链，链底权值是 0，他的父亲和祖先的权值是形如“若重儿子权值为 v，则自身权值为 a，否则为 b”的函数形式，这个东西实际上可以放在线段树上维护，我们只关心链头，链尾和中间的和，写个结构体封装一下即可。复杂度为树剖加线段树的 $O(n\log^2 n)$

对于这种当前节点的值由所有儿子推出来，都可以用这种算法来解决，好像比较经典。ddp 也是运用了类似的思想。

结语

感谢聆听。