A - 选举

考虑容斥，我们会钦定一些人投给自家人。

先想一下钦定一部分人后的方案数怎么算，令 x_i 表示一个人接收到的钦定票数，发现方案即为 $\frac{(n-\sum x_i)!}{\prod (c_i-x_i)!}$ 。

那来思考，如果确定了序列 x ，怎么算有多少种钦定方式会得到 x 呢？发现我们只需要对每个家庭分别考虑，答案即 $\prod\limits_{i=1}^{n} \frac{a_i!}{(a_i-\sum\limits_{t_j=i} x_j)! \prod\limits_{t_j=i} x_j!}$ ，其中令 a_i 表示家庭 i 的人数。

那么一个序列 x 给答案带来的贡献即为：

$(-1)^{\sum x_i} \frac{(n-\sum x_i)!}{\prod (c_i-x_i)!} \prod\limits_{i=1}^{n} \frac{a_i!}{(a_i-\sum\limits_{t_j=i} x_j)! \prod\limits_{t_j=i} x_j!}$ ，其中 $(-1)^{x_i}$ 是容斥系数。

整理一下式子，把只和一个人相关的部分整理到一起，即 $\prod a_i! * \prod \frac{(-1)^{x_i}}{(c_i-x_i)!x_i!} * (n - \sum x)! * \prod\limits_{i=1}^{n} \frac{1}{(a_i-\sum\limits_{t_j=i} x_j)!}$ 。

现在对每个家庭分别求出 f_i 表示满足 $\sum x = i$ 的方案的 $\prod \frac{(-1)^{x_i}}{(c_i-x_i)!x_i!}$ 之和，这是容易的，可以直接 dp 出来。算完了之后把 f_i 乘上 $\frac{1}{(a-i)!}$ 。

然后开始第二层 dp ，我们只关心 $\sum x$ ，就设 $dp_{i,j}$ 是考虑了前 i 个家庭，当前 $\sum x = j$ 的所有方案的权值和，转移也是容易的。 答案即 $\prod a_i! \sum dp_{n,i}(n-i)!$ 。

复杂度 $O(n^2)$ 。

bonus: $O(n \log^2 n)$?

B - 替换

考虑二分答案 mid 。

先想一想暴力怎么做。注意到每个位置的操作其实是独立的：我们可以看成先对 1 进行不超过 mid 次操作，再对 2 进行不超过 mid 次操作，这样做下去。

从左往右依次确定 a_i 最终等于多少。肯定希望它尽量小，但也需要 $\geq a_{i-1}$ 的最终值。

接下来是这个题最灵魂的一步：令 X 是 a_{i-1} 的最终值，我们只需要依次判断：从 a_i 开始能否通过至多 mid 次操作成 $X, X+1, X+2, \ldots$ 这样尝试下去，直到尝试成功就停止，得到 a_i 的最终值。

可以发现一次尝试要么成功，要么 X 会 $+1$ ，所以我们只会做至多 $n+m$ 次尝试！

问题转化成：若干次询问，每次给出 s, t ，想知道从 s 开始做多少次操作才能变成 t 。

怎么做呢，考虑把这些数看成图上的点，我们建立一个有向图，连边 (i, b_i) 。可以发现每个点出度为 1 ，即内向基环森林。现在问题变成：从 s 出发，走多少步才能走到 t 。

先考虑每个环都是自环的情况：可以忽略自环变成森林，那 s 能走到 t 等价于 t 是 s 的祖先，步数即 $d_s - d_t$ (d 是深度)。

现在不是自环，也可以先删掉环上任何一条边 (u, v) ，此时得到一颗以 u 为根的内向树。先判掉 t 是 s 的祖先的情况，和 s, t 不在一棵树的情况；接下来想走到 t 必须得先往上走到 u ，u 再跳到 v ，再尝试从 v 往上走到 t 。也就是说如果 t 是 v 祖先，答案即 $d_s + d_v - d_t + 1$ 。

可以发现预处理后单次判断是 $O(1)$ 的，本题复杂度即 $O((n+m)\log m)$ 。

C - 小半

对于查询二分答案，考虑二分答案 m，问题转化成在 $b[l:r]$ 中选尽量多的数，使相邻二者的和都 $\leq m$ 。

把 $\leq \frac{m}{2}$ 的看成 1 ，$> \frac{m}{2}$ 的看成 0 。可以发现两个 1 是一定能相邻的，两个 0 一定不能相邻。由于相邻两个 1 之间只能取至多一个 0 ，所以 1 一定是会全部取的，否则可以调整。

问题变成计算有多少个相邻的 1 之间能再选一个 0 。我们一定会选最小的那个 0 。考虑特别地判断第一个 1 与最后一个 1 中间能否选 0；再来看中间的部分。

考虑 m 从小到大增长的过程中这个 01 序列的变化，01 序列只会变化 n 次，且每次都是某个 0 变成了 1 。那不同的“相邻的 1” 只有 $O(n)$ 对。考察每一对可能出现的相邻的 1 ，我们设这一对数位置在 x, y ，$A = \max(b_x, b_y), B = \min\limits_{x<i<y} b_i$ 。可以发现当 $m \in [A+B, 2B)$ 的时候这一对数会产生贡献。

对询问整体二分，这样我们就可以以 $O(n \log^2 n \log V)$ 的复杂度算出答案，因为我们要做动态的二维数点。这是不够优的。

考虑优化。我们二分时需要计算的是会产生贡献的相邻的 1 的个数，限制这些相邻的 1 满足 $l \leq x < y \leq r$ 。考虑直接计算满足 $l \leq x \leq r$ 的相邻 1 个数。接下来只需要判断 $\leq r$ 的最后一个 1 ，和 $> r$ 的第一个 1 之间是否产生了贡献，如果产生了就减掉即可。转化了计算方式后我们甚至不需要整体二分，直接用主席树处理即可。复杂度 $O(n \log n \log V)$ 。