XJOI 进阶训练（一）讲评

Mr_Spade

2021.9

T1 是我以前在 LOJ 刷题时刷到的一道质量较高的题目。原题的版本为输出 double，在经过一番思考后，我将它改为了取模输出，同时添加了多组数据。（数据范围也是我添加的）

T2 是我和姜迅驰同学一起原创的一道题目，因为觉得作为计数题质量不错，所以放到了比赛中。

T3 则是一场 LOJ 的 NOI Round 的题目，因为质量不错，所以我搬运到了比赛中。

本场比赛预计的难度是：T1 难度介于 NOI 的 T1 和 T2 之间，T2 和 T3 则对标 NOI 的 T2，T3 难度。

从模拟情况来看，大家需要继续努力提升水平。一名成熟的 NOI 选手本场比赛的参考目标分数是 200+；一名强基签约选手的参考目标分数是 150+。

本场比赛预计的难度是：T1 难度介于 NOI 的 T1 和 T2 之间，T2 和 T3 则对标 NOI 的 T2，T3 难度。

从模拟情况来看，大家需要继续努力提升水平。一名成熟的 NOI 选手本场比赛的参考目标分数是 200+；一名强基签约选手的参考目标分数是 150+。

另一方面，由于本场比赛的难度相对较高，如果没有取得理想成绩，大家正常对待即可，仍要对自己的实力和进步的可能有信心！

题意简述：给定 n 张牌，点数至多 m 种，每次随机选取两张牌 A，B，将 A 的点数变为 B 的点数。求所有牌变为同一点数的期望步数。

题意简述：给定 n 张牌，点数至多 m 种，每次随机选取两张牌 A，B，将 A 的点数变为 B 的点数。求所有牌变为同一点数的期望步数。

得分情况：部分同学在比赛时得到了 10 分或 30 分，部分同学没有得分，低于预期。其实，本题的前 30 分通过一些并不繁琐的化简式子的步骤就能得到，大家还是要积极锻炼推理式子的能力，不要害怕在一场比赛中推导一些看上去有些难度的式子，或许实际上它比你想象的简单。

欢迎吐槽！

部分分做法

欢迎发言！

满分做法

欢迎发言！

初步观察，发现最坏情况下操作次数可能为无穷，有些棘手。

初步观察，发现最坏情况下操作次数可能为无穷，有些棘手。

尝试先从部分分获取一点思路。第一档部分分为 $n \leq 1000, m = 2$。$m = 2$ 看上去更特别，探究一下这种情况下的性质。

此时，所有牌只有两种不同的点数，假设第一种点数有 k 张，那么第二种点数必然有 $n - k$ 张，状态数是 $O(n)$ 的！

自然可以尝试着这样设计量：f_i 表示当前第一种点数有 i 张牌，那么操作的期望步数是多少。边界条件显然有 $f_0 = f_n = 0$，否则，计算出第一种牌的数目 $+1, -1$ 或保持不变的概率，就可以得到方程：

$$f_i = \frac{i(i-1) + (n-i)(n-i-1)}{n(n-1)}(f_i + 1) + \frac{i(n-i)}{n(n-1)}(f_{i-1} + 1) + \frac{i(n-i)}{n(n-1)}(f_{i+1} + 1) \tag{1}$$

]

将所有的 $+1$ 直接求和，式子右侧的 f_i 转移到左边，即有：

$$\frac{2i(n-i)}{n(n-1)} f_i = 1 + \frac{i(n-i)}{n(n-1)} f_{i-1} + \frac{i(n-i)}{n(n-1)} f_{i+1} \tag{2}$$

继续转化式子，两边除以频繁出现的因子 $\frac{i(n-i)}{n(n-1)}$：

$$2f_i = \frac{n(n-1)}{i(n-i)} + f_{i-1} + f_{i+1} \tag{3}$$

移项即可得到 f_i 的递推式：

$$f_{i+1} = 2f_i - f_{i-1} - \frac{n(n-1)}{i(n-i)} \tag{4}$$

为了通过递推式求解 f_i，我们可以假设 $f_1 = x$，于是，所有 f_i 可以表示为 x 的线性函数。最后递推至 $f_n = 0$ 时，可以计算出 x，于是所有的 f_i 都可得到。这样就解决了第一档部分分，得分 10 分。

继续考虑下一档部分分，将 $m=2$ 的限制去除了。不过，$m=2$ 的做法是否能给我们一点启发呢？

不难考虑到讨论最终状态会是什么。最终状态下点数只有一种，但对于具体是哪一种没有限定，有 m 种可能。如果枚举每一种可能会怎么样？假设枚举最后的点数都变为 x，那么想要讨论这种情况下的步数，所有的牌可以只分为两类：点数是 x 和不是 x，这不是和 $m=2$ 的部分有些相似吗？

继续这个思路，我们尝试计算最后的点数都变为 x 的情况的贡献。这里的“贡献”的含义还需要仔细思考一下，我们知道，所谓的步数期望，可以粗略理解为枚举每一种可能的过程，将这种过程发生的概率与这种过程的步数相乘，再求和就是期望步数；因此，要计算“贡献”，就是把所有过程中“最后的点数都变为 x”的那些过程的概率乘以步数求和。

我们仍然使用 f_i，表示若当前点数为 x 的牌有 i 张，那么产生的贡献的值是多少。边界条件仍然有 $f_0=f_n=0$，不过，此时 $f_0=0$ 的原因是已经不可能变为点数全为 x 的情况，故过程数目为 0；$f_n=0$ 的原因则是唯一的过程就是不需要任何操作，步数为 0。

仍然采用和 10 分部分分相似的方法，讨论牌的数目 $+1, -1$ 或保持不变的概率。然而，这里有一点需要特别注意：由于我们统计的不再是期望而是贡献，因此对于那些最终不变为点数全 x 的过程，我们想统计的是 0，而不是它的概率乘以步数。因此，之前式子中的那一项“$+1$”此时就不对了：并不是所有情况下，这一步都会被计算进答案，而是只有最终到达了点数全为 x 的状态，这一步的“1”才能产生贡献！

仍然采用和 10 分部分分相似的方法，讨论牌的数目 $+1,-1$ 或保持不变的概率。然而，这里有一点需要特别注意：由于我们统计的不再是期望而是贡献，因此对于那些最终不变为点数全 x 的过程，我们想统计的是 0，而不是它的概率乘以步数。因此，之前式子中的那一项“$+1$”此时就不对了：并不是所有情况下，这一步都会被计算进答案，而是只有最终到达了点数全为 x 的状态，这一步的“1”才能产生贡献！

看来我们需要引入一个新变量：p_i 表示若当前点数为 x 的牌有 i 张，那么最终变为全是 x 的概率是多少。此时可以写出 f_i 的方程：

$$f_i=\frac{i(i-1)+(n-i)(n-i-1)}{n(n-1)}(f_i+p_i)+\frac{i(n-i)}{n(n-1)}(f_{i-1}+p_{i-1})+\frac{i(n-i)}{n(n-1)}(f_{i+1}+p_{i+1}) \quad (5)$$

不过此时，我们需要先求解 p_i。首先，边界条件有 $p_0=0,p_n=1$。

p_i 的方程和 f_i 是类似的：讨论牌的数目 $+1, -1$ 或保持不变的概率，就有：

$$p_i = \frac{i(i-1)+(n-i)(n-i-1)}{n(n-1)}p_i + \frac{i(n-i)}{n(n-1)}p_{i-1} + \frac{i(n-i)}{n(n-1)}p_{i+1} \tag{6}$$

移项化简可以得到：

$$p_{i+1} = 2p_i - p_{i-1} \tag{7}$$

于是可以故技重施：假设 $p_1 = x$，通过递推和边界条件反解出 x，从而得到所有 p_i。得到 p_i 之后，再次用相同方法得到 f_i。最终，答案就是 $\sum_i f_{a_i}$。

最终就在 $O(n)$ 的复杂度内解决了问题，得分 30 分。

接下来，n 的值突然提升到 10^9，原先的方法不太奏效了。

不过，f_i 和 p_i（尤其是 p_i）的递推式看上去还挺优美的，能不能直接手算呢？

接下来，n 的值突然提升到 10^9，原先的方法不太奏效了。

不过，f_i 和 p_i（尤其是 p_i）的递推式看上去还挺优美的，能不能直接手算呢？

我们仍设 $p_1 = x$，代入计算，可以得到 $p_2 = 2x, p_3 = 3x, \ldots$，规律非常明显，我们直接就可以验证 $p_i = ix$。

于是，根据 $p_n = nx = 1$，我们得到了 $x = \frac{1}{n}$，故 $p_i = \frac{i}{n}$。这样，f_i 的方程也不用再借助 p_i 了：

$$f_i = \frac{i(i-1)+(n-i)(n-i-1)}{n(n-1)}(f_i+\frac{i}{n})+\frac{i(n-i)}{n(n-1)}(f_{i-1}+\frac{i-1}{n})+\frac{i(n-i)}{n(n-1)}(f_{i+1}+\frac{i+1}{n}) \quad (8)$$

仍然把 $\frac{x}{n}$ 的部分拿出来直接求和，可以发现结果是优美的 $\frac{i}{n}$，再将 f_i 移项，就得到：

$$\frac{2i(n-i)}{n(n-1)}f_i = \frac{i}{n} + \frac{i(n-i)}{n(n-1)}f_{i-1} + \frac{i(n-i)}{n(n-1)}f_{i+1} \quad (9)$$

$$\frac{2i(n-i)}{n(n-1)}f_i = \frac{i}{n} + \frac{i(n-i)}{n(n-1)}f_{i-1} + \frac{i(n-i)}{n(n-1)}f_{i+1} \tag{10}$$

对上式两边除以 $\frac{i(n-i)}{n(n-1)}$，再移项就得到：

$$f_{i+1} = 2f_i - f_{i-1} - \frac{n-1}{n-i} \tag{11}$$

真优美！我们假设 $f_1 = x$，代入计算，可以得到 $f_2 = 2x - \frac{n-1}{n-1}, f_3 = 3x - \frac{2(n-1)}{n-1} - \frac{n-1}{n-2}, f_4 = 4x - \frac{3(n-1)}{n-1} - \frac{2(n-1)}{n-2} - \frac{n-1}{n-3}, \ldots,$ 规律同样容易发现，验证之后就可以得到：

$$f_i = ix - \sum_{j=1}^{i-1} \frac{(i-j)(n-1)}{n-j} \tag{12}$$

$$f_i = ix - \sum_{j=1}^{i-1} \frac{(i-j)(n-1)}{n-j} \tag{13}$$

现在可以利用边界条件 $f_n = 0 = nx - \sum_{j=1}^{n-1} \frac{(n-j)(n-1)}{n-j} = nx - (n-1)^2$，得到 $x = \frac{(n-1)^2}{n}$，就得到了 f_i 的公式：

$$f_i = \frac{i(n-1)^2}{n} - \sum_{j=1}^{i-1} \frac{(i-j)(n-1)}{n-j} \tag{14}$$

进行进一步的化简：

$$f_i = \frac{i(n-1)^2}{n} - \sum_{j=1}^{i-1} \frac{(n-j)(n-1)}{n-j} + \sum_{j=1}^{i-1} \frac{(n-i)(n-1)}{n-j} \tag{15}$$

$$f_i = \frac{i(n-1)^2}{n} - (i-1)(n-1) + (n-i)(n-1) \sum_{j=1}^{i-1} \frac{1}{n-j} \tag{16}$$

$$f_i = \frac{i(n-1)^2}{n} - (i-1)(n-1) + (n-i)(n-1)\sum_{j=1}^{i-1}\frac{1}{n-j} \tag{17}$$

现在，我们想求解 $O(m)$ 个 f_i，不过，f_i 的表达式中尚带有一个倒数的区间和的项。

$$f_i = \frac{i(n-1)^2}{n} - (i-1)(n-1) + (n-i)(n-1)\sum_{j=1}^{i-1}\frac{1}{n-j} \tag{17}$$

现在，我们想求解 $O(m)$ 个 f_i，不过，f_i 的表达式中尚带有一个倒数的区间和的项。

说起区间和，我就想起前缀和加差分。于是它可以转化为两个调和级数的差，我们只要求解 $O(m)$ 个调和级数即可。

$$f_i = \frac{i(n-1)^2}{n} - (i-1)(n-1) + (n-i)(n-1)\sum_{j=1}^{i-1}\frac{1}{n-j} \tag{17}$$

现在，我们想求解 $O(m)$ 个 f_i，不过，f_i 的表达式中尚带有一个倒数的区间和的项。

说起区间和，我就想起前缀和加差分。于是它可以转化为两个调和级数的差，我们只要求解 $O(m)$ 个调和级数即可。

说起求解调和级数，我就想起分段打表。本地预处理所有调和级数，每隔 T 个打一个表，那么求解调和级数的复杂度就降低到 $O(T)$（需要利用线性求逆元的技巧）。同时，我们要注意 $\frac{n}{T}$ 不超过代码长度限制。

有一档部分分中 $m \leq 200$，那么利用分段打表就可以直接通过了。得分 60 分。

不过，极限数据范围有 $m \leq 10^5$，每一次都分段打表求解调和级数太耗时了。f_i 的式子看上去已经不能简化了，怎么降低复杂度呢？

我们观察到一个事实：f_i 的表达式中的倒数的区间和的项，实际上可以看成以 $\frac{1}{n-1}$ 为右端点的后缀和。那就是说，通过预处理后缀和，可以 $O(k)$ 得到 f_i 的前 k 项。

还有一个重要的事实是，我们求解的 m 个 f_{a_i}，a_i 不是随意分布在 $[1, n]$ 中的，而是满足 $\sum a_i = n \leq 10^9$，直觉上来说，比较小的 a_i 可能很多，似乎有优化的空间！

这里有一个类似于分块的思想：我们分大小讨论，设置一个阈值 T'，预处理前 T' 个 f_i，对于 $a_i \leq T'$ 的情况直接得到答案，否则分段打表 $O(T)$ 求解，这样的复杂度如何呢？

关键点在于，由于 $\sum a_i = n$，因此实际上大于 T' 的项不会超过 $\frac{n}{T'}$ 项，如果 T 较大，那么用分段打表求解的次数是可以被有效限制的！

现在，我们的复杂度优化到了 $O(T' + T\frac{n}{T'})$，首先测试代码长度的上限得到 T 的值，再通过均值不等式求出最优的 T' 即可。经过计算可以发现能够通过所有数据，得分 100 分。

本题应用到的一个重要思想是，对于一个支持分段打表求解的递推数列，如果题目要求求出其中若干项，且对这些项的下标之和有所限制的话。就可以采用本题的方法：递推出一定长度的前缀，对于落在这个前缀内的询问直接给出答案，否则用分段打表求解答案。通过平衡阈值，可以结合分段打表和递推的优势，有效降低求解的复杂度。

本题应用到的一个重要思想是，对于一个支持分段打表求解的递推数列，如果题目要求求出其中若干项，且对这些项的下标之和有所限制的话。就可以采用本题的方法：递推出一定长度的前缀，对于落在这个前缀内的询问直接给出答案，否则用分段打表求解答案。通过平衡阈值，可以结合分段打表和递推的优势，有效降低求解的复杂度。

实际应用中，如果允许离线查询，直接排序询问可以得到复杂度相同但常数更小的方法，有的学弟就是用此方法订正得到满分的。

题意简述：求满足所有连通块中标号最小值的最大值为 t 的 n 个点的无向连通图的数量，对 $t = 1, 2, \ldots, n$ 求答案。

题意简述：求满足所有连通块中标号最小值的最大值为 t 的 n 个点的无向连通图的数量，对 $t = 1, 2, \ldots, n$ 求答案。

两名同学在比赛时获得了 40 分，其余同学则没有得分，略低于预期。比较意外的是，没有得分的同学都没有写出基本没有难度的 10 分部分分。在正式的比赛中，大家要养成有分必争的习惯，到了比赛的最后时刻，即使你对已有的成绩不满意，也应该尽量把能得到的低级部分分全部得到，除非你正在调试比较高分的代码。有时，巨大的排名差距反而是在简单的部分分上体现的，大家一定不要放弃得到简单部分分的机会。

欢迎吐槽！

部分分做法

欢迎发言！

说起最小值的最大值，大家应该能联想到许多二分答案的题目。最小值的最大值之所以和二分答案有密切关系，是因为如果你想知道这个值能否 $\leq t$，那么只要让每一个最小值都 $\leq t$ 即可，从而一定程度上将整体分离为部分。

仿照这个思想，我们可以先求出最小值的最大值 $\leq t$ 的无向图数目 f_t，之后，只要进行一次差分就可以得到答案。而最小值的最大值 $\leq t$，只需要每个连通块都包含至少一个 $\leq t$ 的点就可以了。

不过，“至少一个”想要满足有些困难，我们更喜欢“一个也没有”。那么不妨这么思考：我们从所有的方案中，减去不合法的方案。所谓不合法，就是存在若干只由 $> t$ 的点组成的连通块，这看上去更好解决一些。

一个首要的问题是，我们应该从什么角度对不合法方案进行分类，以便我们枚举。一个简单的方法是枚举组成不合法连通块的点是哪些，当然，我们随即可以发现具体选哪些点是无所谓的，关键在于点的数目。于是枚举 k 表示点的数目，我们就有：

$$f_t = g_n - \sum_{k=1}^{n-t} \binom{n-t}{k} g_k f_{t,n-t-k} \tag{18}$$

其中，$g_n = 2^{\binom{n}{2}}$ 表示 n 个点的无向图数目。枚举非 0 的 k，再从 $n-t$ 个 $> t$ 的点中选出具体的 k 个点，这些点随意组成连通块，方案为 g_k。随后，由于我们要求独立组成连通块的 $> t$ 的点恰好是这些，因此对于剩下的 $n-t-k$ 个 $> t$ 的点，它们只能存在于包含至少一个 $\leq t$ 的点的连通块中，于是，问题被转化为一个类似的子问题：$n-k$ 个点中，有 $n-t-k$ 个点不能单独组成连通块，求方案数，把这个记为 $f_{t,n-t-k}$，于是 f_t 可以记为 $f_{t,n-t}$。

考虑到 $g_0 = 1$，我们甚至可以将 $f_{t,n-t}$ 改写为 $g_0 f_{t,n-t-0}$，再将右侧的和式转移到左侧拼成完整的卷积式：

$$\sum_{k=0}^{n-t} \binom{n-t}{k} g_k f_{t,n-t-k} = g_n \tag{19}$$

于是我们有：

$$\sum_{k=0}^{n-t} \frac{g_k}{k!} \frac{f_{t,n-t-k}}{(n-t-k)!} = \frac{g_n}{(n-t)!} \tag{20}$$

那么，令 $G = \sum_{k\geq 0} \frac{g_k}{k!}$，$F_t = \sum_{k\geq 0} \frac{f_{t,k}}{k!}$，$\hat{G} = \sum_{k\geq 0} \frac{g_{k+t}}{k!}$，就有 $G * F_t = \hat{G}$，于是 $F_t = \frac{\hat{G}}{G}$，利用多项式求逆，可以 $O(n \log n)$ 求出 $[x^{n-t}]F_t$，也就是 f_t。从而得到所有答案的复杂度是 $O(n^2 \log n)$，得分 40 分。

考虑到 $g_0 = 1$，我们甚至可以将 $f_{t,n-t}$ 改写为 $g_0 f_{t,n-t-0}$，再将右侧的和式转移到左侧拼成完整的卷积式：

$$\sum_{k=0}^{n-t} \binom{n-t}{k} g_k f_{t,n-t-k} = g_n \tag{19}$$

于是我们有：

$$\sum_{k=0}^{n-t} \frac{g_k}{k!} \frac{f_{t,n-t-k}}{(n-t-k)!} = \frac{g_n}{(n-t)!} \tag{20}$$

那么，令 $G = \sum_{k\geq 0} \frac{g_k}{k!}$，$F_t = \sum_{k\geq 0} \frac{f_{t,k}}{k!}$，$\hat{G} = \sum_{k\geq 0} \frac{g_{k+t}}{k!}$，就有 $G * F_t = \hat{G}$，于是 $F_t = \frac{\hat{G}}{G}$，利用多项式求逆，可以 $O(n \log n)$ 求出 $[x^{n-t}]F_t$，也就是 f_t。从而得到所有答案的复杂度是 $O(n^2 \log n)$，得分 40 分。

满分做法的思路将在明天的容斥原理讲义中呈现。

题意简述：给定排列的每个相邻元素的大小关系，求满足要求的排列数目。

题意简述：给定排列的每个相邻元素的大小关系，求满足要求的排列数目。

所有同学都在比赛中获得了 40 分，高于预期，大家基本的 dp 功底还不错。希望在后续的训练中大家的水平能够进一步进步，最终达到能在比赛中当场秒掉此题的能力。

欢迎吐槽！

部分分做法

欢迎发言！

大家都给出了较为一致的利用 dp 直接计数的 $O(n^2)$ 做法。不过，这个做法比较明显地没有进一步优化的空间了。我们可以尝试另一个方向——间接计数法。

大家都给出了较为一致的利用 dp 直接计数的 $O(n^2)$ 做法。不过，这个做法比较明显地没有进一步优化的空间了。我们可以尝试另一个方向——间接计数法。

间接计数法的思路将在明天的容斥原理讲义中呈现。

综合来看这场比赛，应该采取怎样的比赛策略呢？

首先可以关注到 T2 和 T3 都有一档难度很低但分数也很低的部分分。这样的部分分显然不会带来太大的启发，因此，在比赛初期，可以选择不写这些部分分，专注于思考更高的部分分。在比赛后期，如果这两道题依然没有什么进展，那么这些部分分一定要及时补上，保证自己的底盘。

其次，T2 题意在化简之后非常简洁，看上去非常像一道经典问题。不过，如果这样的题目出现在了比赛中，那么它通常不是一道做法已经为大家所熟悉的题目，更可能是一道需要较复杂推导的难度较高的题目（否则，出现模板题并不有利于选拔），因此不能对此盲目自信，花大量时间进行推导。

另外，T1 的 $m = 2$ 的部分看上去是非常具有启发性的，因此积极思考这一部分分的做法是解决 T1 的关键。T1 的部分分基本是循序渐进的，因此通过 $m = 2$ 找到突破口以后，选手可以根据自己的能力向前获取部分分。

而 T2 和 T3 则比较类似，都有一档 $O(n^2)$ 级别的部分分和 poly log 的部分分。有时，思考 $O(n^2)$ 做法可能有利于直接优化至 $O(n \log n)$，不过这两道题都存在着较好思考但难以继续优化的 $O(n^2)$ 级别做法，大家应该已经发现了这一点。

T2 和 T3 这样存在着较好思考但难以继续优化的 $O(n^2)$ 级别做法的特点往往也是许多难题的难点所在，也是大家需要突破的地方。在一个做法陷入瓶颈以后，大家能否找到新的思路继续做题是直接和大家的算法竞赛水平相关的。突破这一点并不容易，从 T2、T3 得分从 40 分到 100 分的飞跃就可以看出来。不过，决定大家能否成为优秀的算法竞赛选手的关键也在于此。如果在经过一段时间的训练后，大家能够在比赛中在 T2 和 T3 的某一道题上找到突破口，从而取得满分，同时将另一题的平方级别打满，那么可以说大家就已经基本完成了走向成熟的算法竞赛选手的任务。