persona

签到题。

直接计算是三维偏序，时间复杂度为 $O(n\log^2 n)$ 且常数较大，可能无法通过此题。

考虑降维。如果把 $\max$ 和 $\min$ 分开计算，这没有优化的空间。因此依然需要将所给式子视为一个整体计算。

不妨设 $a \le b \le c \le d$，则 $\max(a,b,c,d)+\min(a,b,c,d)=a+d$。由于我们需要降维且已经给 a,b,c,d 定序，故我们需要使用 $\max(*,*,*),\min(*,*,*)$ 构造一个**对称式**，使其等于 $a+d$。

注意到 $\sum_{sym}\min(a,b,c)=3a+b,\sum_{sym}\max(a,b,c)=c+3d$（$\sum_{sym}$ 为对称求和之意），则 $a+d=(\sum_{sym}\min(a,b,c)+\sum_{sym}\max(a,b,c)-\sum_{sym}a)/2$，成功实现了降维。

当然如果你熟悉 min-max 容斥，你可以跳过以上推导步骤。

接下来的任务是简单的，转化为二维偏序后树状数组维护即可。时间复杂度 $O(n\log n)$。

maze

子任务 1,2

对于静态问题，一个自然的想法是二分答案并验证可行性。建立二分图，左侧对额外残片建一个点，右侧对每个箱子建一个点，每枚额外残片对它能放入的箱子连一条边，则问题转化为每个箱子有容量上限，是否存在完全匹配。使用网络流即可。

每次修改重新二分，时间复杂度 $O(qm\sqrt{m}\log m)$。

子任务 3~5

在初始的图上二分得出答案后，由于每次修改只会改变一条边的容量或者建新点，答案的改变量不会超过 1。这提示我们直接在现有的残量网络上进行修改。

有 A 性质时，要么建立新点，要么箱子对应的连边容量 -1。对于前者，在残量网络上 bfs 看是否能找到从源点到汇点经过该点的路径；对于后者，尝试从该边退流，即从汇点出发寻找一条回到汇点且经过该边的路径。根据 bfs 的结果，相应调整答案及箱子对应边的容量即可。

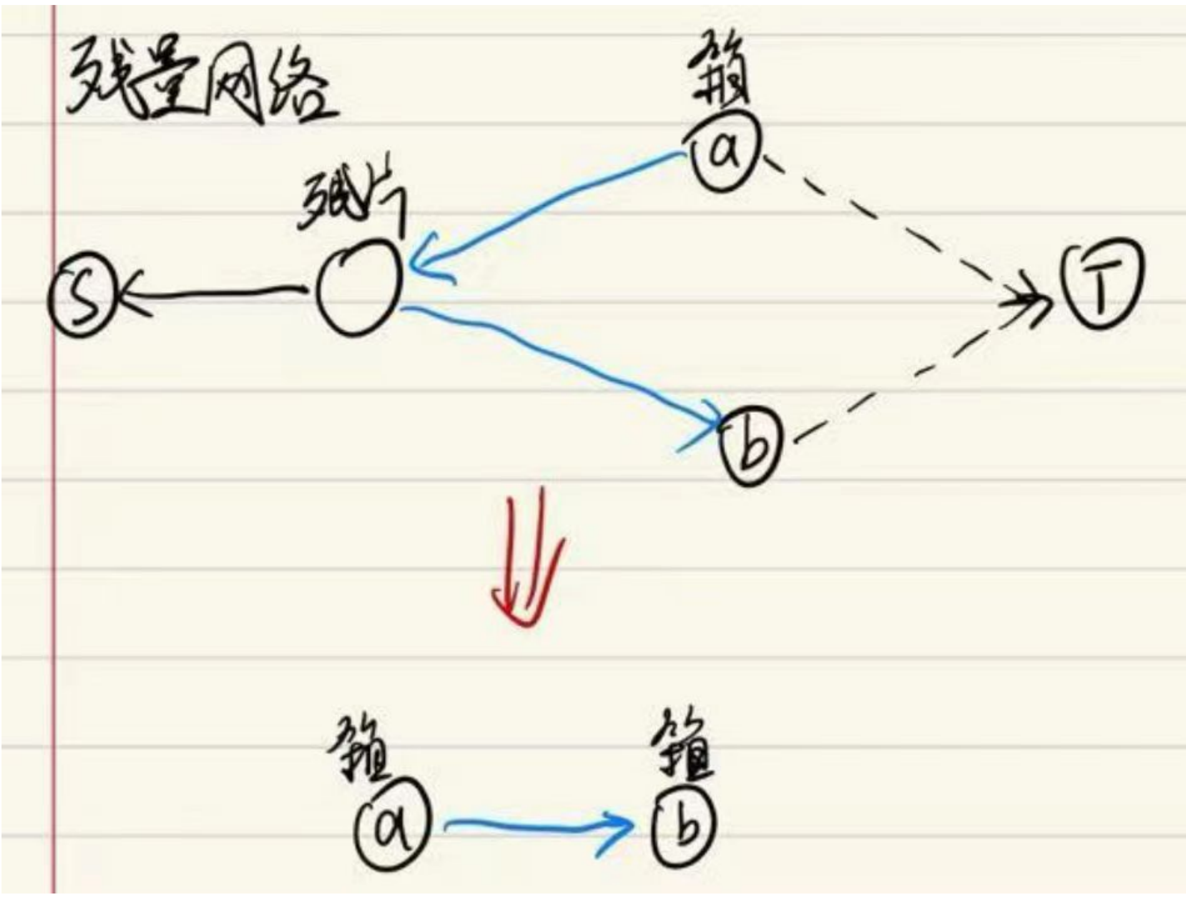
至于 3,4 操作，一种思路是直接线段树分治。但实际上不难发现其和 1,2 操作是相似的，直接在残量网络上进行修改即可，因此不再赘述。

时间复杂度 $O(m\sqrt{m}\log m + qm)$。

子任务 6~10

发现瓶颈在于 bfs。但由于点数、边数为 $O(m)$ 规模，因此需要对图进行压缩。

注意到每枚残片恰能放进 2 个箱子，因此满流时残量网络一定如图所示，故可以压缩：



进一步，初始残片可以视为自环，这样 3,4 操作就转化为了 1,2 操作。

将子任务 3~5 的思路迁移过来，发现 A 性质（仅操作 1）变成了这样：

- 设要加入边 (a, b)。
- 尝试分别从 a, b 出发，寻找一条路径，到达一个尚未放满的箱子。
- 若从 a 出发找到了，则加入边 $a \to b$；若从 b 出发找到了，加入边 $b \to a$。同时，将路径上的边全部反向。
- 若没有找到路径，答案 $+1$ 并更新箱子容量。

上述思路也可以直接通过反悔贪心想到。

对于操作 2，如果是网络流迁移的思路，可以自然地想到如下操作：

- 设要删除边 (a, b)。
- 若存在 $b \to a$ 的边，寻找剩余容量最小的箱子且可达 a；若存在 $a \to b$ 的边，箱子需要可达 b；
- 两者选择剩余容量最小的箱子出发进行退流（过程类似操作 1），更新答案与边的方向。

正确性通过网络流模型不难验证。而如果是通过反悔贪心想到的思路，可能会考虑直接线段树分治解决（子任务 6），而想到上述操作可能需要一点大胆猜想。

由于点的规模降到了 $O(n)$，故采用 `bitset` 优化，一次 bfs 时间复杂度为 $O(\frac{n^2}{\omega})$。

总时间复杂度 $O(\frac{n^2}{\omega}(m+q))$。

flyburg

定义两行的位置差为 $\sum_{i=1}^{c}|p_{l,i}-p_{l-1,i}|$，$p_{l,i}$ 为第 l 行第 i 个 1 的位置，则要求所有位置差 $\leq k$。

首先可以猜想由于要组成一条链，实际上相邻两行的位置差应该不会太大，进而实际所用的列数也应该不会太大。

考虑枚举限定的列数 m。由于 m 较小，使用状压 DP。设状态 (d,S,bl,deg) 表示当前矩阵中度数为 1 的点有 d 个，最后一行选择 1 的位置集合为 S，每个 1 属于联通块的编号数组为 bl、度数数组为 deg。

转移过程比较平凡，在此不多赘述。

状态数量看似很多，实际上有效且可达的状态数并不多。通过 bfs + Hash 表建有限状态自动机，当 $c=8,m=14,k=9$ 时，有效状态数、转移边数量为 $141316,791740$。

同时可以发现，由于题目中 $k>c$ 的限制，在 m 不太大的时候，就可以构造出任意行数的矩阵。例如当 $c=8$ 时，$m=14$ 就可以构造出任意行数的矩阵。

于是可以得到算法：从小到大枚举列数限制 m，对每个 m 分别预处理出 DP 的状态自动机，直至任意行数的矩阵都可以被构造。询问 n,m 时，直接在查询即可。

直接实现，应该可以通过 $n\leq 6$ 的部分。以下为 std 采用的优化手段：

- k 过大时，实际上枚举了相当多无用的状态与转移边。可以考虑在 m 较大、已经可以构造任意矩阵时，对 k 的范围进行限制。
- 枚举转移边时，有很多转移边是无需计算转移后状态就可以知道其不合法的，直接跳过即可。如 1 形成田字、丁字形状等。
- 实际上我们并不需要完整的状态自动机的信息，只保留部分点与边也可能可以构造出合法矩阵。可以设置只枚举前若干条转移边。经过实践，枚举的转移边（指尝试转移，并不一定是合法转移）不超过 4×10^7 时可以保证正确性。

代码实现较为繁琐。