1.【互测】Cyberangel

给定一个长度为 n 的数组 A，其中 $a_i \in [1, m]$，令 $f(l, r, i)$ 表示 $j \in [l, r]$ 中所有满足 $a_j \le i$ 的 a_j 的最大值，如果不存在这样的 j，则 $f(l, r, i)$ 为 0。

求 $\sum_{l=1}^{n} \sum_{r=l}^{n} \sum_{i=1}^{m} f(l, r, i)$。

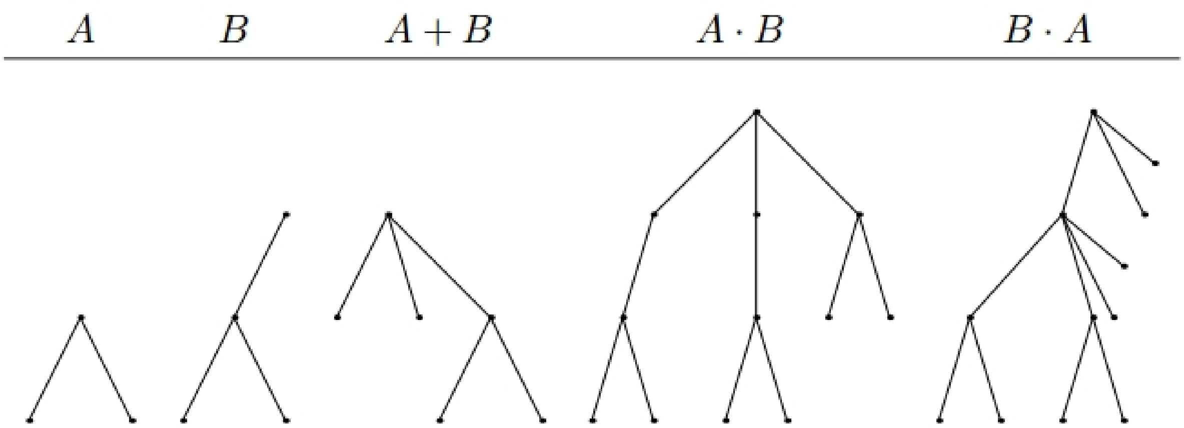
对于所有测试点，$1 \le n \le 1 \times 10^6, 1 \le m \le 1 \times 10^9$。

题解看我互测题解。

2.[Tree Equation - Problem - Universal Cup Judging System (ucup.ac)]

题意：

定义树的加法和树的乘法为以下：



请注意，乘法没有交换律。

给定树 A, B, C，要你解方程 $AX + BY = C$ 中的树 X 和树 Y。

$|A|, |B|, |C| \le 2 \times 10^6$

题解：

先考虑对于单个方程 $AX = C$ 怎么解。

首先我们可以通过 $|AX| = |A||X| = |C|$ 解出 $|X|$。

由于 $\forall A$，$|AX| \ge |X|$，而对于所有 X 的儿子的子树 $|X'| < |X|$。

所以我们可以知道 X 由 C 的所有大小小于 $|X|$ 的子树加上根组成。

再考虑两个相加的情况。

由于做了一次树的加法，我们不好判断 C 的每个子树是怎么来的。

但是特判 A, B 都是单点的情况之后，C 的最大的子树一定是由 A 的最大子树乘上 X 或者 B 的最大子树乘上 Y 得到。

枚举这两种情况，不妨假设是由 A 的最大的子树得来的。

虽然可能有很多最大子树，但我们仍然能解出 $|X|$，从而在 C 的最大子树 C' 中找到 X。

找到 X 之后，做树哈希，我们可以去除 AX 的部分，从而解剩下的一元方程 $BY = C$。

复杂度可以做到 $O(n\log n)$ 或 $O(n)$。

3.[Triangle - Problem - Universal Cup Judging System (ucup.ac)]

题意：

定义字符串的加法为拼接，比较为字典序比较。

定义字符串 A,B,C 满足三角关系，当且仅当三个条件都满足：

1.$A+B>C$ 或 $B+A>C$。

2.$B+C>A$ 或 $C+B>A$。

3.$A+C>B$ 或 $C+A>B$。

$n\leq 3\times 10^5,\sum|S|\leq 10^6$

题解：

显然只用考虑字典序最大的字符串作为斜边。

考虑如果满足 $A+B>C$，且 $A\leq C$，则 A 一定是 B 的前缀。

对于每个 C 枚举 A 的复杂度是 $O(\sum|S|)$ 的。

把所有字符串从小到大排序后，满足条件的 B 肯定构成了一段区间，可以后缀排序预处理，加上二分可以做到 $O(\sum|S|\log n)$。

但是这样会算重，因为可能 $A+B>C$ 和 $B+A>C$ 同时满足。

发现这种情况是个二维偏序，用树状数组处理即可。

总复杂度 $O(\sum|S|\log n)$。

4.[Add One 2 - Problem - QOJ.ac]

给定一个序列 a，初始有一个全 0 序列 b。

每次可以选择一个长度为 k 的前缀或者一个长度为 k 的后缀加一，代价为 k。

问最少需要多少代价才能使所有 i 都满足 $b_i\geq a_i$。

$n\leq 10^6,a_i\leq 10^9$

做法：

考虑可能的序列 b 满足什么条件，可以考虑反向操作，将这个序列减为 0。

考虑差分，如果 $b_i>b_{i+1}$，那么至少要进行 b_i-b_{i+1} 次前缀 i 减一的操作。反之亦然。

上述操作完之后，所有的元素都变成相同，如果这个时候 b 非负，那么满足条件。

为了方便，我们在序列开头和结尾各加一个足够大的元素 M，则上述条件可以变成 $b_0\geq\sum_{i=0}^{n}max(0,b_i-b_{i+1})$。

由于 $b_0=b_{n+1}$ 且足够大，则 $\sum_{i=0}^{n}max(0,b_i-b_{i+1})=\sum_{i=0}^{n}max(0,b_{i+1}-b_i)$，把左右两边对应的限制相加，那么上述条件可以变成 $\sum_{i=0}^{n}|b_{i+1}-b_i|\leq 2M$。

在原问题中，答案等于 b_i 之和，所以问题变为，可以花 1 的代价让 a_i 加 1，问最少的代价满足上述条件。

令 $F=\sum_{i=0}^n |a_i-a_{i+1}|$，每次我们可以选择一段极长的值相同的区间，并且 $a_{l-1}\ge a_l, a_{r+1}>a_r$，将这段整体加一就能使 F 减 2，代价为这段长度。

我们贪心的找这样最短的连续段即可，可以对这一段加，然后加到这一段与两头的某个数字相同，一直加到 F 不超过 $2M$。

考虑段合并的过程，这很类似于笛卡尔树，我们从短到长贪心，时间复杂度 $O(n)$。

5. Periodic Sequence - Problem - QOJ.ac

题意：

给定 n，对于 $i=1,2,\cdots,n$ 求出最长可能的周期字符串序列长度，满足序列中字符串的长度 $\le i$。

一个字符串序列是周期字符串序列，当且仅当每个 S_i 都是 S_{i+1} 的周期，并且它们两两不同。

假设字符集无穷大。

$n\le 2\times 10^5$

首先考虑对于一个 n 如何求 $F(n)$。我们可以贪心地增量构造字符串序列。

初始序列为单个字符 $\{a\}$，对应了 $F(1)=1$。

对于长度 l，假设我们已经有了一个所有串长度 $\le l$ 的最优序列，那么我们将这个序列中的每个字符串后插入一个长度为 $l+1$ 的字符串，且在序列的开头插入由一个 a 和 l 个 b 构成的字符串 $ab\dots b$ 。

显然，根据题目的性质，每个位置能添加的字符串是唯一的，如果出现相同的情况，优先保留靠前的位置。

下面说明这个构造是最优的：

首先我们为 $F(n)$ 分析一个上界，设最优的字符串序列为 $S_1,\dots,S_{F(n)}$。若 $|S_1|\ne n$，我们可以在开头插入一个长度为 n 的，满足 S_1 是其前缀的字符串，一定不会使答案变劣，我们允许开头元素存在和后面的重复，不影响 $F(n)$ 的上界，设 $|S_1|=n$。

容易利用题目条件归纳得出，对于任意一个序列中的字符串 S_i，其可以表示成若干个字符串 T_j 的拼接，且每个 T_j 都是 S_1 的前缀，更进一步的，我们要求这些字符串满足 $|T_1|=max_{j=1}^k|T_j|$。同样归纳得出，对于任意一个字符串，均有满足上述两个条件的表示方法。

由序列中任意两个串不同可以得出，当且仅当上面的表示方法不同序列合法，则 $F(n)$ 的一个上界为满足 $\sum_{i=1}^k x_i\le n, x_1=max_{i=1}^k x_i$ 的序列 $\{x_i\}$ 的个数。

取 $S_1=ab\dots b$ 可以满足上述条件。且可以归纳证明，上述增量构造可以满足所有上述可能的字符串均存在。

于是答案变成了这个上界的大小。

可以写出 $F(n)$ 的生成函数为：

$$\frac{1}{1-x}\sum_{k=1}^{\infty}\frac{x^k}{1-\frac{x-x^{k+1}}{1-x}}$$

化简可得：

$$\sum_{k=1}^{\infty}\frac{x^k}{1-2x+x^{k+1}}$$

撒一点扑朔迷离的小把戏：

$\sum_{k=1}^{\infty}\dfrac{x^k}{(1-2x)(1+\dfrac{x^{k+1}}{1-2x})}$

对于 $k \le \sqrt{n}$，则可以暴力计算，复杂度 $O(n\sqrt{n})$。

对于 $k > \sqrt{n}$，我们考虑将展开式展开，则最多有 $\frac{n}{k}$ 个项有用，复杂度 $O(n\sqrt{n})$。

总复杂度 $O(n\sqrt{n})$。

6.Min or Max 2 - Problem - QOJ.ac

题意：

给两个 1 到 n 的排列 a, b。

初始有个二元组 $(x, y) = (a_1, b_1)$，对于每个 $2 \le i \le n$，每次可以选择

把 (x, y) 变成 $(\max(a_i, x), \max(b_i, y))$

把 (x, y) 变成 $(\min(a_i, x), \min(b_i, y))$

对于 $k = 1 \dots n$，你要回答有多少最终的 (x, y) 满足 $|x - y| = k$。

$n \le 5 \times 10^5$

题解：
令最终的二元组为 (a_i, b_j)，考虑有哪些 (i, j) 是可能取到的。

不妨令 $i \le j$。先枚举 i 处的操作类型，则 i 到 n 的所有操作类型已经确定。

若 2 到 $i-1$ 的操作结束后的二元组为 (x, y)。则 i 到 n 中操作对 y 的影响为 $\min(\max(y, l), r)$。

因此 j 只有两种可能的取值 $b_j = l$ 或 $b_j = r$。

l, r 可以在扫描 a_i 的过程中用线段树维护相应信息，复杂度 $O(n \log n)$。

此时我们只需再用用算出进行完 2 到 $i-1$ 的操作后 $x < a_i$ 和 $x > a_i$ 时，y 的最大和最小值，即可判断 l 和 r 是否可以取到。

总复杂度 $O(n \log n)$。

7.Sticks - Problem - QOJ.ac

给你一个 $n \times n$ 的矩阵，每行的最左边和每列的最上面有一根小木棍，令它们长度分别为 $x_1, \dots, x_n, y_1, \dots, y_n$。

我们要求这些木棍长度为 0 到 n 的整数，并且这些木棍不交。

定义一个 01 矩阵 A，其中 01 表示每格是否被木棍占据。

现在给你一个带 01? 的矩阵，问有多少种方法把 ? 换成 01 使其合法。

$n \le 3000$

题解：

考虑如何判定 A 是否合法，假设存在一组 $x_1, \dots, x_n, y_1, \dots, y_n$ 满足条件。

显然我们一定可以找一条从 $(0,0)$ 到 (n,n)，每次往右或往下走一格的路径 P，使得所有行木棍都在 P 左下方，列木棍都在右上方。

且存在这样的 P 一定合法。

我们求出 a_i 表示第 i 行表示第 i 行最多有多少个在 P 左下，b_i 表示第 i 列最多有多少个在 P 右上。

具体地，对于第 i 行，我们只需要找到最小的 j 是第一个断开的地方即可。

则我们可以根据 a 和 b 求出两条路径 P_1 和 P_2，则一条 P 合法，当且仅当 P 在 P_1 左下，且在 P_2 右上。

考虑如何计数。

可以发现，如果至少存在一个 P 合法，则 P_1 一定合法。

那么我们把所有的方案在对应 P_1 进行计数，我们只需要对 P_1 进行 dp，复杂度 $O(n^2)$。

8.Bot Friends - Problem - QOJ.ac

题意：

一条数轴上有 $n+1$ 个洞，每两个洞之间有一个球。

你可以按任意顺序推球，每个球会有一个限制：只能往左， 只能往右，或者可以朝两个方向。

球会一直移动，直到碰到一个空的洞。

问最多会有多少个球，不会掉到相邻的洞里。

$1 \le n \le 5000$

题解：

先考虑那个没有球的洞在哪，然后分别考虑左右两边，相当于每次可以删除一个球，如果这个球指向的球和它方向相反，就把它指向的球涂黑，最终就是要最大化涂黑的球的数量。

假设考虑的是右边，那么删除的过程中第一个球必须要朝右。

对于右边的情况，这个题就变成了从右往左依次扫，因为一定恰好有一个球会贡献，所以每次遇到 >< 而且两个都没有涂黑就要决策一下涂黑哪个。

如果剩下来的是 > 就不会对后续做出贡献直接删掉，否则如果是 < 就可以观望。

dp 过程中记录一下剩多少个 < 和第一个 < 是否被涂黑，就可以做到 $O(n^2)$。

9.[P10197 USACO24FEB] Minimum Sum of Maximums P - 洛谷 | 计算机科学教育新生态 (luogu.com.cn)

虽然这题被讲过很多遍了，但是选自 USACO 24 Feb Pt T2 题解 - 洛谷专栏 (luogu.com.cn) 。

题意：

给定一个长度为 n 的序列，有 K 个位置被固定住了，剩下的数之间可以随意交换，问最小的 $\sum_{i=1}^{n-1} \max(a_i, a_{i+1})$ 的权值是多少。

$n \le 300, K \le 6$。

题解：

考虑将 $\max(a_i, a_{i+1})$ 拆成 $\dfrac{a_i + a_{i+1} + |a_i - a_{i+1}|}{2}$。

其中 $a_i + a_{i+1}$ 是固定的，意思就是我们要最小化 $|a_i - a_{i+1}|$ 的总和。

对于边界的情况，我们可以往两端塞入一个被固定的充分大的数 C，最后再减去这部分的答案即可。

由于原来固定的 K 个位置将原序列分为至多 $K+1$ 段，考虑假如我们已经将数分配进了这 K 段里面，段内如何分配才是最优。

由于段与段之间互不影响，我们考虑对于一段计算贡献。

考虑这一个段左边的数为 L，右边的数为 R，不失一般性地假设 $L \leq R$。

那么我们肯定是将分配进来的数从左到右，从小到大排最优，因为可以通过如下的调整法证明：

1. 假如相邻的三个数 x, y, z 满足 $x \geq y \leq z$，那么将其变为 y, x, z 不会更劣。
2. 同理，对于 $x \leq y \geq z$ 将其变为 x, z, y 不会更劣。

然后一直调整可以使分配的数有序，然后对于边界，我们已经假设 $L \leq R$，那么把小的放左边肯定更优。

设这段数的最小值为 mi，最大值为 ma，那么这一段的贡献是
$|L - mi| + ma - mi + |R - ma|$。

则最后答案为

$\sum_{i=1}^{K+1} |L_i - mi_i| + ma_i - mi_i + |R_i - ma_i|$

答案只和每一段的最小值 mi_i 和 ma_i 有关。

考虑如何分配 mi_i 和 ma_i。

性质：存在一种最优方案，使得对于任意的数对 (i, j)，都满足 (mi_i, ma_i) 和 (mi_j, ma_i) 的关系为相离或者包含。

证明：

还是考虑调整法，不妨设存在 (i, j) 满足 $mi_i < mi_j < ma_i < ma_j$，则我们可以通过交换第 i 段里最大的数和第 j 段里最小的数，直到 $ma'_i \leq mi'_j$，此时 $ma'_i \leq ma_i$ 且 $mi'_j \geq mi_i$。

考虑 ma_i 变小对第 i 段贡献的影响，由于第 i 段贡献与 ma_i 有关的只有 $ma_i + |ma_i - R|$，分类讨论可得 ma_i 变小后，这个式子要么不变要么变小。

同理 mi_i 变大，对答案的贡献也要么不变要么变小。

所以这样调整会使得答案不劣。

证毕。

从上面的式子我们可以得到一个大概的思路，通过确定每段的边界，然后将剩下的数填进去。

容易发现我们不关系中间的数怎么填的，只需要有一个合法的填数方案即可。

考虑我们从小到大考虑每个数，要么它作为某一段的左边界，要么它填入目前左边界最大的没填满的段（因为根据上面性质可知，左边界最大肯定有边界最小，我们先贪心地满足更紧的限制），此时若将这个段填满了就直接作为这个段右边界。

考虑我们如何实现这个填数过程，考虑区间 dp，设 $f_{l,r,S}$ 满足下标为 $[l, r]$ 的数已经填满集合为 S 的段的最小贡献。

由上面的填数过程可知，我们不会在内部的小段留一些数给外面更大的段。（调整法也可以证明）

因此，设 len_S 表示集合为 S 的段的长度之和，满足 $f_{l,r,S}$ 最小且 $r-l+1$ 最小的 l,r 一定满足 $r-l+1=len_S$。

于是我们只用考虑三种转移，

1. 目前左边/右边的数留给下一个包住它的段，$f_{l,r,S}=\min(f_{l+1,r,S},f_{l,r-1,S})$，这部分转移是 $O(1)$ 的，复杂度等于状态数 $O(2^K n^2)$。
2. 把两端拼起来，考虑枚举子集，$f_{l,r,S}$ 可以从 $f_{l,l+len_{S'}-1,S'}+f_{r-len_{S-S'}+1,r,S-S'}$ 转移过来，复杂度为 $O(3^K n^2)$。
3. 用一个大段包住中间的小段，$f_{l,r,S}$ 可以从 $f_{l+1,r-1,S-\{x\}}+F_x(a_l,a_r)$，其中 $F_x(a_l,a_r)$，表示第 x 段最小值为 a_l 最大值为 a_r 的贡献，这部分由于用大段包住小段的时候一定满足 $r-l+1=len_S$，所以复杂度为 $O(n2^K K)$ 的。

最后复杂度为 $O(3^K n^2)$，瓶颈为第二种转移。

10.Two Transportations - Problem - QOJ.ac

题意：

这是一道通信题。

有一个点数为 n 边数为 m 的带正权的无向图。其中边集被划分为两个不交的集合 A，B。

有两台机器 MA,MB，初始时 MA 只知道 A，MB 只知道 B。你需要让机器 A 回答 0 到所有点的最短路。

两台机器可以互相发 bit。具体地，存在两个队列 a,b，MB 可以调用函数 $SendA$ 去将一个 bit 加入到队列 a 的末尾，$SendB$ 同理。grader 会不断调用 $ReveiveA$ 将 a 队首的 bit 发给 MA，$ReveiveB$ 同理。

$n\le 2\times 10^3, 0\le |A|,|B|\le 5\times 10^5$，边权 $\in[1,500]$。

两台机器加起来只能发 58000 个 bit。

题解：

考虑我们可以在两张图上一起跑最短路，每次拿出两个堆里堆顶较小的那个，然后用这个点在两张图上 分别更新。

那么就是这样的过程：

1. 两个人都往对面传堆顶的值。
2. 发现自己比较小的那一方向对面传堆顶那个点的编号。
3. 分别更新，再回到第一步。

现在每取出一个点需要 $\log n+2\log(wn)$，但是注意到我们并不需要真正地传距离的实值，只需要传这 个值和已经取出的点中最大距离的差值即可，这个差会在 $[0,w]$ 中，所以花费变成 $\log n+2\log w$，恰好能过。

11.【ULR #2】Picks loves segment tree IX - 题目 - Universal Online Judge (uoj.ac)

题意：

给定一个操作序列，有四种类型：

1. or x
2. and x

3. xor x
4. $+1$

多组询问，问 v 在经过 l 和 r 的操作后第 t 位是 0 还是 1。

强制在线。

$n \le 10^5$，$q \le 6 \times 10^5$

做法：

考虑 +1 操作，等价于把后面若干位连续的 1 变成 0，再往高位进 1，因此，第 i 位被加 1 影响了，那么 0 到 $i-1$ 位都得是 0。

记 $f_{i,j}$ 表示从第 i 个操作开始，假如至少前 j 位都是 0，则满足操作完 i 到 $f_{i,j}-1$ 的操作后，至少前 j 位都是 0 的最小位置。

同理定义 $g_{i,j}$ 表示前 j 位都是 0，下一位是 1，则满足操作完 i 到 $g_{i,j}-1$ 的操作后，至少 $j+1$ 位都是 0 的最小位置。

则第 j 位在操作 i 到 $f_{i,j}-2$ 的过程不会被操作 4 影响，如果 $f_{i,j}-1$ 是操作 4，则等价于异或上 $2^{i+1}-1$，然后其他三种操作都可以快速计算某一位的答案。

考虑如何计算 $f_{i,j}$ 和 $g_{i,j}$。

假设第 $j-1$ 位为 0，考虑它在操作完 i 到 $f_{i,j-1}-1$ 会变成什么，如果还是 0，则 $f_{i,j}=f_{i,j-1}$，否则 $f_{i,j}=g_{f_{i,j-1},j-1}$。

同理可以计算 $g_{i,j}$。

然后询问我们可以先跳到至少前 t 位都是零，然后不停跳 $f_{i,t}$，维护每个 $f_{i,t}$ 跳完第 t 位会变成什么，转移可以写成矩阵或向量的形式，然后用倍增/树剖维护。

复杂度 $O(n\log V+q\log V)$。

12.[Mini Metro - 洛谷 | 计算机科学教育新生态 (luogu.com.cn)]

同 [铁路 - 题目详情 - NFLSOJ]。

有 n 个站台，从左到右编号为 $1\ldots n$，第 i 个站台初始有 a_i 个人。

每个时刻你都能进行任意次如下操作：喊一辆货车，把所有人里从左到右前 K 个接走（不够 K 个全都接走）。

在每个时刻末，第 i 个站台会来 b_i 个人，如果第 i 个站台人数超过 c_i 你就输了，你需要 t 个时刻都不输，求至少需要喊来几辆火车。

$n,t \le 200, a_i,b_i,c_i,k \le 10^9$

容易发现，当你开始拉第 i 个站台的时候，前 $i-1$ 个站台肯定是空的。

为了方便处理边界情况和思考，我们可以在末尾加个无穷多人的站台。

我们可以定义动态规划状态 $f_{i,j,k}$ 为只考虑前 i 个站，存活到 j 时刻，初始状态为 k 最少需要的车数，其中 k 为 1 表示这个车站还未被清空过，为 0 表示这个车站因为要拉到后面的车站而被强制清空了。要求所有的车必须满载（不准拉后面站台的乘客），如果不行则为 $+\infty$，答案就是 $f_{n+1,t,1}$。

为了方便转移，类似地，我们定义 $g_{i,j,k}$ 表示在 j 时刻 k 状态下把 $1,2,\ldots,i-1$ 全清空的最少车数。

考虑转移，如果 $a_i + j \times b_i \le c_i$ 就可以直接从 $f_{i-1,j,k}$ 转移到 $f_{i,j,k}$，且只要 $f_{i-1,j,k} \ne \infty$ 就有 $g_{i,j,k} \leftarrow \lfloor \frac{k\sum_{a_i} + j\sum b_i}{K} \rfloor$。

对于 $a_i + j \times b_i > c_i$ 或者 $f_{i-1,j,k} = \infty$ 的情况，我们可以枚举上一次拉到第 i 站人的时刻 t，然后计算在这个时刻需要拉几辆火车，以确保第 i 个站能存活到第 j 时刻，然后剩下的时间只需要考虑 $i-1$ 站台，g 的计算同理。

复杂度 $O(nt^2)$。

13.[AGC061E] Increment or XOR - 洛谷 | 计算机科学教育新生态 (luogu.com.cn)

题意：

你有一个非负整数 X 初始为 S。给定 N 和数列 $Y_1, Y_2, \ldots, Y_N$ 以及 $C_1, C_2, \ldots, C_N$。

你可以进行下列操作任意多次：

· 令 $X \leftarrow X + 1$，花费 A 的代价。

· 令 $X \leftarrow X \operatorname{xor} Y_i$，花费 C_i 的代价。

你想把 S 变成 T，求最小代价。

$1 \le N \le 8, 0 \le S, T, Y_i \le 2^40, 0 \le A \le 10^5$，$0 \le C_i \le 10^{16}$。

题解：

分析一下 $+1$ 的操作，等价于把末尾一段 1 变成 0 。

然后接下来从 0 开始转移，于是我们可以记录末尾 0 的个数。

某次 $+1$ 操作后会把 $[0, i]$ 清空，会形成一个子问题（从 0 到某个状态），则初始状态只有 S 和 0 两种。

并且，在 $+1$ 操作冲破第 i 位之前，在第 i 位之前的更高位不会受 $+1$ 操作的影响，此时只用知道每个数被异或了多少次就可以知道这些位的值。

可以从低到高考虑每一位，假设现在只考虑最低的 i 位。

初始状态有两种， 0 到 $i-1$ 位从 S 开始和从 0 开始。

结束状态有两种，0 到 $i-1$ 和 T 相同且不向更高一位进位；0 到 $i-1$ 位清零并向上进一位。

设 $f_{i,0/1,0/1,S}$ 表示只考虑最低的 i 位（即 $+1$ 操作不影响后面的位置），初始和结束状态为 $0/1$ ，且此时每个 Y_i 被异或的状态。

考虑转移：

1. 如果第 i 位是可以直接匹配的，那么 f_i 可以转移到 f_{i+1}。
2. 否则，考虑进行了哪些操作。

   首先，会先调用一次 $f_{i,j_0,1,msk}$，从开始状态 j_0 开始把第 i 位从 0 进位到 1，且不影响更高的位。

   然后，会调用若干次 $f_{i,1,1,msk}$ 进位操作，且每次都是只影响第 i 位，此时第 i 位一定是 1，不影响更高位。

   最后，调用一次 $f_{i,1,j_1,msk}$ 操作，此时第 i 位符合 j_1 的要求。

相当于每一步都是进行了一次 $+1$ 把前缀清零去影响第 i 位，容易发现这是最短路的形式，转移的复杂度为 $O(2^{2n})$。

总复杂度 $O(2^{2n}\log V)$。

后言：

最后三道题有类似之处，都是分阶段进行的操作，也就是低位想要影响高位不能跨过中间的位置去影响，则 dp 的转移是一种分阶段的转移。