rap

先把每个拼音处理只剩韵母比较部分：处理包括：删去声母，删去后鼻音的 `g` ，删去 `ia,ie,ue...` 前面的 `i,u` 。然后给每个韵母给一个数字代表他，把一整句话放在类似字典树的东西里面。会计数重复，最后做一个后缀差。也可以暴力二分判断，但复杂度 $O(n^2 \log n)$ 写得优秀的话可以过。

draw

构造题。

自行手搓我们不难发现，在大部分情况下，我们只要定好了 1 的位置，其他的位置只需要在斜线上依次放上 $2 \sim n$ 即可

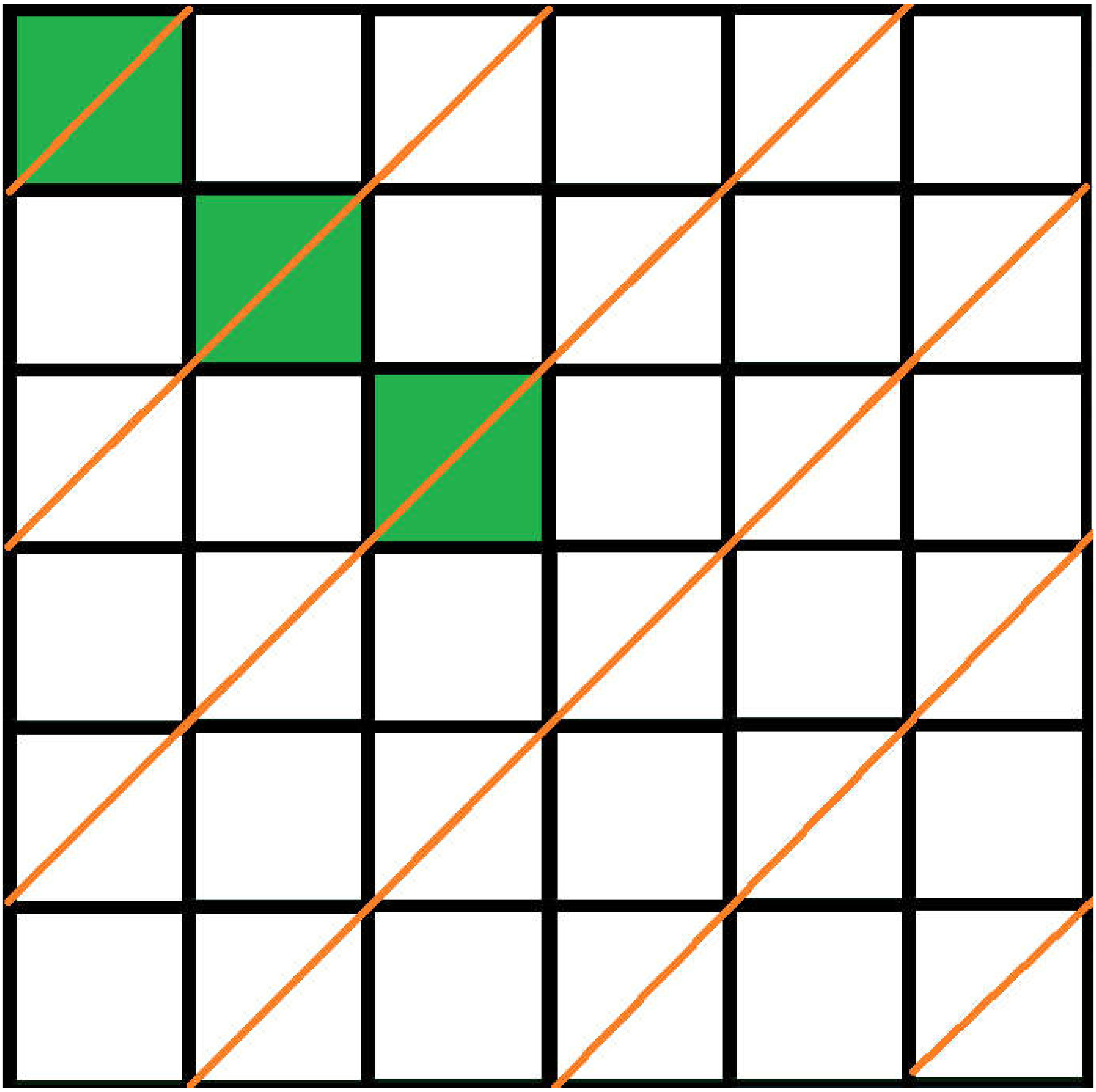
我们先构造行列为奇数 $(20pts)$ 的方法：

从**样例#2**，我们可以较为轻松地得到构造方案：

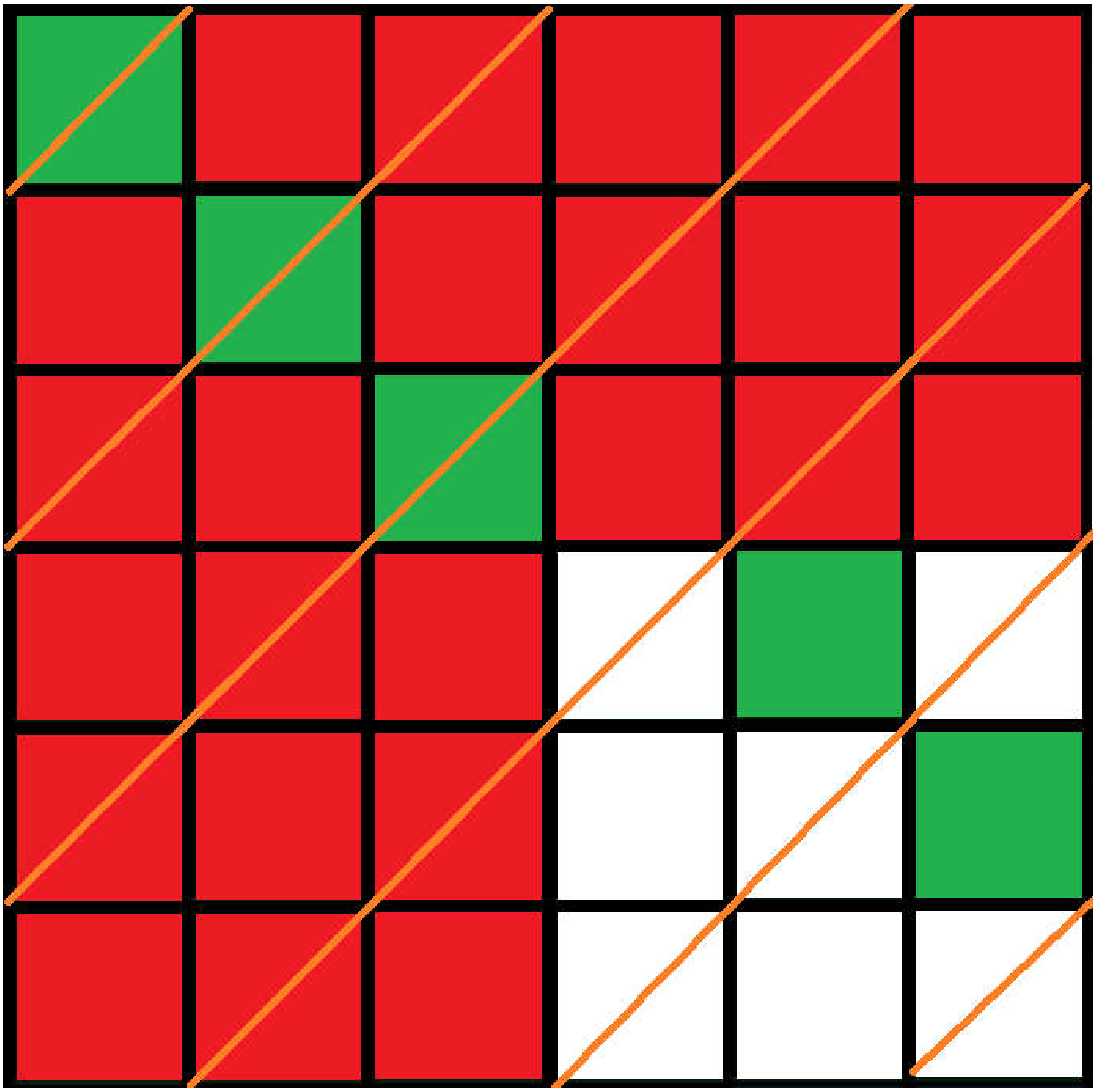
- 从**左上到右下**的**对角线**全部放 1
- 对于所有**斜线**，依次放上 $2 \sim n$ 即可

困难的部分是后面的 $(80pts)$ 的方法，以 6×6 的方格为例：

- 出题人给 n 为偶数的部分分肯定不止是为了防爆零对吧……
- 我们还是先按 $20pts$ 的方法放 1，如下图（绿色为放 1 的格子，橙色线为放了这些 1 之后通过 斜线上为排列 的性质后对其他格子不能放 1 的限制）：



- 我们可以发现，我们只能放 $20pts$ 方案中左上角的格子，然后我们就会发现行列编号之和为偶数的格子都不能放了，只剩下行列编号之和为奇数的格子。
- 然后我们就在行列编号之和为奇数的格子里面继续找地方放 1，如下图（红色为通过 行/列为排列 的性质不能放 1 的格子）：



我们就是将原先右下角放 1 的格子向右移了 1 格。

然后还是在**斜线**上放剩下的 $2 \sim n$

至于对角线上的就直接通过该 行/列 直接推即可。

$Then\ that's\ all, thanks.$

love

整理式子:

$$\sum_{i=1}^{k} |a_i x + b_i|$$

Sol 1

首先考虑绝对值是个凸函数的性质：`凸函数 + 凸函数 = 凸函数`。可以发现，所求函数满足两段单调。

所以我们可以三分 f 然后 $O(n)$ 算出最终答案。

复杂度 $O(n^2 \log v)$ 期望得分 20。

然而三分瓶颈在于求出最终答案，考虑如何优化。

我们把绝对值拆开分段考虑。

当 $x < -b_i/a_i$ 时，与 $x \ge -b_i/a_i$ 时贡献不一样，考虑将函数按照 $-b_i/a_i$ 排序，将函数相加（对 a_i 和 b_i 做前后缀和，利用数据结构维护）

所以可以三分断点，以此求出答案。

注意 $a_i = 0$ 的情况，以及需要使得 a_i 为正（a_i, b_i 都取相反数即可）。

复杂度 $O(n \log n \log v)$，期望得分 75。

---

sol 2

先考虑特殊性质。

不难发现，可以转化为最小化数轴上一个点到其他点的最小距离。

这应该给了很多提示。所以这一部分可以通过 `heap` 水过。

实测，其实利用堆可以过大部分数据，只是这复杂度……$O(玄学)$

期望得分：玄学

类似的考虑正解：

利用绝对值，**强制**使得 $a_i \ge 0$，同时也改变 b_i 的符号，可以得到类下的式子：

$$\sum_{i=1}^{k} a_i |x + \frac{b_i}{a_i}|$$

于是相当于在数轴上 $-\frac{b_i}{a_i}$ 的位置放 a_i 个点。求 x 到这些点的最小距离。

而如果要最小化距离，那么显然是在中间的位置。

所以考虑使用平衡树维护中位数，以及 b_i 前缀和，求解距离即可。

> 同时，也可以考虑离线后离散化，在线段树或者树状数组上二分可以做到 $O(n\log n)$。

期望得分 100。

---

sol 3

SMB 给出了另一种想法，每加入/删除一个一次函数相当于区间加/减一个斜率。

离散化出加减的断点，维护线段树支持区间加，树上二分找到最低点即可。

复杂度 $O(n\log n)$，期望得分 100。

plant

首先考虑 $m=1$ 的情况，容易发现此时 a_i 是确定的，我们只需要求最小代价。

容易发现操作的执行顺序不影响结果。我们钦定一定是先用机器再用人工，假设执行完所有用机器的区间操作之后的序列为 p_i，那么机器操作的贡献是 $c\sum\max(p_{i+1}-p_i,0)$，人工操作的贡献是 $\sum|p_i-a_i|$。

考虑 dp，设 $f_{i,j}$ 表示考虑到前 i 盆花且 $p_i=j$ 时上式的最小代价，初始 $f_{0,j}=cj$，答案为 $f_{n+1,0}$， 那么有转移：$f_{i,j}=\min\limits_k f_{i-1,k}+c\cdot\max\{j-k,0\}+|j-a_i|$。可以通过 Subtask4。

观察这个式子，发现后面加的东西是凸函数。考虑一种经典维护凸函数的方法(slope trick)：我们维护一个可重集合，表示这个凸函数 “拐弯” 的一些地方。点 x 出现了 k 次代表斜率在这个点变化了 k。

考虑 f 这个函数在中间长什么样：一开始一段斜率为 -1 的直线，中间斜率递增，最后斜率为 $c+1$ 的直线。先不看 $|j-a_i|$ （与 k 无关），只考虑加入 $c\cdot\max\{j-k,0\}$，它带来的影响是前面斜率为 -1 的一段变为 0，斜率为 $c+1$ 的一段变为 c。再加上 $|a_i-j|$ 就是以 a_i 为分界，前面斜率 -1，后面斜率 $+1$。

我们运用之前的方法维护这个凸函数，f 的图像初始是一条直线，即这个集合中初始有

c 个 0。加入 $c \cdot max\{j-k,0\}$ 之后，斜率为 -1 段变为 0，斜率为 $c+1$ 的段变为 c，对应的变化就是删除集合中的最大最小值（注意最开始的时候不能删，因为没有斜率为 -1 或 $c+1$ 的段）。然后再考虑添加 $|a_i-j|$，实际上就是在 a_i 这个位置斜率变化了 2，我们加入两个 a_i 即可。

考虑统计答案。容易发现取出集合中的最小元素 mn，它一定是斜率从 -1 到 0 的一个拐点，这里 a_i 的加入使得最小值增加了 a_i-mn，即将答案加上 a_i-mn。

再考虑 m 1 时的计数，首先 a_i 的和是好计算的，我们只需要考虑 $-mn$ 的部分，容易想到统计每个数作为最小值被删除了多少次。差分一下转化为求 $<x$ 的数被作为最小值删除了多少次。将 i 看成 $[i \ge x]$，统计 0 被删了几次，容易发现只有全是 1 的时候 0 才不会被删除。设 $g_{i,j}$ 表示考虑到前 i 个数，集合中有 j 个 1 的方案数，答案即为 $nk^n-\sum g_{i,c+2}\cdot m^{n-i}$。

考虑 g 的求法，假设第 i 个数有 c_1 个 1，c_0 个 0，那么令 $t=j-[j>0]-[j==c+2]$ 表示有 j 个 1 时去掉最大最小还剩多少个 1，那我们有：$c_1g_{i-1,j} \to g_{i,t+2}, c_0g_{i-1,j} \to g_{i,t}$