2023“钉耙编程”中国大学生算法设计超级联赛（4）• 题解

2023 年 7 月 27 日

目录

1 Number Table

（出题人太菜被 $O(n^2)/O(n^3)$ 的做法薄纱了）

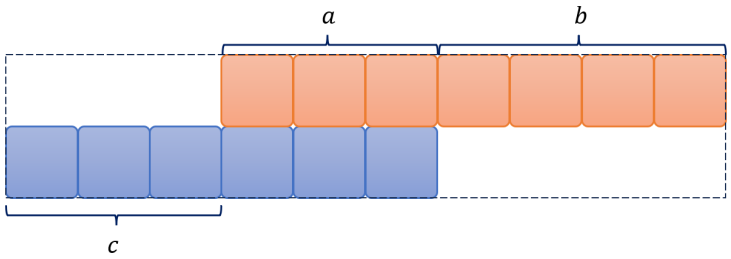
题意相当于每行内有 $n(n-1)/2$ 对不等关系的限制，每列内有 1 对不等关系的限制，总共 n^2 对不等关系限制的情况下，求出有多少种方案满足所有数的异或和为 0 .

假如不考虑不等关系的限制，答案很简单，就是前 $2n-1$ 个位置任选，最后一个位置凑成异或和为 0 即可。

考虑不等限制时，我们假设一个图论模型，图上有 $2n$ 个点对应表中的 $2n$ 个格子，并且图上有 n^2 条边对应 n^2 对不等关系。那么我们的思路就是，用没有不等限制的方案，减去违反限制的方案，违反限制的方案就是说枚举一条边，强制边两边的点相等，但是这样又会多减，所以我们再枚举两条边强制相等，算方案数，加回来，以此类推，是一个经典的容斥原理，我们要做的就是枚举边集的全部子集，然后强制这个子集内的边为相等，在此情况下计算方案数，并且乘上容斥系数之后再全加起来就是答案，容斥系数是 $(-1)^{\text{子集大小}}$。

这样对于一个枚举出来的子集，一张图被划分成若干个连通块，每个连通块上所有点上填的数都是相等的，如果一个连通块的大小是偶数，那么无论填多少都不会影响答案，当连通块大小为奇数时，其等价于一个点。所以如果所有连通块的大小都是偶数，那么方案数就是 $2^{k*\text{连通块个数}}$ ，如果不全是偶数，那方案数就是 $2^{k*(\text{连通块个数}-1)}$ 。

直接这样做复杂度过大，无法通过本题，但是注意到其实我们计算的时候用的是连通块，答案也只和连通块个数以及是不是全是偶数大小有关，因此是有更高效的做法的。首先先统一用 $2^{k*\text{连通块个数}}$ 来计算方案数，最后统一再除以 2^k，这样得到的是假设一定有奇数大小联通块情况的答案，暂时先不考虑纯偶数大小连通块的情况。我们可以 dp，状态 $f(a,b,c)$ 表示一个不完整的表格的方案数，这个不完整的表格有 a 列是完整的，有 b 列仅上半部分保留，下半部分残缺，有 c 列仅下半部分保留，上半部分残缺，如图：



转移时枚举一个连通块，转移时除了连通块本身贡献的 2^k 种方案，还有连通块的容斥系数，这个容斥系数是，所有能使得这个连通块处于连通状态的边集选法对应的容斥系数之和，即 $\sum_{E'\subset E}(-1)^{|E'|}\times[E'\text{确保此连通块能连通}]$ ，此处的 E 是此连通块内部所有可能的边。

这个式子直接求，复杂度很高，所以我们需要更高效的求法，具体如何求，稍后再讲，继续说这个 dp。

转移时要枚举连通块，那么要注意顺序，一定要强制一个特殊点必须在连通块内，这样可以保证每一种拆分方法唯一，不然可能同一个划分方法被多种不同的顺序枚举，这个特殊点的选取方式并不唯一，比如可以强制如果 b 非 0，就让右上 b 个点里面选第一个为特殊点，否则如果 c 非 0，就让左下 c 个点里第一个作为特殊点，再否则就让中间 $2a$ 个点的第一个当特殊点。

枚举连通块时，可以这么枚举，在图中右上的 b 个里面枚举了多少个算入连通块内？在左下的 c 个里面枚举了多少个算入连通块内？中间 $2a$ 个点里面，有多少列是上下两点一起选的？有多少列是只选了上面？有多少列只选了下面？这样总共需要枚举五维，之后再用计算一下有具体的选法数量，注意计算方案数时要考虑到上面强制枚举的一个特殊点。

这样是一个 $O(n^8)$ 复杂度的 dp，直接计算此复杂度会发现无法通过，但是上面各个维度之间有各种相互限制的关系导致常数非常小，仅为 $1/40320$ ，故而能在时限内计算完成，计算出 $f(n,0,0)$ 后，再除以 2^k 就得到了假设有奇数大小连通块情况的答案，如果考虑纯偶数大小连通块的情况，其实也可以如法炮制，再用类似的方法 dp 一遍即可，最后减去之前多算的一部分也就是本来是纯偶数大小连通块的情况，但是却按照 $2^{k*(\text{连通块个数}-1)}$ 来计算的部分即可。

接下来是连通块容斥系数的计算，设 $g(a,b,c)$ 表示一个连通块的容斥系数，且这个连通块的长相也如上图所示。连通这个限制比较难以处理，所以依然使用容斥的方法，用任意图减去不连通图的系数，那么对任意图来说，因为每条边都可选可不选，最终容斥系数相抵消，容斥系数就是 0，除非此图内没有任何边，此时容斥系数为 1。不连通的情况的贡献是枚举一个连通块，剩余部分随意，只要不与这个连通块相连即可。枚举时依然是确定一个特殊点，然后枚举一个连通块要包含这个点，这样就强制枚举出了不连通图，且不重不漏，这里的问题就是，剩余部分的贡献如何计算，还是如上结论，如果剩余部分里面有边，那么贡献就是 0，可以不用考虑，只有几种特殊情况无边，讨论一下即可。这样可以在 $O(n^3)$ 的时间内计算出 g 数组。

另外再提一个常数优化的小技巧：计算假设一定有奇数大小连通块的情况时，分析这个式子的含义，其实是等价于没有异或和为 0 的限制，只考虑不等的限制的情况的方案数再除以 2^k ，这样其实只需要一个 $O(n^3)$ 的 dp 即可。但是纯偶数大小连通块的部分依然要 $O(n^8)$ 计算，这样整体复杂度是不变的，但是常数小了很多（因为纯偶数大小连通块的 dp 本身限制更多，常数更小，所以此优化效果还是非常好的）。

2 Simple Tree Problem

对于每条边的询问，相当于询问只有该子树内的信息的最大值以及全局的信息只扣掉该子树内信息的最大值，对于这两种情况都可以使用线段树合并去处理答案。特别的，处理第二种情况时，子树内维护的是被删除过的值的最大剩余数量，而对于未被删除过的值，可以首先预处理出一棵存有完整信息的权值线段树，在合并过程中残树和整树一起递归，用两棵树的信息一起合并上传，就可以得到当前子树外的真实最大值，能同时递归的原因是残树是整树的子图，时间复杂度是 $O(n\log n)$，空间复杂度是 $O(n\log n)$。

3 Simple Set Problem

首先将元素的值 x 以及所属集合的编号 y 作为二元组 (x,y) 存入数组，然后按照 x 升序排列，之后使用双指针扫描数组（尺取法），当区间内出现了所有编号时更新答案的最小值，时间复杂度 $O(n\log n)$，空间复杂度 $O(n)$。std 运行时间 1.5s，仅关闭流同步，未使用读入优化。

4 Data Generation

考虑记随机变量 X_i 表示 a_i 是否和 i 相等，若相等为 1，不等为 0，则答案是 $n-E(\sum_{i=1}^{n}X_i)$，期望根据线性性可以展开，$E(\sum_{i=1}^{n}X_i)=\sum_{i=1}^{n}E(X_i)=nE(X_1)=n\times P(a_0=0)$ 考虑设 f_k 表示随机 k 次，$a_0=0$ 的概率，需要求 f_m。有 $f_k=\frac{1+(n-1)^2}{n^2}f_{k-1}+\frac{2}{n^2}(1-f_{k-1})=\frac{n-2}{n}f_{k-1}+\frac{2}{n^2}$，一阶递推，可以配凑成等比数列，或者直接矩阵快速幂求解，时间复杂度 $O(T\log\max(n,m))$。

5 Teyvat

考虑对图建出广义圆方树，每次询问一个点集，点集中的点用圆方树上的圆点表示出来，如果点集中的所有点可以作为一点路径 (a,b) 的原图中必经点，那么 (a,b) 在圆方树上可以覆盖点集中的所有点对应点，分以下几种情况讨论：

- 1.$k=1$，预处理统计圆方树上通过有多少圆点点对路径覆盖过这一个点
- 2.$k>1$，这 k 个点在圆方树上不能被一条路径覆盖，答案为 0。
- 3.$k>1$，这 k 个点在圆方树上能被一条路径覆盖， 取出最远的两个点 (u,v)，若 u,v 不为祖孙关系，则答案为 $siz[u]\times size[v]$，其中 $siz[u]$ 表示 u 子树内的圆点数量。
- 4.$k>1$，这 k 个点在圆方树上能被一条路径覆盖，取出最远的两个点 (u,v)，若 u,v 为祖孙关系，假设 u 是祖先，则答案为 $(n-siz[w])\times size[v]$，其中 w 为 u 的孩子，v 的祖先，w 的确定可以用过树剖或者倍增等方法确定。

由于出题和验题在同一个地方失误，导致数据出错，给大家带来了不好的比赛体验，非常抱歉。

6 PSO

一句话题意：给定一张 n 个点的菊花图，所有边长度都是 1，求选两个点的期望距离和最大距离。

先求期望 $E(X)$，长度为 1 的点对有 $n-1$ 种选法，长度为 2 的点对有 $\frac{(n-2)\times(n-1)}{2}$ 种选法，

$$E(X)=\frac{(n-1)\times 1+\frac{(n-2)\times(n-1)}{2}\times 2}{\frac{(n-1)\times n}{2}}=2-\frac{2}{n}$$

再求期望 X_{max}，当 $n=2$ 时，$X_{max}=1$；当 $n>2$ 时，$X_{max}=2$。

7 Guess

$S(n)=\sum_{d|n}\mu(d)\ln(\frac{n}{d})$。

- 若 $n=1$ 则 $S(1)=1$。
- 不妨设 n 的唯一分解式 $n=\prod_{i=1}^{k}p_i^{q_i}$，此时 $k\geq 1$。
  由于 $\mu(d)\neq 0$ 当且仅当 d 中不含平方因子，这些 d 形如 $d=\prod_{i=1}^{k}p_i^{c_i}$，其中 $c_i\in\{0,1\}$。
  此时枚举 n 的因子 d 就等价于枚举 $R=\{1,2,\ldots,k\}$ 的子集，则 $S(n)=\sum_{T\subset R}(-1)^{|T|}(\ln n-\sum_{i\in T}\ln p_i)$，拆成两部分。
- 其中 $\ln(n)\sum_{T\subset R}(-1)^{|T|}=\ln(n)\sum_{i=0}^{k}\binom{k}{i}(-1)^i=\ln(n)((-1)+1)^k=0$（当 $k>1$）.
- 另一边 $-\sum_{T\subset R}\sum_{i\in T}\ln p_i=-\sum_{i=1}^{k}\ln(p_i)\sum_{i\in T\subset R}(-1)^{|T|}=\sum_{i=1}^{k}\ln(p_i)\sum_{j=0}^{k-1}\binom{k-1}{j}(-1)^j$
  若 $k=1$，即 n 为某个素数 p 的整数次幂时，上式为 $\ln(p)$；否则每一项 $\sum_{j=0}^{k-1}\binom{k-1}{j}(-1)^j=((-1)+1)^{k-1}=0$，即 $S(n)=0$。

综上，若 n 为某个素数 p 的正整数次幂时，$S(n)=\ln p$，$e^{S(n)}=p$，否则 $S(n)=0$，$e^{S(n)}=1$。

可以用 Pollard-Rho 算法做大数质因数分解，判断质因子是否只有一种。

8 String and GCD

给定字符串 $S[1:n]$，求 $f(i)=\sum_{j=1}^{i-1}[S[1:j]==S[i-j+1:i]]\times\gcd(i,j)$。

对于 $S[1:j] == S[i-j+1:i]$ 的限制，显然可以对 S 跑 kmp 算法，求出其 $fail$ 树，若使等式成立，则有 $fail$ 树上 j 是 i 的祖先。

对于 $\gcd(i,j)$，我们可以通过莫比乌斯反演推得 $\gcd(i,j)=\sum\limits_{d|i,d|j}\varphi(d)$。

对于所有的 i，预处理所有因子 d，在深搜 $fail$ 树时，记录一个桶 $cnt[d]$，表示 i 的所有祖先 j，满足 $d|j$ 的个数，$f(i)=\sum\limits_{d|i}cnt[d]\times\varphi(d)$。

由于 $i=1,2,...,n$，所有 n 以内正整数正好取遍一次，复杂度为 $1\sim n$ 的因子个数和 $\sum\limits_{i=1}^{n}\lfloor\frac{n}{i}\rfloor=O(n\ln n)$

总时间复杂度 $O(Tn\log n)$

9 WO MEI K

该式子的期望等于求出 $\frac{\text{该式子的总和}}{\binom{n}{k}}$

先考虑 $k=2$ 的情况，相当于所有两个不同的点之间只出现过一条边的个数的和。这是一个经典问题，可以考虑每条边的贡献，答案就是所有边的贡献的和。对于每条边，它的贡献就是有多少个顶点 x 和 y，满足 x，y 之间的路径经过该边，且 x，y 的路径中该边的权值只出现过一次。这个问题可以用虚树、LCT、并查集等方法来解决，这里提供一种 $O(n)$ 的做法：

设 1 为树的根，对于一条边 (u,v)，边权为 c。假设 x 在 u 的子树中，那么 y 只能在除 u 的子树中，这样 x，y 才能经过 (u,v)，x 可放置的位置是所有 x 到 u 之间没有边权为 c 的边的位置，y 可放置的位置是所有 y 到 v 之间没有边权为 c 的边的位置。

对于 x 的位置的数量，可以直接在树上从下往上求出。

对于 y 可放置的位置有两种情况：

- 第一种情况：v 到根结点有权值为 c 的边，对于这种情况可以通过存权值为 c 的最近父节点。
- 第二种情况：v 到根结点没有权值为 c 的边，减去其他子树从根往下的第一个权值为 c 的边的子树即可。

对于 $k>2$ 的情况，由于对 $k=2$ 我们求的是两个点对于某条边的贡献，其他的 $k-2$ 个点可以任选，所以只需要对 $k=2$ 的答案乘上 $\binom{n-2}{k-2}$ 即可。

10 Kong Ming Qi

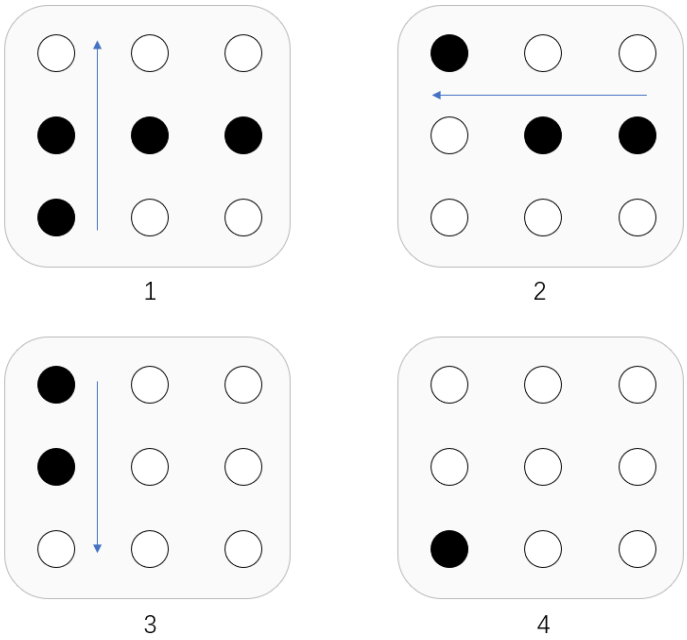
首先不妨设 $n\le m$。分类讨论：

- 当 $n=1$ 时，答案显然为 $\lceil m/2\rceil$。
- 否则当 $m\ge 5$ 时，可以证明一个结论：$n\times m$ 的情况下答案等价于 $n\times(m-3)$ 情况下的答案，证明附后。
- 当然还有一个十分简单的结论，$n\times m$ 和 $m\times n$ 的情况，答案相等。
- 因此，我们可以把所有情形都转换成 $2\le n\le m\le 4$ 的情形，对于这几种情形，可以简单手玩或者爆搜等得到答案。

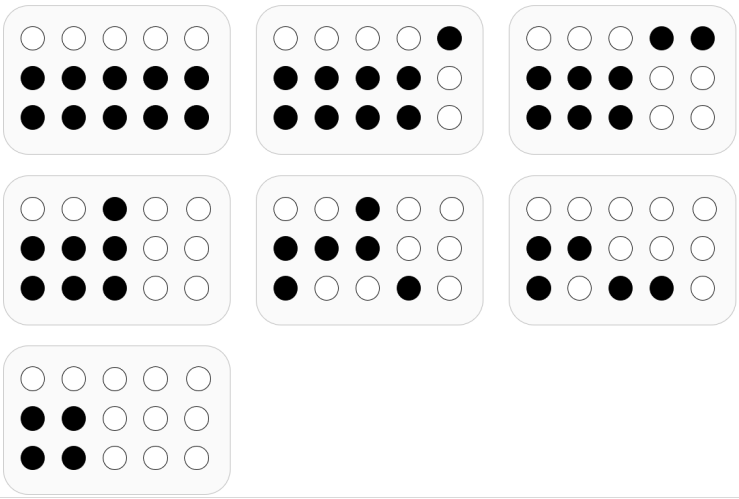
接下来先说明一个三染色引理，来证明一下小情形答案的正确性，将棋盘第一行按 123123123.... 以此类推进行染色，第二行则是 231231231... 以此类推，第三行则是 312312312... 以此类推，第四行则又是 123123123... 以此类推。按这种方式染色后，当我们进行一次操作时，一定是将某两种颜色数量的棋子个数减 1（起始位置和间隔位置），另一种颜色的棋子个数加 1（目标位置的颜色）。

现在假设我们的操作不考虑棋子的具体位置的限制，那么我们可以把操作就当成对颜色数量的操作，当三种颜色的数量为 $1,0,0$ 或者 $2,0,0$ 时，总和无法继续减小，也就意味着答案不能继续变小了。而且，一次操作会改变所有三种颜色数量的奇偶性，也因此，能变成 $1,0,0$ 的状态无法变成 $2,0,0$，反过来也是一样，这也就意味着当我们得到 $2,0,0$ 时，就已经是最优解了，检查一下 $2 \le n \le m \le 4$ 的答案我们会发现，答案只有 1 和 2，且答案为 2 的情况，其三种染色的棋子数量一定是全奇或者全偶，因此无法变成 $1,0,0$，只能变成 $2,0,0$，所以 2 就是最优答案。所以基于这个三染色引理方法给出的下界，我们可以确定答案的正确性。

接下来是是 $n \times m$ 与 $n \times (m-3)$ 答案相等的正确性证明，先定义一个基本操作：



由此基本操作我们可以一次消掉 1×3 区域的棋子（注意前提条件），那么当 $n \ge 3$ 时，利用这个基本操作可以轻松把问题转换成 $n \times (m-3)$ 的情况，当 $n = 2$ 时无法利用这个基本操作，但是可以用如下方法完成



用此方法依然可以把 $2 \times m$ 转换成 $2 \times (m-3)$。这样，我们可以说明答案至少可以小到 $n \times (m-3)$ 这种程度，接下来的一个问题是，答案能不能更小呢？（也就是答案能不能从 2 变成 1 呢？）可以证明

是不行的，依然基于以上三染色引理，假如 $n \times (m-3)$ 时答案为 2，那么其三种染色的棋子数量一定是 $2,0,0$，属于全偶的情况，全偶的情况只能由全偶或者全奇的情况变化过来，那么假如 $n \times (m-3)$ 没有操作时，三种染色的数量分别是 a,b,c，那么 $n \times m$ 的三种染色的数量一定是 $a+n, b+n, c+n$，所以一定也是全奇或者全偶，最终最少也只能变成 $2,0,0$，所以答案不会更小。

11 Circuit

最小环计数, 考虑不重不漏地统计每个合法的环，对于一个环，只有枚举到标号最大的点的时候我们才把方案计入答案。

考虑 $Floyd$ 算法如何实现，在维护 $dis[i][j]$ 表示 i 到 j 的最短距离，$cnt[i][j]$ 表示 i 到 j 的最短路数量。$Floyd$ 有三层循环实现，当最外层循环枚举到 k 时，$dis[i][j]$ 表示除了 i,j，其余点 $\leq k$ 的最短距离，枚举一条边 (k,i)，将此时的 $dis[i][k]$ 拼接上，由 $w(k,i)+dis[i][k]$ 更新最小环即可。

12 a-b Problem

考虑转化一下问题，假如一开始所有石子全在 Bob 手里，那么题中轮到 Alice 取时就拿走一个 Bob 手中的石子，轮到 Bob 取时 Bob 就藏起来一个石子不让 Alice 拿，这样显然与之前的问题等价。然后 Alice 取时自己获得 A_i 的分数，Bob 失去 B_i 的分数，相当于拉开了 A_i+B_i 点分数差，所以 Alice 一定会优先取 A_i+B_i 最大的石子，而 Bob 也会优先藏住 A_i+B_i 最大的石子，所以只需要按 A_i+B_i 排序再轮流按顺序取即可。