IOI2015 中国国家集训队第一次作业：试题泛做

VFleaKing

January 22, 2015

目录

1 Codeforces 235C：Cyclical Quest

【大意】

给一个长度为 n 的字符串 s，每次询问一个串 q 问 s 有多少个子串与 q 循环同构。

【解法】

对 s 建立后缀自动机，然后把 s 复制两份拼起来带进去跑。不过这样有重复计数，所以当走完一个循环节的时候停下来就行了。

时间复杂度：$O(n\sigma + L_q)$

空间复杂度：$O(n\sigma + L_q)$

2 Codeforces 235E：Number Challenge

【大意】

给定 a, b, c，求

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} d(ijk)$$

其中 d 是除数函数。$a, b, c \leq 2000$。

【解法】

使劲推就行了。前两维暴力，求出 $f(m)$ 表示前两维有多少个 i 和 j 乘起来等于 m。

用 $[P]$ 表示当 P 为真时为 1 否则为 0。

$$\begin{aligned}
&\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} d(ijk) \\
=\ &\sum_{i=1}^{ab} f(i)\sum_{k=1}^{c} d(ik) \\
=\ &\sum_{d\geq 1}\sum_{i=1}^{ab} f(i)\left\lfloor\frac{c\gcd(i,d)}{d}\right\rfloor \\
=\ &\sum_{d\geq 1}\sum_{i=1}^{ab}\sum_{g\mid d}[\gcd(d,i)=g]f(i)\left\lfloor\frac{cg}{d}\right\rfloor \\
=\ &\sum_{g\geq 1}\sum_{d\geq 1}\sum_{i=1}^{\lfloor ab/g\rfloor}[\gcd(d,i)=1]f(ig)\left\lfloor\frac{c}{d}\right\rfloor \\
=\ &\sum_{g\geq 1}\sum_{e\mid g}\sum_{d\geq 1}\sum_{i=1}^{\lfloor ab/g\rfloor}\mu(e)f(ig)\left\lfloor\frac{c}{ed}\right\rfloor \\
=\ &\sum_{g\geq 1}\left(\sum_{i=1}^{\lfloor ab/g\rfloor} f(ig)\right)\left(\sum_{e\mid g}\mu(e)\sum_{d\geq 1}\left\lfloor\frac{c}{ed}\right\rfloor\right)
\end{aligned}$$

分开的这两部分。求左边部分使用暴力本身就是 $O(ab\log(ab))$ 的，无须优化。求右边部分我们只有 $O(ab)$ 使用欧拉筛法预处理 μ，枚举约数 e 算对每个 g 的贡献，这样就是 $O(c\log c)$ 的。最后 $O(ab)$ 合并起来就行了。

时间复杂度：$O(ab\log(ab)+c\log c)$

空间复杂度：$O(ab+c)$

3 Codeforces 238D：Tape Programming

【大意】

有一种编程语言，在这个语言下一个程序是由数字和 “<”,“>” 构成的非空串。程序运行时有一个指针。最开始指针的指向最左字符，移动方向为向右。

我们重复以下操作直到指针指向串外：

1. 如果指针指的位置是一个数字，输出这个数字，然后将指针沿着原来移动方向移动，同时将原来的数字减一。如果原来的数字为 0 则删掉；
2. 如果指针指的位置是 “<” 或 “>”，那么指针的移动方向对应修改，接着沿着新的移动方向移动。如果新的位置也是 “<” 或 “>”，则删除原来的 “<” 或 “>” 字符。
3. 指针指向了串外就结束运行。

现在我们有一个由 n 个由 “<”, “>” 和数字构成的串 s，你需要回答 q 个询问。每个询问会给你两个数 l，r，如果把区间 $[l,r]$ 内的 s 看成一个单独的程序，问每个数字会被输出多少次。

【解法】

应该能注意到这个编程语言有特殊性质即指针的移动是连续的，这意味着假设我们在开头处放置足够多的 “>”，那么单独取一个区间出来执行时的程序一定是完整地执行整个代码的程序的一部分。

于是我们可以做一个前缀和。$f[i][d]$ 表示第 i 个字符在准备第一次访问到的时候数字 d 输出了多少次，$g[i][d]$ 表示第 i 个字符在指针准备从此处向左移走的时候数字 d 输出了多少次。然后查询一段区间时，程序肯定从左端点开始执行，但有可能从左边或者右边出去。于是我们可以比较下左端点的 f 和右端点的 g 的时间，然后安心地使用前缀和相减得到结果。

假设数字有 d 种（本题中 $d = 10$），那么程序执行次数肯定是 $O(nd)$ 的。我们可以用链表纯模拟求出 f 和 g，那么预处理就是 $O(nd)$ 的。

时间复杂度：$O(nd + qd)$

空间复杂度：$O(nd)$

4 Codeforces 238E：Meeting Her

【大意】

有一个 n 个结点的有向图，边权均为 1。Urapl 想从 a 出发去 b。有 p 个公交车公司。在每一秒的开始，第 i 个公司的公交车随机选择一条从 s_i 到 t_i 的最短路径然后走这条路径。如果一个公交车经过 Urpal 所在的交叉点，则 Urpal 可以上这辆公交车，他可以在中途任意一个结点下车。

在任何时刻 Urpal 只知道他自己的位置和约会地点。当他上了公交车时他只知道这辆公交车属于第几个公司。当然 Urpal 知道城市地图和每个公司的 (s_i, t_i)。

求最坏情况下他需要乘坐公交车的次数。

【解法】

先 Floyd 一遍，然后对于每个公交车每个 k 求出走 k 站可能停在的位置。接着我们依次求出最坏情况下乘坐 $0, 1, 2, \ldots$ 次公交车能到达 b 的位置，每次求的方法就是扫一扫每条公交线路看看能作出什么贡献。

时间复杂度：$O(pn^3)$

空间复杂度：$O(pn + n^2)$

5 Codeforces 240E：Road Repairs

【大意】

有一个有向图边权为 0 或 1，求最小树形图。

【解法】

使用朱刘算法，用左偏树和并查集搞搞就能做到 $O(n + m \log n)$。

不过我不是写的传统的朱刘算法，我是依次访问每个未访问的结点，一步一步找最小边往上爬，碰到环就缩一下。并查集维护被缩起来的结点，左偏树维护连进结点的边。

时间复杂度：$O(n+m\log n)$

空间复杂度：$O(n+m)$

6 Codeforces 240F：Torcoder

【大意】

给出一个长度为 n 的字符串，以及 m 个操作。每个操作给出一个区间 $[l,r]$，你需要将 $[l,r]$ 中的字符重排使得这个子串成为一个回文串且字典序最小。如果某个操作不可能排出回文串，则直接无视这个操作。你需要输出经过 m 次操作后的字符串。

【解法】

直接用线段树维护区间内字符个数，以及使用区间覆盖的标记。每次查询区间内每个字符出现多少次然后按顺序重排就行了。

时间复杂度：$O(n\sigma+m\log n\sigma)$

空间复杂度：$O(n\sigma)$

7 Codeforces 241B：Friends

【大意】

给出 n 个非负整数 a_i，两两配对成无序对，每对 (a_i,a_j) 的好看度为 $a_i\oplus a_j$。请选出 m 对使得好看度之和最大。

【解法】

设 $w=\max\{\log_2 a_i\}$

显然我们要求的是好看度前 m 大的数对的好看度之和。

如果问题是请把好看度大于 l 的数对的好看度之和，是很好做的。而求出最小的 l 使得好看度大于 l 的不超过 m 对，也是很好做的。所以我们把这个问题分两步。

一个性质是，$a\oplus b>l$ 等价于：假设 $a\oplus b\oplus l$ 写成二进制时从高到低第一个非零位是第 i 位，那么 l 的第 i 位为 1。

我们把所有的 a_i 扔到一棵 Trie 树上去。首先我们来求 l。这显然只要预处理每个子树对应着多少个 a_i，然后在 Trie 树上顺着走一走就可以了。从大到小枚举每个数位，每次都尝试把 l 的那个数位变为 1，看满足条件的个数是否超过 m。这样就能在 $O(nw)$ 时间内求出 l 以及大于 l 的个数 c。

接下来我们求大于 l 的数对的好看度之和。还是老样子，我们枚举 a_i，然后在 Trie 树上走走，就能把求和问题分解成若干个 Trie 树子树中所有元素与 a_i 的异或值之和。我们只要对于每个子树求出对于每个数位 i，有多少个数的第 i 位为 1 即可，这样是 $O(nw^2)$ 的空间。当然我们可以先对 a 排序，然后一个 Trie 树子树一定对应着一个区间。我们只要对每个数位做 a 中的前缀和，并且对于每个子树记录对应的 a 的区间左端点和右端点，就能把空间降为 $O(nw)$。当然时间复杂度大概没救了，只能对于每个子树 $O(w)$ 扫一下每个数位。这样我们就能在 $O(nw^2)$ 时间内求出总和 s。

最后答案就是 $s+l\times(m-c)$。

时间复杂度：$O(nw^2)$

空间复杂度：$O(nw)$

8 Codeforces 241D：Numbers

【大意】

给一个 1 到 n 的排列 a 和一个素数 p，构造一个子序列，使得元素异或和为 0 且首位相接形成的大整数是 p 的倍数。$n, p \leq 50000$。

【解法】

如果数据范围很小，我们显然可以有一个 $O(n^2p)$ 的 dp 算法可以解决。但是现在数据范围很大，那么我们无视排列中所有 ≥ 32 的元素，只剩下 $1 \sim 31$ 然后跑这个 dp。

为什么这样是靠谱的？因为 $1 \sim 31$ 组成的异或和为 0 的集合共有 67108864 个，而当 $p \neq 2$ 或 5 时，一个序列拼起来模 p 几乎可以看作是一个随机函数，异或和为 0 跟拼起来模 p 等于 0 之间几乎没有相关性，于是当 p 只有区区 50000 这么大，我们就有非常非常大的概率找到解。当然了，对于 $p = 2$ 或 5 的情况，显然在 31 以内随便就能找到解的。所以这样做是靠谱的。

至于时间复杂度嘛，要是这个数据范围再扩大一些，我们应该设定一个 l，忽略 2^l 以上的数。异或和为 0 的大约会有 $\frac{2^{2^l}}{2^l}$ 个，所以我们可以选一个 $\Theta(\log\log p)$ 级别的 l。所以，时间复杂度应该为 $O(n + p\log^2 p)$，空间复杂度也是如此。

时间复杂度：$O(n + p\log^2 p)$

空间复杂度：$O(n + p\log^2 p)$

9 Codeforces 241E：Flights

【大意】

有一个 n 个结点且每条边权值均为 1 的有向图，要你把一些边权改成 2 使得任意一条从 1 到 n 的路径长度都相等。

【解法】

如果已经知道每个点到 n 的最短距离，每条边的权值就容易得出来。所以就是去掉 1 不能走到的和不能走到 n 的结点后每条边两端的距离差不超过 2。所以就得到了一个差分约束问题，最短路跑跑就行了。

时间复杂度：$O(nm)$

空间复杂度：$O(n + m)$

10 Codeforces 241F：Race

【大意】

有一个长方形的城市，有着 $n \times m$ 个方格状的街区。

城市由建筑物、双向直线的道路和交叉路口组成。每个建筑物占恰好的一个街区，所有道路的宽度都为一个街区，并且道路都是水平的或竖直的。每个交叉路口占一个街区，位于道路的交汇处。我们称两个街区是相邻的，当且仅当它们之间存在着公共边。没有两条道路或两个交叉路口是相邻的。

在每年的狂欢节中，Old Peykan 总会按照一条特殊的路线游行。Old Peykan 会从一个道路的中间出发，接下来经过若干个交叉路口，最终在一个道路的中间停下。Old Peykan 知道从某一个街区到其相邻的街区所花费的时间。同时，从交叉路口需要花费 1 分钟来到达相邻的街区。Old Peykan 不能经过有建筑物的地方。

我们知道 Old Peykan 的初始与结束位置，也知道 Old Peykan 途中经过的交叉路口的顺序 $s_1, s_2, \ldots, s_l$。但他完成游行之后，他会一直呆在结束位置。Old Peykan 总会沿着最短的道路经过给定的交叉路口到达终点。

你的问题是，找出在他出发了 k 分钟之后，位于哪个位置。

【解法】

真是无聊题啊，就是按照题意裸模拟就行了。

时间复杂度：$O(nm + k)$

空间复杂度：$O(nm + l)$

11 Codeforces 243C：Colorado Potato Beetle

【大意】

有一个人在方格上行走，每次给出方向和距离，求围住了多少个格子（包括边界）。$n \leq 1000$

【解法】

范围居然只有 1000……直接离散化之后纯模拟出行走路线，最后再 Flood Fill 就行了。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

12 Codeforces 243D：Cubes

【大意】

有一座积木城市建在一个 $n \times n$ 的正方形上的，第 x 行 y 列的单元格上有高度为 $a_{x,y}$ 的积木塔。问在无穷远处以向量 $v = (v_x, v_y, 0)$ 的方向观望可以看到多少个立方体（看到正方形上一个点即为看到）。$n \leq 1000$。

【解法】

一开始我想复杂了走上了各种奇形怪状的查询的不归路，但其实正解就是简单粗暴的一个类似画家算法的算法。我们可以由近及远地把每个积木塔画到画面上，每次画时候查询有多少被覆盖了，就可以知道有多少能被看见。

我们以俯视观察，对每个格点引方向为 v 的平行线，我们知道每两个相邻平行线间的积木之间的遮挡关系是很单纯的，只用记录当前最高的积木塔高度就行。所以我们可以按这个平行线来离散化，每次查询积木塔高度就是查询区间内被遮挡的高度的最小值，绘制积木塔就是用自己的高度更新区间内被覆盖的最大高度。显然我们可以用线段树维护这个过程。

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n^2)$

13 Codeforces 243E：Matrix

【大意】

给定一个 $n \times m$ 的 01 矩阵，要求重排矩阵的列的顺序，使得每一行的 1 是连续的。

【解法】

PQ 树裸上即可。PQ 树中 P 结点表示儿子顺序任意，Q 结点表示儿子只能顺序或逆序。知道了 PQ 树定义之后其实算法可以自己想出来了，就是各种 dfs 乱搞就好了。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

14 Codeforces 248E：Piglet's Birthday

【大意】

有 n 个柜子，第 i 个柜子里有 a_i 个装了蜜的蜜罐。有 q 个操作，每次给出 v, u, t，在第 v 个柜子里随机取 t 个蜜罐吃光后放到第 u 个柜子。每次操作后输出蜜罐全空的柜子数量。$a_{max} \leq 100, t \leq 5$。

【解法】

一个很自然的想法是记录第 i 个柜子有 j 个非空蜜罐的概率，但是虽然初始时不超过 100 个，可能从其它柜子运来一些蜜罐就会有很多了，这岂不是会很慢？

其实这是不必要的担心，因为从其它柜子运过来的蜜罐一定是空的。所以我们安心 dp 就好了。每次其实只用更新下 v 的 dp 数组，对于 u 只有罐子总数变了而已。

时间复杂度：$O(na_{max} + qa_{max}t_{max})$

空间复杂度：$O(na_{max})$

15 Codeforces 249D：Donkey and Stars

【大意】

给你 n 个点，求最长的从原点开始的一条折线使得折线上线段的倾斜角均介于 α 到 β 之间。

【解法】

其实这题就是个最长不下降子序列。把点放在分别倾斜 α 和 β 的坐标轴建立的坐标系里看，然后按 x 排序，就成了普通的最长不下降子序列了。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

16 Codeforces 249E：Endless Matrix

【大意】

有一个无穷的矩阵，大概是这样：

1	2	5	10	17	26	...
4	3	6	11	18	27	...
9	8	7	12	19	28	...
16	15	14	13	20	29	...
25	24	23	22	21	30	...
36	35	34	33	32	31	...
...	...	...	...	...	...	...

T 个询问，每次框一个矩形要你求和。如果和太大超过 10 个字符，前面用 ... 表示，后面只保留 10 位。

【解法】

首先使劲推公式。求尾数显然用取模，难点在于怎么判断和太大了。这个其实也不难，用 `double` 算一遍就行了。

时间复杂度：$O(T)$

空间复杂度：$O(1)$

17 Codeforces 251D：Two Sets

【大意】

有 n 个不超过 2^d 的整数，要分成两个集合使得两个集合内元素异或和加起来最大。

【解法】

求出所有元素的异或和 s，假设其中一个集合的异或和为 x 那么另一个就是 $s \oplus x$。如果 s 上某位为 1 那么这一位对答案的贡献肯定是 1，如果是 0 的话那么要么是 0 要么是 2，我们就希望有更多的 2。

所以高斯消元搞搞，每次从高到低考虑每一位，尝试加入方程强制它产生 2，如果变无解了就去掉该方程。

时间复杂度：$O(d^2 n)$

空间复杂度：$O(dn)$

18 Codeforces 251E：Tree and table

【大意】

给你一棵结点数为 $2n$ 的树，要放进 $2 \times n$ 的方格纸里，使得相邻结点在方格纸中相邻。问方案数。

【解法】

这是一道分类讨论题。

首先如果所有结点的度数小于等于 2，那么说明是一条链。我们考虑一条长度为 $2n$ 的链放进方格纸的方案数，我们可以枚举链的一端放在哪里，然后肯定是拐向某一侧填满，然后填满剩下的一侧，或者干脆没有“剩下的一侧”。由于要拐出去，所以填满前一侧只有 1 种方案，对于后一侧，转化为已知链的端点是左上角求一条长度为 $2n$ 的链放进方格纸的方案数。对于这个问题，肯定是先上下扭再打个来回，枚举上下扭结束的位置就可以知道方案数为 n，所以一条长度为 $2n$ 的链放进方格纸的方案数就可以推出是 $\frac{n(n-1)}{2} \times 4 + 4$。有个小细节是当 $n = 1$ 时有些方案就是一样的了，该式子不成立，方案数应该为 2。

现在我们考虑树中有度数大于 2 的。如果有度数大于 3 的显然无解。我们随便取出一个度数为 3 的结点作为根，枚举它在方格纸中的位置。我们翻转格子使得它在第一行，那么它的三个儿子分别要在左边下边和右边。我们暴力枚举儿子的位置，然后就转化成了这两类问题：

1. 已知以 v 为根的子树在左上角，求放进方格纸的方案数。
2. 已知以 v 为根的子树在左上角，以 u 为根的子树在左下角，求放进方格纸的方案数。

对于第二个问题，一定是上下都先直走，最后有一边走到了尽头，另一边露出头来转化为问题一。

于是我们考虑问题一，假如这个子树内没有度数为 3 的结点那么肯定是条链，问题已经解决。否则我们顺着 v 往下找找到第一个度数为 3 的结点 u，我们称 u 的儿子分别为 l 和 r。我们需要把 v 到 u 之前的链给折叠起来，然后放 u 的子树。我们先暴力枚举 l, r 的顺序，然后 u 的位置只有两类：

1. u 与父亲的右边，l 与 u 在同一列但不在同一行，r 在 u 的右边。l 的子树可能往左边填充空隙也可能往右。
2. u 与父亲在同一列但不在同一行，l 在 u 的左边，r 在 u 的右边。l 的子树会在左边填充空隙。

于是讨论每一种情况算一算就行了。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

19 Codeforces 253E：Printer

【大意】

有 n 个任务，已知每个任务有开始时间 t_i，持续时间 s_i，优先级 p_i。每次会执行优先级最大的任务。但是现在有一个任务 x 的优先级弄丢了，但是知道结束时间 T。保证问题有解，请求出一组可能的解。

【解法】

我们首先想象任务 x 的优先级为无穷小，也即无视掉任务 x，然后求出每个任务在 $[t_x, T]$ 这个时间段内的执行时间。这个显然可以用堆来处理。

接着我们想象逐渐增大 x 的优先级，这时我们注意到，这段时间内优先级低的任务被一个一个吞噬了。

所以实际上我们可以把任务按优先级从低到高排序（该时间段内的空闲时间当作一个优先级为 0 的任务），然后就是要找前若干个任务使得执行时间之和等于 s_x。我们可以知道 x 的优先级比它们都高就能吞噬掉他们，所以我们可以令 x 的优先级为这些任务中优先级最高的那个任务的优先级加 1，然后进行微调使得不与其它任务的优先级相同。

这样我们就得到了一组合法解。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

20 Codeforces 254D：Rats

【大意】

有一个 $n \times m$ 的地下室，每个单元格要么是空地要么是有老鼠的空地要么是障碍物。现在可以在地下室的空格子投 2 个手榴弹，距离不超过 d 的老鼠会被炸死（四连通）。问是否能炸死所有老鼠。$n, m \leq 1000, d \leq 8$。

【解法】

其实关键在于 d 如此之小。考虑最好情况一个手榴弹能炸死 $2d(d+1)+1$ 个老鼠，所以当老鼠的数量超过 $2 \times (2d(d+1)+1)$ 时肯定是无解的。剩下的老鼠就很少了，我们可以枚举第一只老鼠被哪个手榴弹炸死，然后再找到编号最小的没被这个手榴弹炸死的老鼠，枚举它被哪个手榴弹炸死，然后再判定。对每只老鼠用 bfs 进行预处理哪些手榴弹能炸到它，判断时只用 $O(\log d)$ 二分一下。

时间复杂度：$O(nm + d^6 \log d)$

空间复杂度：$O(nm)$

21 Codeforces 256D：Liars and Serge

【大意】

有 n 个人坐在一张长桌边上。对于每个人，我们知道他总是说真话或是说谎。

Little Serge 问他们：“你们中有几位总是说真话呢？”桌子上的每个人都知道关于桌子上的所有人的所有事情（某个人说真话或是说谎）。诚实的人总是回答正确的答案，而说谎的人会回答 1 到 n 之间除了正确答案以外的任意一个整数。

Little Serge 除了知道这些人对于他的问题所给的答案之外，并不知道其他的信息。他拿来一张纸并写下一个序列表示每个人的答案，接着他断定，桌上至少 k 个人明显说谎了。

求这样的序列的方案数。$n \leq 2^8$，保证 n 是 2 的整数次幂。

【解法】

无聊题啊！首先一个序列的最多可能说谎人数就是满足说答案为 v 的恰有 v 个人的 v 中最大的 $n-v$，所以 DP 搞搞就行了。

这时就凸显出题人的搞笑之处了，直接做会 TLE，所以我们打个表交上去就行了。

时间复杂度：$O(n^4)$

空间复杂度：$O(n^3)$

22 Codeforces 257E：Greedy Elevator

【大意】

有一幢 m 层的办公楼，其中有一个先进的电梯控制系统，工作方式如下：

在 $t=0$ 的时刻，电梯位于第 1 层。电梯是空的而且别的楼层上没有人在等待。接着，在时刻 t_i 会有人来到某楼层等待电梯。对于每一个人我们知道三个参数：他开始等电梯的时刻，他初始位于哪个楼层以及他想去哪个楼层。

电梯的移动方式如下：在一个时刻 t 电梯肯定停在某一个楼层。电梯中所有想去这个楼层的人会出电梯，然后所有在这个楼层等待的人会进入电梯。我们可以假设以上的过程都是瞬间完成的。然后，电梯需要决定向哪个方向开动。电梯决定的方式如下：

1. 如果电梯已经是空的而且整幢楼中都没有人在等电梯，那么电梯在 $t+1$ 时刻仍然会保持在那一楼层。
2. 否则，如果电梯在楼层 x，我们定义 p_u 表示电梯中要到编号比 x 大的楼层的人和当前时刻 t 在编号比 x 大的楼层等电梯的人的总数，p_d 表示电梯中要到编号比 x 小的楼层的人和当前时刻 t 在编号比 x 小的楼层等电梯的人的总数。如果 $p_u \geq p_d$，那么电梯在时刻 $t+1$ 会到楼层 $x+1$，否则电梯会到楼层 $x-1$。

你的任务是模拟电梯的工作并且告诉每一个人他能在哪个时刻到达他想去的楼层。

【解法】

依题意模拟就好了，用一棵平衡树存下当前等待的人和当前电梯里的人，每次算出下一个事件发生的时间然后跳到下一个事件发生的时刻进行处理。可以使用 STL 的 set，移动时算一下 $p_u - p_d$ 的增量。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

23 Codeforces 258D：Little Elephant and Broken Sorting

【大意】

有一个长度为 n 的排列 p，会进行恰好 m 次操作，第 i 次操作会交换第 a_i 个数和第 b_i 个数，每次操作是会有一半几率不操作，求逆序对的期望个数。$1 \leq n, m \leq 1000$

【解法】

维护数组 $f[i][j]$ 表示 $p_i > p_j$ 的概率，每次操作时更改下影响了的元素对的这个概率，最后把是逆序对的概率加起来就行了。

时间复杂度：$O(n^2 + nm)$

空间复杂度：$O(n^2 + m)$

24 Codeforces 260E：Dividing Kingdom

【大意】

平面上有个 n 个点，要你用两条平行于 x 轴的直线和两条平行于 y 轴的直线严格地划分成 3×3 块，每一块内点数已知，顺序未知。

【解法】

直接枚举 9! 种可能的排列然后就知道了直线的位置然后判定即可。考虑判定，其实只需判定 4 个形如 "x 坐标小于等于 x_0 且 y 坐标小于等于 y_0 的点有 s 个" 的条件。所以把坐标离散化，离线把插入点和询问点按 x 排序然后 y 方向用树状数组维护就好了。

复杂度貌似不太好写的样子。我们假设该问题是划分出 $d \times d$ 块好了。

时间复杂度：$O((n + d^2) \cdot d^2! \cdot \log n)$

空间复杂度：$O(n + d^2 \cdot d^2!)$

25 Codeforces 261D：Maxim and Increasing Subsequence

【大意】

给出一个长度为 n 的序列 b，把序列 b 拷贝 t 次拼起来成序列 a。求 a 的最长上升子序列。$n \cdot b_{max} \leq 10^7$。

【解法】

a 的最长上升子序列的长度不会超过 b_{max}，所以最多只用拷贝 b_{max} 遍。

最裸的 dp 方法就是 $f[i]$ 表示前 i 个元素的最长上升子序列长度。回忆最长不上升子序列的单调栈的那个做法，在 $s[k]$ 这个地方存下 $f[i] = k$ 的最小的 $a[i]$，结合二分可以做到 $O(n \cdot b_{max} \log b_{max})$。但是这样是过不了的。

注意到如果 $i < j, a[i] = a[j]$，那么 $f[j] \geq f[i]$。于是我们可以从上一个与我当前的 $a[j]$ 相等的 $a[i]$ 处的 $f[i]$ 开始暴力在栈中往上爬，爬到不能爬为止。由于 $f[j]$ 不会超过 n，所以对于每个取值来说，往上爬的次数是 $O(n)$ 的。总共不超过 b_{max} 种取值，所以总的往上爬的次数是 $O(n \cdot b_{max})$ 的。

时间复杂度：$O(n \cdot b_{max})$

空间复杂度：$O(n + b_{max})$

26 Codeforces 261E：Maxim and Calculator

【大意】

对于数对 (a,b)，每次可以 $b=b+1$ 也可以 $a=a\times b$。问不超过 n 次操作后给定区间 $[l,r]$ 内有多少种可能的 a。$n\le 100, r\le 10^9$。

【解法】

设 $f(n,r)$ 表示 $[1,r]$ 中仅包含大小不超过 n 的素数的整数的个数。对于题目给的范围，这个函数值不超过 3×10^6。所以我们暴力找出这些数然后再暴力 DP 一下求出最小步数，再暴力统计 $[l,r]$ 内可能的 a 就行了。

时间复杂度：$O(f(n,r)\times\pi(n))$

空间复杂度：$O(f(n,r))$

27 Codeforces 263E：Rhombus

【大意】

给你一个 $n\times m$ 的矩阵和一个正整数 l。求 $\sum_{i=1}^{n}\sum_{j=1}^{m}a_{i,j}\max\{l-|i-x|-|j-y|,0\}$ 在 $l\le x\le n-l+1, l\le y\le m-l+1$ 中的最大值。

【解法】

设 $f(x,y)$ 为题目中要最大化的函数，显然我们应该对每个格子把 $f(x,y)$ 求出来。然后考虑 $f(x,y)$ 的 x,y 方向的二阶偏差分，发现由大概 5 个和式组成，和式中每项的系数都为常数，其中 2 个和式对应到矩阵中是从左上到右下斜着的，2 个和式是从右下到左上斜着的，1 个和式是横着的或竖着的。

于是我们可以暴力求出 $f(l,l), f(l+1,l), f(l,l+1)$，然后其它的 $f(x,y)$ 通过二阶偏差分求出来就行了。当然我们要预处理出横着竖着斜着的前缀和。

时间复杂度：$O(nm)$

空间复杂度：$O(nm)$

28 Codeforces 264D：Colorful Stones

【大意】

有两个长度为 n 和 m 的 `RGB` 序列，两个人初始时分别站在序列开头，每次喊出 `RGB` 中的一个字符，站在那个字符上的人会向前一步。问有多少种可能的站立位置的状态。

【解法】

对于一个序列上区间对应的子序列 A，我们用 A^+ 表示 A 的右端点向后挪一个形成的区间，用 ${}^-A$ 表示左端点向左挪一个形成的区间。用 $A\subseteq B$ 表示 A 是 B 的子序列。

考虑这两个序列上的两个区间对应的子序列 A,B，那么判断是否能分别站在 A 和 B 的开头走到 A 和 B 的结尾等价于问是否存在一个序列 I 使得 $A\subseteq I, B\subseteq I, A^+\not\subseteq I, B^+\not\subseteq I$。

显然，$A \not\subseteq B^+$ 且 $B \not\subseteq A^+$，于是我们考虑满足这两个条件后是否一定存在 I。我们考察 A 和 B 的第一个字符，假设分别为 c_1, c_2。如果 $c_1 = c_2$，那么只能让他们同时向前走一步，A^- 和 B^- 显然仍然满足条件。假设 $c_1 \neq c_2$，那么：

1. 如果 $^-A \not\subseteq B^+$，那么我们可以喊出 c_1，然后 A 变为 ^-A，仍然满足条件。
2. 如果 $^-B \not\subseteq A^+$，那么我们可以喊出 c_2，然后 B 变为 ^-B，仍然满足条件。
3. 如果不满足以上条件，那么说明 $^-A \subseteq B^+$ 且 $^-B \subseteq A^+$。这隐含着 $|A|$ 和 $|B|$ 之差的绝对值不超过 1，从而说明 ^-A 是 B^+ 的后缀，^-B 是 A^+ 的后缀，于是 A 和 B 必为 $c_1c_2c_1c_2\ldots$ 的形式。

另一方面，A 和 B 的末尾两个字符如果是 c_1c_2 和 c_2c_1 那么肯定不可行。我们排除掉这种情况，那么就一定不存在情况 3 了。

所以我们只要统计出满足 $A \not\subseteq B^+$ 且 $B \not\subseteq A^+$ 的方案再去掉那个特殊情况就行了。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

29 Codeforces 266D：BerDonalds

【大意】

给定一个无向带权连通图，求图上一个在边上或结点上的点使得到最远的结点距离最短。

【解法】

Floyd 一遍求出任意两点间距离，如果在结点上显然裸枚举。如果在边上，我们设到边的左端点距离为 x，就可以画出到每个结点距离的函数图像，是一个山峰，而最终要关注的是这些图像叠加起来取 max。我们只要把山峰按左边的起始高度排序（即结点到边的左端点的距离）然后删除包含的山峰，就能够枚举山谷得到答案了。

时间复杂度：$O(n^3 \log n)$

空间复杂度：$O(n^2)$

30 Codeforces 266E：More Queries to Array...

【大意】

有一个长度为 n 的数组 a，有 q 个操作，每次可以区间赋值，或者给出 l, r, d 查询 $\sum_{i=l}^{r} a_i(i-l+1)^d$。$0 \leq d \leq 5$。

【解法】

用线段树维护 $d = 0 \ldots d_{max}$ 时每个元素乘以下标的 d 次方的和。显然通过对每个 m 和 d 预处理 $\sum_{k=1}^{m} k^d$ 能做到 $O(d_{max} \log n)$ 时间内进行区间覆盖。

考虑查询，直接 $\sum_{i=l}^{r} a_i(i-l+1)^d = \sum_{k=0}^{d} \sum_{i=l}^{r} a_i i^k \cdot \binom{d}{k}(-l+1)^{d-k}$ 就行了。这样询问也能做到 $O(d_{max} \log n)$。

时间复杂度：$O(nd_{max} + qd_{max} \log n)$

空间复杂度：$O(nd_{max})$

31 Codeforces 268D：Wall Bars

【大意】

对于一个长度为 n 的仅包含 ABCD 的字符串，如果我们可以从第 1 到 h 个中的某个字符出发，每次跳不超过 h 个字符落在相同的字母上，最后跳到 $n-h+1$ 到 n 间的某个字符，那么就称为好的。求好的字符串的个数。$n \leq 1000$，$h \leq 30$。

【解法】

直接 dp 即可。记录当前填了第几个字符以及最后一个 ABCD 的位置。由于 ABCD 中肯定有一个是当前填了的，所以我们可以把当前的看作 A，其它的看作 BCD，这样就省掉了一维。如果距离曾经超过 h 就记为 h，由于 A 也可能曾经超过了 h，所以加一维记录是否超过。

时间复杂度：$O(nh^3)$

空间复杂度：$O(nh^3)$

32 Codeforces 269D：Maximum Waterfall

【大意】

一个人造瀑布包括许多个嵌在墙壁中的水平板，水从一块水平板流向另一块，直到它从墙的顶端流到底端为止。

墙的高度是 t，有 n 块水平板嵌在上面。每块水平板可以看作一条水平的线段，高度为 h_i，从 l_i 到 r_i。墙的顶端可以看作一块在 $(-10^9, t)$ 到 $(10^9, t)$ 间的水平板。相似的，墙的底端可以看作一块在 $(-10^9, 0)$ 到 $(10^9, 0)$ 间的水平板。不会有两块水平板之间有公共点。

水可以从第 i 块板流向第 j 块板当且仅当：

1. $\max(l_i, l_j) < \min(r_i, r_j)$
2. $h_j < h_i$
3. 不存在 k 满足 $h_j < h_k < h_i$，且 (i, k) 与 (k, j) 均满足以上两个条件。

从 i 到 j 的流量为 $\min(r_i, r_j) - \max(l_i, l_j)$，相当于二者的水平相交部分的长度。

水会沿着一条单向的路径从顶部流到底部。如果水流到了一块水平板上（除了墙的底部），水会流向恰好一个更低的水平板。整个瀑布的水流量被定义为水流路径上每连续的两块水平板间的水平相交部分长度的最小值。

求最大水流。

【解法】

扫描线直接上，用一个平衡树维护可见的水平板。插入一个水平板时，检查它覆盖的水平板的被覆盖情况只需要检查与其相邻的，然后求出从这个水平板出发到底部的最大流量，然后把它覆盖的部分替换成它自己就行了。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

33 Codeforces 269E：String Theory

【大意】

有一架长方形的竖琴，由 n 行 m 列构成的 $n \times m$ 的网格。每一行从上到下一次依次标号为 1 到 n，每一列从左到右依次标号为 1 到 m。在竖琴的四周边界上，用来扣住琴弦的钉子均匀地分布在每一格内。同时，这架竖琴恰好有 $n+m$ 根不同的琴弦，每一根琴弦的两端分别被钉子固定在竖琴不同侧的边界上，而且每个钉子上有且仅有一根琴弦。

现在有一架竖琴。你可以进行以下两种操作：

- 选择不同的两列，对应地交换处于同侧的钉子（两侧必须同时交换），但不改变钉子与其所固定的琴弦的连接；
- 选择不同的两行，对应地交换处于同侧的钉子（两侧必须同时交换），但不改变钉子与其所固定的琴弦的连接；

要使得最后竖琴上没有交叉的琴弦。

【解法】

感觉这题还蛮好玩的。首先我们要考虑什么情况下不会有交叉的琴弦。琴弦大概有 6 种，左上，左下，右上，右下，还有横向和纵向。如果横向和纵向的琴弦同时存在，那么肯定无论怎么调整都有交叉出现。假设只有横向，那么我们可以转一下竖琴使得它变成纵向的。所以我们不用考虑存在横向琴弦的情况。

由于每个钉子上都有琴弦，所以左上的琴弦肯定是紧挨着的，左下、右上、右下也是如此。纵向的琴弦会填补剩下的空隙，而纵向琴弦肯定也是紧挨着的。

再进一步观察发现左上和右下的琴弦条数必定相等，左下和右上的琴弦条数必定相等。不妨设个数分别为 n_a 和 n_b。由此推知纵向琴弦的上端点的列编号与下端点的列编号之差必定为 $n_b - n_a$。

接下来对于 “不相交” 好像没什么新发现了。于是我们来换一个角度，看看 “交换行列” 的含义是什么。诚然，如果上下左右是能独立交换的，这题是没什么难度的。关键在于行是绑一起交换的，列是绑一起交换的。所以我们可以把同在一行的钉子间连一条边，同在一列的钉子间连一条边，这些边在交换行列时不会被破坏，而我们一旦知道一条弦的一端，由于要求不相交，所以它的位置也确定了，所以另一端也确定了，所以与这个钉子相连的钉子也确定了。这样，我们一旦确定了一个钉子，也就确定了一个连通块。

观察这些连通块，必定是环。我们不妨假设 $n_a < n_b$。有 n_a 个环位于竖琴角落且只有 4 条弦。我们可以把这些环找出来然后安全地放到竖琴角落里就可以了。剩下的环就比较麻烦，需要进一步观察特性。竖琴上第 $n_a + 1$ 个钉子到第 $m - n_a$ 个钉子所在的环必定是在这种环上。考虑纵向的琴弦，总是上端点在 x，下端点在 $x + n_b - n_a$。注意到最后一根纵向琴弦的右边那根从上边界出发的琴弦，连向了右边界，好像破坏了这个性质。但是紧接着从右边界连向了左边界，左边界再连向了下边界。这个可以视为神奇的穿墙效应，最终连向的钉子为 $x + n - m$。于是可以联想到取模，如果把第 $n_a + 1$ 个钉子编号为 0 的话，连边就是每次 x 连向 $(x + n_b - n_a) \bmod (m - 2n_a)$。并且每次穿墙都会导致多带一个钉子。其实我们知道这样的环不会超过 $n_b - n_a$ 个，我们取起始点 0 到 $n_b - n_a - 1$，然后每隔 $n_b - n_a$ 个钉子往后跳。由于起始点的间距不超过跳的长度，所以穿墙的时间是一致的。这也就意味着这些环都是等长的且还是同构的。

所以我们可以预处理某个起始点的跳跃时的穿墙与否的序列，然后每次找到一个环时，先判断是不是角落里的，如果是就放角落。否则做一次 KMP 判循环同构，然后找到起始点，把这个环放到下一个空闲起始点所在环上。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

34 Codeforces 273D：Dima and Figure

【大意】

一块大小为 $n \times m$ 的长方形纸片，初始所有格子都是白色的。求把纸片上一些格子涂黑后好看的方案数。

好看当且仅当包含至少一个涂黑的格子，且所有的涂黑的格子形成一个连通块，从一个涂黑的格子到另一个涂黑的格子所需的最少移动步数等于曼哈顿距离。

【解法】

很久以前做过这题，当年做到这题时觉得刷新了我对 DP 的世界观。首先要好好得出什么叫好看，其实最准确的意思是：上下边界都是凸的。关键在于思考给你一个图形你如何检验这是答案，我们发现我们只用一条线横着扫过去，记录上端点和下端点，以及现在是在上升还是下降就能判定了，所以自然我们也可以一条线扫过去进行 DP，然后就行了。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

35 Codeforces 273E：Dima and Game

【大意】

有一个长度为 n 的整数对序列，第 i 对为 (l_i, r_i) $(1 \le l_i < r_i \le m)$。然后玩家轮流进行操作。轮到自己时，可以选择一对数 (l_i, r_i)，满足 $r_i - l_i > 2$，然后将第 i 对数替换为 $(l_i + \lfloor \frac{r_i - l_i}{3} \rfloor, l_i + 2\lfloor \frac{r_i - l_i}{3} \rfloor)$ 或者 $(l_i, r_i - \lfloor \frac{r_i - l_i}{3} \rfloor)$。不能进行操作的玩家则输。给出 n, m，问有多少个整数对序列使得先手必胜。

【解法】

显然每个数对的 SG 函数值只跟长度有关，而且由于后继状态数只有 2 个，所以 SG 值不超过 2。这个 SG 函数是个分段函数，我们可以递推求出每一段。由于这个函数的特殊性质，我们可以用归纳法证明段数是 $O(\log m)$ 的。然后我们就能求出 SG 值分别为 $0, 1, 2$ 的区间个数，最后 $O(n)$ DP 一下即可。

时间复杂度：$O(\log m + n)$

空间复杂度：$O(\log m + n)$

36 Codeforces 274C：The Last Hole!

【大意】

白纸上有个 n 个点，第 t 秒每个点都有一个以该点为圆心以 t 为半径的黑色的圆。问被黑色包围的白洞的最后消失时间。

【解法】

看起来完全无从下手，但其实只需要切换视角。我们考虑平面上一个点 p 成为黑色的时刻，如果那个时刻来临之前一点点它四周都是黑色的，说明这个位置是一个白洞。由于所有圆半径都相等，所以一个点 p 是白洞且在 t 时刻消失的条件是初始时半径为 t 的严格范围内没有黑点，边界上有若干个黑点且这些黑点在以 p 为圆心以 t 为半径的圆上相邻两点之间的弧都是劣弧。那么只有一两个黑点显然不可能，必须至少三个。于是我们 $O(n^3)$ 枚举所有三角形的外心，剩下的就是简单的判定。

当然由于我们枚举了每三个点，所以判定可以利用对称性只判一部分。首先判掉圆内有其它黑点的情况，然后如果这三个点组成了锐角三角形返回可行。现在我们只用考虑圆上不存在任何一个锐交三角形却仍是个洞的情况，那么只有可能圆上只有四个点且形成一个矩形。于是接下来我们尝试枚举第四个黑点看看是否能组成矩形就好了。

时间复杂度：$O(n^4)$

空间复杂度：$O(n)$

37 Codeforces 274E：Mirror Room

【大意】

有一个 $n \times m$ 的网格，有 p 个堵塞的格子，其他的格子是空的。在空格子 (x_s, y_s) 的中心向一个对角线方向发射一束激光。如果光束碰到堵塞的格子或网格边界，它会反射。过了一会儿，光束进入了一个无限的循环。计算至少被光束通过一次的空格子数。

【解法】

首先肯定是模拟一个周期，然后求面积。我还以为要写矩形面积并呢，结果发现不用。因为每次反射是由主对角线切换到次对角线方向并改变格子奇偶性，所以要想有交集运动方向不可能垂直。所以就直接把面积加起来就行了。特别的地方是，在往返跑的时候答案要除以二。

时间复杂度：$O((n+m+p)\log(n+m))$

空间复杂度：$O(n+m+p)$

38 Codeforces 277D：Google Code Jam

【大意】

考试时间为 T，有 n 道题，每道题的简单部分 a_i 分，需要 s_i 的时间，困难部分 b_i 分，需要在做完简单部分的前提下多花 t_i 的时间，但有 p_i 的概率写挂。问最大期望得分以及最大期望得分情况下的最小期望罚时。罚时定义为最后一次成功提交的时间。

【解法】

首先我们无视“先做简单才能做困难”的限制，把简单部分的 p_i 当作 0。

假设我们知道以什么顺序解决哪些任务，那么我们可以得到总分的期望为 $\sum_i a_i(1-p_i)$，而期望罚时可以用递推解决：$pl_i = p_i \times pl_{i-1} + (1-p_i) \times \sum_j t_j$。

如果知道要解决哪些任务但不知道顺序，我们可以很简单地用调整法证明顺序一定是按 $\frac{p_i}{1-p_i}t_i$ 排序才能使得期望罚时最小。我们发现这样的排序自动把简单部分放到了前面。

那么假设现在我们什么都不知道，我们就可以把题目先排个序，然后用一个简单的背包式的 dp 来解决。$f[i][j]$ 表示前 i 个题目花了 j 的时间的情况下的最优期望得分与罚时，然后 dp 方程就很好列出了。

时间复杂度：$O(nT)$

空间复杂度：$O(n+T)$

39 Codeforces 283E：Cow Tennis Tournament

【大意】

有 n 只奶牛参与比赛，每只奶牛能力值为 s_i，任两只奶牛的能力值互不相同。任意两只奶牛在锦标赛中只交手一次，等级高的总是击败等级低的。Farmer John 准备改变一些比赛结果。他会改变 q 次，每次选择两个整数 l, r 并翻转能力值在 l 到 r 范围内的奶牛相互之间的比赛结果。求多少组奶牛 $\{p, q, r\}$ 满足 p 打败 q，q 打败 r，r 打败 p。

【解法】

首先应该注意到其实是给一个竞赛图求三元环个数。进一步观察发现任意一个竞赛图都有可能通过一些操作构造出来。所以我们先考虑一个普通的竞赛图怎么求三元环。

考虑补集转化。我们先任取 3 个结点，这样一共 $\binom{n}{3}$ 种，然后我们去掉不是三元环的情况。如果不是三元环那么一定是拓扑图，那么一定有恰好一个结点的度数为 2。我们枚举这个结点 v，剩下的两个点之间的连边其实是任意的，那么假设结点 v 的度数为 d_v，方案数就是 $\binom{d_v}{2}$，所以三元环就有 $\binom{n}{3} - \sum_v \binom{d_v}{2}$ 个。

所以问题变成求每个结点的度数。于是把所有牛按能力排序。然后离线把操作按左端点排序扫一遍利用线段树维护翻转操作求出从前往后的出边，再把操作按右端点排序扫一遍利用线段树维护翻转操作求出从后往前的出边，加起来就能得到度数了。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

40 Codeforces 285E：Positions in Permutations

【大意】

求 $|p_i - i| = 1$ 恰有 g 个的排列 p 的数目。$n \leq 1000$。

【解法】

DP 搞搞。想象成二分图匹配，每次我们加入左边的第 i 号点和右边的第 i 号点并让他们两个配对，然后去找一个之前的配对关系进行交换。所以要考虑的因素只有：这个配对是否满足条件，这个配对里有没有出现 $i-1$。所以额外加两维记录左边第 i 号点和右边第 i 号点跟谁配对就好了。当然我们只用记录 i 跟 i、i 跟 $i-1$、其它这三种情况。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

41 Codeforces 286D：Tourists

【大意】

直角坐标系下，在一些时刻会有两个游客分别同时从点 $(-1,0)$ 和 $(1,0)$ 出发。每一对游客每秒都向 y 轴正方向前进一个单位长度。在一些时刻墙会出现，墙 (l_i,r_i) 是一条在点 $(0,l_i)$ 和 $(0,r_i)$ 之间的线段。给出 m 堵墙的出现时间 t_i 以及出现的区间，给出 n 对游客出发时间 q_i。请求出对于每一对游客有多长时间无法彼此望见。

【解法】

以原坐标系的 y 为横轴，时间为纵轴，我们就可以发现墙变成了若干个上无尽头的矩形，而一对游客是一个从纵轴出发的射线。

我们用纵向的扫描线去扫这些矩形，求出这些矩形的并集。然后找出顶点作为事件点，用倾斜的扫描线去扫，碰到顶点就加加减减，碰到询问的射线就把当前长度作为答案。

时间复杂度：$O((n+m)\log(n+m))$

空间复杂度：$O(n+m)$

42 Codeforces 288E：Polo the Penguin and Lucky Numbers

【大意】

定义幸运数字是正整数，同时它们的十进制数表示只能包含幸运数字 4 和 7。

有两个正整数 l 和 r，他们都是幸运数字。此外，他们的长度是相等的。

假设在 l 与 r 之间有 n 个不同的幸运数字，a_k 表示序列里的第 k 个（以升序排列）。请计算出 $\sum_{k=1}^{n-1} a_k a_{k+1}$。

【解法】

裸数位统计。只要搞搞题目中要求的那个和、序列中最后一个元素值、序列长度、序列总和就可以很方便地支持 “每个数前面加一个 4”、“每个数前面加一个 7” 这两种操作了。

时间复杂度：$O(\log r)$

空间复杂度：$O(\log r)$

43 Codeforces 293B：Distinct Paths

【大意】

有一个 $n \times m$ 的木板，一些块已经被涂上给出的 k 种颜色中的一种。你需要把每个没涂色的块涂色使得从左上角到右下角的每条路径都不会经过两个颜色一样的块。路径只能向右或向下走。$k \leq 10$

【解法】

显然右上的那条长度为 $n+m-1$ 的路径上颜色必须互不相同，所以 $n+m-1 \leq k$，于是其实数据范围非常小。

我先尝试暴力状压 DP 去了，结果 TLE 得非常爽，最后怎么也卡不进时限。去看了看题解得到一个复杂度不明的算法……囧。

我们暴力搜索每个格子的颜色，与之前不同的是我们对颜色进行最小表示法，也就是说更早搜到的颜色编号更早，然后尝试对颜色重标号使得满足题目条件，这一步可以直接乘法原理搞搞。但是爆搜的搜索量居然是惊人的小，吓傻了（好吧仔细想想我之前的那个状压也挺复杂度不明的）。

我写一个很松很松的上界作为时间复杂度好了。

时间复杂度：$O(k^{k^2})$

空间复杂度：$O(k)$

44 Codeforces 293D：Ksusha and Square

【大意】

给一个 n 个整数顶点的凸多边形，问在内部选两个不同的整点以此为对角线的正方形的大小的期望。绝对值坐标范围不超过 $L = 10^6$。

【解法】

面积显然是欧几里德距离的平方除以 2。拆成 x 的差的平方加 y 的差的平方。

每个结点向下搞个垂线形成一个梯形剖分，然后就可以对于每条边算贡献求出每个 x 坐标上有多少个在凸多边形内的整点，然后就搞搞算期望就行了。

时间复杂度：$O(n+L)$

空间复杂度：$O(n+L)$

45 Codeforces 293E：Close Vertices

【大意】

给你一棵 n 个结点的带权树，两个结点是相邻的当且仅当两点间的路径上的边数不超过 l_m, 权值和不超过 w_m。求相邻无序点对数。

【解法】

一道点分治的大水题。每次找重心，然后求出过重心的所有路径中的满足条件的，然后把重心删掉。这个显然可以求出每个点到重心的边数和边权和。如果不强制经过重心显然只要按边权和排序然后扫一扫，每次求出边数 l 小于等于 $l_m - l_i$ 的点的个数就好，这个用树状数组维护。现在强制经过重心，显然可以对整棵树求一下答案然后分别减去每棵子树的答案就好了。

时间复杂度：$O(n \log^2 n)$

空间复杂度：$O(n)$

46 Codeforces 294D：Shaass and Painter Robot

【大意】

一个 $n \times m$ 的白棋盘，机器人初始位于棋盘边界 (x, y)，它会朝某个方向（左上、左下、右上、右下）一直走直到撞墙然后按光的反射定律改变方向，经过的格子会被染黑。问多少步后棋盘变为黑白相间的。$n, m \leq 10^5$。

【解法】

这题的算法其实很简洁。棋盘变为黑白相间前机器人走到的最后一个格子一定是边界上的格子，因为如果不是的话，这隐含着他马上要撞墙的那个地方是他的起点，而与行进方向垂直引直线碰到的那两个边界上的格子没有互相走到过。而我们只有一个起点，所以只可能最后一个格子是边界上的格子。那么我们只用暴力模拟直到每个边界上的应该走到的格子都是黑的就行了。由于我用了 map 来标记，所以时间复杂度里多了个 $O(\log(n+m))$

时间复杂度：$O(n + m \log(n+m))$

空间复杂度：$O(n+m)$

47 Codeforces 295D：Greg and Caves

【大意】

求 $n \times m$ 的黑白屏幕上呈现洞的方案数。洞的定义是存在 l, r 使得区间 $[l, r]$ 里的行中有且仅有两个黑点，其它位置都是白的，且存在 t 使得 $[l, t)$ 的行中这一行被黑点夹住的区域是下一行的子集，$(t, r]$ 的行中这一行被黑点夹住的区域是上一行的子集。$n, m \leq 2000$。

【解法】

考虑洞的上半部分，DP 求出 $f[i][j]$ 表示高度为 i 底为 j 的“半个洞”的方案数，用前缀和优化下就能 $O(nm)$ 求出。然后枚举洞中最大的满足条件的 t，把两边方案数乘起来即可。

时间复杂度：$O(nm)$

空间复杂度：$O(nm)$

48 Codeforces 297E：Mystic Carvings

【大意】

一个圆上有 n 条弦连接 $2n$ 个端点，要选出三条弦。定义两点间的距离为最少经过多少个选出的弦的端点。问有多少种选出的三条弦的两端距离都想等。

【解法】

对于选出三条弦，只有五种可能的位置关系：三、XD、囧、H、三线交叉。题目要我们统计的是囧和三线交叉的数目之和，我们进行补集转化转而统计三、XD 和 H 的数目之和。对于三，显然可以枚举中间的弦，单独统计 XD 或 H 是很困难的，但是我们可以枚举 XD 的斜线然后剩下的一条在外面，一条跟它交叉，那么就能算出 H 和 XD 的数目之和。

按顺时针展开这个圆，然后用树状数组求出每条弦包含了多少条弦以及被多少条弦包含，然后就好统计了。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

49 Codeforces 301C：Yaroslav and Algorithm

【大意】

Yaroslav 喜欢算法，他最喜欢的算法是：

1. 这个算法的输入为一个字符串 a。
2. 这个算法由一些命令组成。i 号命令的形式为 "s_i>>w_i" 或 "s_i<>w_i"，其中 s_i 和 w_i 是长度不超过 7 的由数字或字符 "?" 组成的字符串。
3. 这个算法每次寻找一个编号最小的命令 i，使得 s_i 是 a 的子串。如果没有找到则终止。在字符串 a 中，s_i 第一次出现的位置会被 w_i 替换。如果这个命令形如 "s_i>>w_i"，那么继续执行，否则终止。
4. 算法会输出最终的 a。

Yaroslav 有一个 n 个正整数的集合，请给出一个这样的算法能够使对于集合内的每个元素 x，输入 x 时会输出 $x+1$。

$n \leq 100$, 集合内元素不超过 10^{25}，输出的命令条数不能超过 50，以集合内每个数为输入时执行的命令条数不能超过 200。

【解法】

给出的集合显然是没用的，不如直接搞个加 1 的通用算法。

考虑加 1 的过程，首先我们要找到字符串结尾，然后尝试给末尾加 1。如果末尾是 9 则变为 0 然后处理倒数第二位，以此类推。

所以我们首先把字符串开头放个 "?"，然后逐渐挪到最后一位去。然后再放个 "?"，逐渐挪到最后面去。于是我们就可以用 "??"，来定位。然后每次考虑 "??" 前面的那个数字尝试加 1 就好了。

code

```
9??>>??0
0??<>1
1??<>2
2??<>3
3??<>4
4??<>5
5??<>6
6??<>7
7??<>8
8??<>9
??<>1
?0>>0?
?1>>1?
?2>>2?
?3>>3?
?4>>4?
?5>>5?
?6>>6?
?7>>7?
?8>>8?
?9>>9?
0>>0?
1>>1?
2>>2?
3>>3?
4>>4?
5>>5?
6>>6?
7>>7?
8>>8?
9>>9?
```

时间复杂度：$O(1)$

空间复杂度：$O(1)$

50 Codeforces 301E：Yaroslav and Arrangements

【大意】

如果一个数列相邻两项之差的绝对值均为 1（特别的，我们认为首项和末项也相邻）那么称为良好的，如果一个数列不降且长度大于等于 1 小于等于 n，每个数的值大于等于 1 小于等于 m，且重排后能得到至少 1 个至多 l_m 个良好的数列就称为优秀的。给出 n, m, l_m 求优秀的数列的个数。$n, m, l_m \leq 100$。

【解法】

假设有一个序列不降，我们来求它能产生多少个良好的数列。我们可以把良好数列中的元素从小到大依次加入，每次当前最大的元素可能不满足相邻两项之差不超过 1，但是其它元素

是肯定满足的。我们可以利用这一点记录添加了 v 之后有多少对相邻的最大元素 g_v。那么假设有 c_{v+1} 个值为 v 的，添加到这个数列中就有 $\binom{c_{v+1}-1}{g_v-1}$ 种放法，而无论怎么放 g_{v+1} 都等于 $c_{v+1}-g_v$。所以我们可以用状态 $f[v][n][g_{v-1}][l]$ 进行 dp，其中 n 是当前数列的长度，l 是方案数。

看起来这个算法是 $O(n^5)$ 的可能会超时，但是其实实际运算量不是很大，因为毕竟 l 是表示方案数的，很容易超过限制 l_m，所以真正方案数不为 0 的状态不是很多。所以这样判掉方案数为 0 的状态就能解决此题。

至于空间嘛，可以滚动数组，所以只有 $O(n^3)$。

时间复杂度：$O(n^5)$

空间复杂度：$O(n^3)$

51 Codeforces 303D：Rotatable Number

【大意】

一个 b 进制下长度为 n 的数 x 被称为可旋转数当且仅当所有 x 通过旋转得到的数是 $x, 2x, \ldots, nx$。

给出 n 和 b_m，找出最大的 b 满足 $1 < b < b_m$ 且 b 进制下存在长度为 n 的可旋转数。

【解法】

我们考察一个 b 进制下长度为 n 的可旋转数 x。$n = 1$ 的情况比较显然，下面只考虑 $n \geq 2$。考虑整数 bx，它相当于把 x 的末尾补了个 0。现在我们把 bx 对 $b^n - 1$ 取模，那么 bx 的第 n 位（即 x 的第 $n-1$ 位）会被放置到第 0 位的位置（数位从 0 开始标号，个位是第 0 位）。所以乘以 b 再对 $b^n - 1$ 取模可以把数旋转一位。于是 x 是可旋转数即：

$$\{kx | 1 \leq k \leq n\} = \{b^k x \bmod (b^n - 1) | 0 \leq k < n\}$$

我们考虑模 $b^n - 1$ 意义下的这个式子：

$$\{kx | 1 \leq k \leq n\} \equiv \{b^k x | 0 \leq k < n\} \pmod{b^n - 1}$$

显然这个还是成立的。设 $m = (b^n - 1)/\gcd(x, b^n - 1)$。我们同时除以 $\gcd(x, b^n - 1)$：

$$\{kx/\gcd(x, b^n - 1) | 1 \leq k \leq n\} \equiv \{b^k x/\gcd(x, b^n - 1) | 0 \leq k < n\} \pmod{m}$$

现在 $x/\gcd(x, b^n - 1)$ 就有逆元了，于是除掉：

$$\{k | 1 \leq k \leq n\} \equiv \{b^k | 0 \leq k < n\} \pmod{m}$$

我们知道 $\gcd(b, b^n - 1) = 1$，那么当然 $\gcd(b, m) = 1$。所以式子右边的集合中没有 m 的倍数，那么式子左边的集合也不应该有。这就证明了 $m \geq n + 1$。

注意式子右边的那个集合带上模 m 意义下的乘法能构成一个群，所以左边的那个集合也应该满足这个性质。也就是说 1 到 n 都与 m 互质，并且他们之间的乘法的结果都还会在 1 到 n 中。如果 $m \geq n + 2$，假设 $n + 1$ 是合数，那么我一定可以找来两个小于等于 n 的数乘起来得到 $n + 1$，这与群的封闭性矛盾。假设 $n + 1$ 是素数，那么 $n + 1$ 肯定是奇素数，而 $n + 2$ 肯定是偶数。如果 $m = n + 2$ 那么说明 m 与 2 不互质，矛盾。如果 $m > n + 2$，那么我一定可以找两个小于等于 n 的数（其中一个是 2）乘起来得到 $n + 2$，与群的封闭性矛盾。所以 $m < n + 2$。

综合上界和下界我们可以知道 $m = n+1$。这意味着 $(b^n-1)/\gcd(x, b^n-1) = n+1$，于是：

$$\begin{aligned} b^n - 1 &= (n+1)\gcd(x, b^n-1) \\ 0 \cdot \frac{x}{\gcd(x, b^n-1)} &\equiv (n+1)x \pmod{b^n-1} \\ 0 &\equiv (n+1)x \pmod{b^n-1} \end{aligned}$$

我们知道 $x \le b^n - 1, nx \le b^n - 1$ 所以 $(n+1)x \le 2b^n - 2$，所以只有下面三种情况：

$$\begin{aligned} (n+1)x &= 0 \\ (n+1)x &= b^n - 1 \\ (n+1)x &= 2b^n - 2 \end{aligned}$$

即：

$$\begin{aligned} x &= 0 \\ x &= \frac{b^n-1}{n+1} \\ x &= \frac{2b^n-2}{n+1} \end{aligned}$$

第一种情况可以排除，因为 $x > 0$。第三种情况，我们考虑 $nx = \frac{(2b^n-2)n}{n+1}$。由于 $n \ge 2$，所以 $\frac{n}{n+1} \ge \frac{2}{3}$，所以 $nx \ge \frac{4}{3}(b^n-1)$。由于 $b^n \ge 4$，所以 $nx \ge b^n$。这个数超过了 n 位，也不成立。

所以唯一的可能是 $x = \frac{b^n-1}{n+1}$。

下面我们证明：x 是可旋转数的充要条件是 $n+1$ 是素数且 b 是 $n+1$ 的原根。

充分性 充分性比较显然。如果 $n+1$ 是素数，而且知道 b 与 $n+1$ 互质（原根必然与模数互质），那么 $b^n \equiv 1 \pmod{n+1}$，所以 x 肯定是整数。考虑 $b^k x \bmod (b^n-1)$：

$$\begin{aligned} b^k x \bmod (b^n-1) &= \frac{b^k(b^n-1)}{n+1} \bmod (b^n-1) \\ &= \frac{b^k(b^n-1) \bmod ((n+1)(b^n-1))}{n+1} \\ &= \frac{(b^n-1)(b^k \bmod (n+1))}{n+1} \\ &= (b^k \bmod (n+1))x \end{aligned}$$

所以 $b^k x \bmod (b^n-1) \in \{jx | 1 \le j \le n\}$。由于 b 是原根，所以对于 j 我们必定能找到一个 k 使得 $b^k \bmod (n+1) = j$，所以 $jx \in \{b^k x \bmod ((n+1)x) | 0 \le k < n\} = \{b^k x \bmod (b^n-1) | 0 \le k < n\}$。所以 x 是可循环数。

必要性 必要性也比较显然。由于 x 是可循环数所以

$$\begin{aligned} \{jx | 1 \le j \le n\} &= \{b^k x \bmod (b^n-1) | 0 \le k < n\} \\ \{j | 1 \le j \le n\} &= \{b^k \bmod (n+1) | 0 \le k < n\} \end{aligned}$$

这说明 b 的所有次幂遍历了 1 到 n。所以 $n+1$ 是素数，b 是 $n+1$ 的原根。 □

所以这题就好做了。一个素数 p 的原根有 $\varphi(\varphi(p))$ 个，大致是均匀分布，所以原根其实是很拥挤的，我们只要暴力 $O(\sqrt{n})$ 判一下 $n+1$ 是否是素数，然后暴力从 b_m 往 1 枚举判定是否是原根。判定一个与 p 互质的 b（不互质当然不是原根）是否是 p 的原根的方法是枚举每个素

因子 q 判断 $b^{(p-1)/q} \neq 1 \pmod{p}$ 是否都成立。素因子个数是 $O(\log p)$ 的，预处理素因子然后每次快速幂求一下就可以做到每次 $O(\log^2 p)$。

时间复杂度：$O(\sqrt{n} + \log^2 n(b_m - answer))$

空间复杂度：$O(\log n)$

52 Codeforces 303E：Random Ranking

【大意】

有个 n 个人，每个人的分数在 $[l_i, r_i]$（可以是实数），对于每个人输出排名为 1 到 $n-1$ 的概率。$n \leq 80$。

【解法】

这题实在是逗得不行，感觉做这题纯属浪费时间。不考虑精度误差显然可以用多项式搞搞，$f_k(x)$ 表示小于等于 x 恰好 k 人的概率，显然可以递推，于是显然是个分段函数，每段是个多项式，对于每个人求出每段多项式然后积分搞搞就行了。这样是 $O(n^5)$ 的，不过我们可以使用多项式除法做到 $O(n^4)$ 但是有点精度问题。可笑的是我最后跟标程的答案的差距大概是 1.5×10^{-6}，稍微比允许的误差大那么一点点。

所以就只好用标程的做法了！对于每个人，枚举其它人，算出在他前面和在他的区间内的概率，背包一下，最后这两段分别搞搞就行了。这样是 $O(n^5)$ 的。加个什么分治就 $O(n^4 \log n)$ 了。但是好像继续上多项式除法还是可以做到 $O(n^4)$？算了不管了。总之我做这题时被精度玩傻了。

时间复杂度：$O(n^5)$

空间复杂度：$O(n^2)$

53 Codeforces 305D：Olya and Graph

【大意】

有一张 n 个点、m 条边的有向无权图。保证原图中任意从 u 到 v 的有向边满足 $u < v$。求有多少种方案添加若干有向边，使得该图满足下列条件：

1. 从点 i 出发，可以到达点 $i+1, i+2, \ldots, n$。
2. 任意从 u 到 v 的有向边满足 $u < v$。
3. 两点之间最多有一条边。
4. 对于一对点 i, j（$i < j$），若 $j - i \leq l$，那么从 i 到 j 的最短距离等于 $j - i$。
5. 对于一对点 i, j（$i < j$），若 $j - i > l$，那么从 i 到 j 的最短距离等于 $j - i$ 或 $j - i - l$。

【解法】

首先最终的图 i 到 $i+1$ 肯定有边。剩下的边一定对应着一个区间。如果区间长度不等于 $l+1$ 那么肯定不满足条件，如果有两个区间不相交那么肯定不满足条件。

所以其实就是问区间两两相交的方案数，枚举左端点最小的区间然后搞搞就行了。

时间复杂度：$O(n+m)$

空间复杂度：$O(n)$

54 Codeforces 305E：Playing with String

【大意】

两个人玩游戏。初始时有一个长度为 n 的字符串，每次可以选择一个字符串找出一个长度大于 1 的奇回文串沿把该字符串拆成三半：对称轴左边，对称轴，对称轴右边。不能走的人输。问谁必胜。

【解法】

首先回文串是唬人的，其实就是说 $s[i-1]=s[i+1]$ 时可以从 i 处拆开。

其次把字符串拆成三半是唬人的，拆出的三半中，对称轴这一半只有一个格子，肯定是没用的。

然后把字符串拆开其实也是唬人的。我们可以把问题转化成这样：有一个黑白纸条，黑格子表示字符串可以从这个拆开。每次可以选一个黑格子，把自己、左边、右边这三个格子都染成白的。不能移动的人输。

于是就很简单了，我们知道每一个黑块块都是独立的，所以我们可以 $O(n^2)$ 预处理出 $f[n]$ 表示长度为 n 的黑块块的 SG 函数值，然后异或一下就可以了。

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

55 Codeforces 306C：White, Black and White Again

【大意】

Polycarpus 的未来 n 天会发生 w 件两两不同的好事和 b 件两两不同坏事，每天至少发生一件事，每天要么全部发生好事要么全部发生坏事。这 n 天会先有若干天发生好事，再有若干天发生坏事，再有若干天发生好事。求方案数。注意要考虑每天发生的时间的顺序。

【解法】

一道组合数学的水题。我们可以先把好事和坏事的顺序分别排好，然后放到每一天里去。我们枚举断点为第 i 天和第 j 天，那么好事总共 $i+n-j$ 天，坏事总共 $j-i$ 天。把 m 件事情放到 k 天的方案数是 $\binom{m-1}{k-1}$，所以方案数就是

$$w!\cdot b!\cdot\sum_{i=1}^{n-2}\sum_{j=i+1}^{n-1}\binom{w-1}{i+n-j-1}\binom{b-1}{j-i-1}$$

数据范围只有 4000，想怎么暴力怎么暴力。

时间复杂度：$O(\max(w,b)\cdot n+n^2)$

空间复杂度：$O(\max(w,b)\cdot n+n^2)$

56 Codeforces 306D：Polygon

【大意】

构造一个内角相等边长互不相等的凸 n 边形。

【解法】

乱搞题。显然如果知道了边长那么我们就确定了这个多边形的形状，所以只用考虑边长。我们可以让前 $n-2$ 条边中第 i 条边的边长为 $500+0.01\cdot i$，然后最后两条边凑一下长度使得恰好能使多边形封闭。$n\leq 4$ 显然无解，其余情况使用这样子的构造可以保证 100 以内的正确性。这个正确性不用证明，用程序输出下 $1\sim 100$ 的情况发现是对的就好了。

时间复杂度：$O(n)$

空间复杂度：$O(1)$

57 Codeforces 309B：Context Advertising

【大意】

你得到了一个包含 n 个单词的文本，需要截取一段连续的单词放到广告标语中。一个正式的广告标语有恰好 r 行，每一行最多包含 c 个字符。标语中一行中的单个单词应该用空格隔开。你不能把单词断开也不能改变单词的顺序。问最多能写多少个单词。

【解法】

对每个单词求出往后至多 c 个字符能到哪个单词。然后我们就希望从每个单词出发求出这样跳 r 步落到的位置。其实这个完全可以线性求出来，不过还是来一个简单粗暴的方法吧！把跳一步看成映射，可以用一个数组表示这个映射。那么就是要求这个映射的 r 次方。使用快速幂即可。

时间复杂度：$O(n\log r+L)$

空间复杂度：$O(n+L)$

58 Codeforces 309D：Tennis Rackets

【大意】

一个正三角形的框，每边各有 n 个等距钉子把边分成 $n+1$ 段。每条边上前 m 个和后 m 个钉子不能使用，求用可使用的钉子组成的钝角三角形数目。$n\leq 32000$。

【解法】

真是无聊题啊！首先建立平面直角坐标系，假设钝角的那个顶点在底边上，然后根据钝角即点积小于 0 列出需要满足的条件。然后枚举底边和另外一边，剩下一边用公式求范围。除此之外卡常数就行了。我不明白这题意义到底何在？

时间复杂度：$O(n^2)$

空间复杂度：$O(1)$

59 Codeforces 309E：Sheep

【大意】

有 n 个区间，区间相交则有边相连。求一个 1 到 n 的排列使得有边相连的区间在排列中的位置差的绝对值的最大值最小。

【解法】

好优雅的贪心题。首先二分答案，判断答案是否可能小于等于 d。注意到由于是区间，那么我每次把一个未选的区间加入排列中，就会给周围的区间加一个 d 的倒计时，表示必须在规定时间内把区间加入排列。粗略地说，如果排列中的第一个区间在中央地带，那么倒计时就会往左右蔓延。于是既要照顾左边也要照顾右边的倒计时。但是显然我们不必这么麻烦，我们可以选一个左端的区间开始拓展。

那么我们选的第一个区间肯定是右端点最小的那个区间，因为能影响到的区间范围最少。而接下来，我们统计下倒计时。假设现在要放第 i 个，那么我们找到一个最小的 t 使得小于等于 t 的倒计时恰好有 t 个。当然了如果超过这个数肯定无解。于是我们得到了下一步只能选哪些区间。由于是从左往右拓展，所以我们只用找一个右端点最小的，它对倒计时的影响肯定是最小的。然后我们把这个区间放到第 i 个，然后更新倒计时，就可以了。

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n)$

60 Codeforces 311E：Biologist

【大意】

SmallR 是个生物学家，她可以把雄性狗变成雌性狗，反之亦然。

有 n 条狗，改变编号为 i 的狗的代价为 v_i 元，每条狗只能被改变一次。

有 m 个富人，第 i 个富人将指定确定的 k_i 只狗狗以及他希望狗狗所变成的性别，如果那些指定的狗全部变为指定的性别那么第 i 个富人会给 SmallR w_i 人民币。

如果 SmallR 不能使某个不是她朋友的人满意的话她不用支付金钱，但是如果她不能使她的好朋友满意的话她必须支付给好朋友 g 人民币。

求 SmallR 能得到的最多的钱。

【解法】

每条狗每个富人各建一个点，然后把最多钱转化成最小损失，然后用在 S 割和在 T 割表示狗改不改变性别和富人满不满意，然后就能裸最小割建图搞搞了，然后跑最大流就行了。

时间复杂度：$O((n+m)^2(n+mk))$

空间复杂度：$O(n+mk)$

61 Codeforces 314E：Sereja and Squares

【大意】

Sereja 在平面上画了 n 个点，点 i 在坐标 $(i, 0)$。然后，Sereja 给每个点标上了一个小写或大写英文字母但不会是 “x”。这些点是漂亮的，当且仅当：

- 所有的点可以被分成若干对，在每对点中，横坐标较小的会被标上小写字母，较大的会被标上大写字母。
- 如果我们以每对点为对角线的顶点画正方形，那么在所有画出的正方形中不会有相交或触碰的情况。

小 Petya 擦掉了一些小写字母和所有大写字母，现在 Sereja 想知道有多少种方法来还原每个点上的字母。

【解法】

首先，n 为奇数的话显然无解。

然后，标大小写字母是在搞笑。其实意思就是给了一个每个元素要么是左括号要么是问号的序列，要你补全成正确的括号序列，把方案数乘以 25 的 $\frac{n}{2}$ − 已知的左括号个数 次方。

那么这显然就是个很简单的 dp 能解决的事情了。但是时间复杂度是 $O(n^2)$ 的。不过没关系，这个可以过，因为出题人脑洞比较大，接下来的事情只是一些常数优化。

注意无视掉一些限制条件的话，我们的 dp 方程大概为 $f[i][j] = f[i-1][j-1]+f[i-1][j+1]$ 这样子，这意味着 $f[i][j]$ 不为 0 的必要条件是 i 和 j 的奇偶性相同。这样我们就可以无视掉奇偶性不同的那些状态。还有一个优化是，有些状态是必然无法给最终的 $f[n][0]$ 做贡献的，要满足 $j-(n-i) \leq 0$ 才可以。这样我们又无视掉了一些状态。除此之外，如果 $j > i$ 的话显然这个状态的值是 0，这样我们又少了一些状态。

总之加上这些常数优化就可以过了。感觉是一道比较无聊的题。

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

62 Codeforces 316D3：PE lesson

【大意】

有 n 个人第 i 个人拿着 i 号球，每次可以操作两个人把他们的球交换。每个人最大可能的交换次数为 1 或 2，求最后交换完后球的排列的方案数。

【解法】

称最大可能的交换次数为 1 的人为黑人。这题只跟 n 和黑人个数有关。

考虑一个置换，我们拆成循环置换，把循环拆成对换需要至少循环长度减 1 个，所以一个循环上不能超过 2 个黑人。没黑人显然合法，有一个黑人显然合法，有两个黑人就一个顺着交换一个逆着交换到最后两头交换一下就行了。所以等价于求循环中包含不超过 2 个黑人的排列的个数。

设 $f_m(z)$ 为黑人个数为 m 的时候方案数关于非黑人的个数的指数型母函数，然后我们可以得到：

$$\begin{aligned} f_m(z) &= f_{m-1}(z)\frac{1}{1-z} + f_{m-2}(z)\frac{1}{(1-z)^2}\cdot(m-1) \\ f_m(z)(1-z)^m &= f_{m-1}(z)(1-z)^{m-1} + f_{m-2}(z)(1-z)^{m-2}\cdot(m-1) \end{aligned}$$

显然 $f_0(z)=\frac{1}{1-z}$，然后后面是个与 z 无关的递推式。所以我们解出 $a_m=a_{m-1}+(m-1)a_{m-2}$ 的 a_m 然后就得到：

$$f_m(z) = a_m \cdot \frac{1}{(1-z)^{m+1}}$$

所以答案就是 $a_m \cdot \binom{n}{m}(n-m)!$。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

63 Codeforces 316E3：Summer Homework

【大意】

维护一个序列，每次可以修改一个数、区间加、查询 $\sum_{k=0}^{r-l} f_k a_{l+k}$。其中 $f_0=1, f_1=1, f_k=f_{k-1}+f_{k-2}$。

【解法】

我们可以用线段树维护 $\sum_{k=0}^{r-l} f_k a_{l+k}$ 和 $\sum_{k=0}^{r-l} f_{k+1} a_{l+k}$。由于是个线性变换所以对于任意 d 很容易推出 $\sum_{k=0}^{r-l} f_{k+d} a_{l+k}$，于是就很好合并区间信息。对于区间加 d 只要算算增量 $\sum_k f_k d$ 就好了。

时间复杂度：$O((n+q)\log n)$

空间复杂度：$O(n)$

64 Codeforces 316G3：Good Substrings

【大意】

计算出字符串 s 中不同的好的子串的个数。一共有 n 条规则，每条规则用一个三元组 (p,l,r) 表示，其中 p 是一个字符串，l 和 r 是整数。我们说字符串 t 符合规则 (p,l,r)，当且仅当字符串 t 在字符串 p 中出现的次数在 l 和 r 之间。字符串 t 是好的当且仅当它符合全部 n 条规则。$n\le 10$，字符串长度 $L\le 10^5$。

【解法】

把这些串建成一个后缀自动机，然后每个结点就表示了一些子串。用 dp 求出每个结点上的子串在每个字符串内出现的次数，然后把子串个数直接加起来就行了。

时间复杂度：$O(nL(n+\Sigma))$

空间复杂度：$O(nL(n+\Sigma))$

65 Codeforces 317C：Balance

【大意】

有 n 个注水容器通过 m 个双向输水管道形成一个图，每个容器的容积为 v 升。给你每个容器初始状态的水量 a_i 和目标状态的水量 b_i，构造一个输水方案，总步骤数不能超过 $2n^2$。

【解法】

这题比较厉害。

对于 $d \leq \frac{v}{2}$，我们可以用一个这样的算法沿某条路径 p 运输 d 升水而路径中间的点的水量不变：

1. 设运送水的算法为 $deliver([p_0, p_1, \dots p_l], d)$。
2. 如果 $a_{p_1} + d \leq v$，那么从 p_0 运送 d 升水到 p_1，然后调用 $deliver([p_1, \dots, p_l], d)$。
3. 否则，这说明 $a_{p_1} \geq \frac{v}{2}$，水资源已经足够丰富。那么我们就先调用 $deliver([p_1, \dots, p_l], d)$，然后从 p_0 运送 d 升水到 p_1。

这样我们就可以每次找一个缺水的容器和一个水过多的容器，然后找一条路径运送一下水。d 取成尽量大的数使得缺水的不会变成水过多，水过多也不至于变成缺水。如果 $d > \frac{v}{2}$，那么就拆成两次运送任务。这样每次缺水的容器和水过多的容器至少会减少一个，所以运送次数不会超过 $2n$。每次运送至多 n 步，所以不会超过 $2n^2$ 步。这样就解决了此题。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

66 Codeforces 317E：Princess and Her Shadow

【大意】

公主和影子所在的森林可以被表示为平面上一系列的整数方格，只能沿上下左右一格一格移动。一些格子长有树木，影子和公主都不允许进入这样的格子。最开始公主在 (v_x, v_y) ，而影子藏在 (s_x, s_y)。一旦公主移动一格，只要这个格子没有树木，影子也在同一方向移动一格，否则影子不移动。影子是虚无的，所以她们并不互相干扰。如果在某次移动后影子和公主在同一格子，那么影子就被抓住了。请输出一个移动方案。输入中的坐标范围绝对值 L 不超过 100。

【解法】

一个粗暴的想法是让公主追着影子跑，最后的结果就有两种，第一种是公主抓住了影子，第二种是最后两者沿同一方向移动永恒地运动下去。由于总是同方向移动肯定不会兜圈子，第一种情况以及第二种情况在永恒地运动下去前肯定不超过 $O(L^2)$ 步。如果由于永恒的运动而到了输入中的坐标范围之外，我们就可以让影子随便找几棵外围的树撞一撞，然后就抓住了影子。

不过这个算法是会超时的，优化的方法是第一次进行 bfs 求最短路，然后每次公主移动后如果影子移动了，就把这个移动方向保存到队列中。当公主到达这个位置时就按找那个方向移动。虽然不是走最短路了但是由于移动方向相同所以这样追影子的方法还是靠谱的。

时间复杂度：$O(L^2)$

空间复杂度：$O(L^2)$

67 Codeforces 319D：Have You Ever Heard About the Word?

【大意】

称一个字符串 s 为重复串当且仅当存在某个字符串 x 使得 $s = x + x$。给出一个长度为 n 的字符串，每次删除最短重复子串中最左边的那个。要求输出最后剩下的字符串。

【解法】

从小到大枚举重复子串长度 l，如果存在长度为 l 的，那么 $O(n)$ 扫一遍从左至右删掉所有长度为 l 的重复子串。这样个操作至多只会进行 $O(\sqrt{n})$ 次。这是因为最坏情况下即 $l = 1, 2, \ldots, l_m$ 时都存在重复子串，那么必然要满足 $1 + 2 + \cdots + l_m \leq n$。所以 l_m 是 $O(\sqrt{n})$ 的。

所以现在问题就是如何判定是否存在长度为 l 的重复子串。这个是个经典问题。我们在字符串上每隔 l 个位置放一个哨兵，那么可以知道重复串一定会经过恰好两个相邻哨兵。于是我们枚举这一对相邻的哨兵 i 和 $i + l$，我们只要求出这两个哨兵所在位置的最长公共前缀和最长公共后缀，然后看这两者的和是否大于等于 l 就立刻能判定是否存在经过这两个哨兵的重复串了。我们可以用哈希来求最长公共前缀和最长公共后缀。对于一个 l 的时间复杂度为 $O(\frac{n}{l} \log n)$，对于所有 l 的时间复杂度即为 $O(n \log^2 n)$。不过判定之后的删重复子串及重建才是主要的时间复杂度。

时间复杂度： $O(n\sqrt{n})$

空间复杂度： $O(n)$

68 Codeforces 319E：Ping-Pong

【大意】

给出 n 个区间，如果区间 i 的左端点或右端点严格地在区间 j 内部，那么就有一条有向边 $i \to j$。现在每次可以加一个区间（保证加入的区间的长度严格递增），或者给出两个区间 x, y 问 x 是否能到 y。

【解法】

如果两个区间严格相交（不包括包含和被包含的情况），那么他们之间就会连有一去一回两条有向边。这样的情况我们可以认为这两个区间可到达的区间集合是相等的。所以我们可以把这两个区间缩起来合并成一个大区间。

这样所有的区间之间的关系只有包含与被包含了，那么容易得出结论是，如果 x 所对应的大区间包含于 y 所对应的大区间，那么就有路可达。当然要特判两个大区间相等的情况，如果是同一个大区间那么显然可达，如果不是同一个，那么当区间 x 不等于 x 所对应的大区间时才有路可达。

接下来的问题就是如何把一个个区间合并起来。我们可以用线段树维护每个大区间覆盖了哪些位置，然后新增一个区间时，查询左右端点被哪些大区间覆盖，然后把他们都用并查集合并成一个更大区间。由于加入的区间的长度是递增的，所以不用担心现在加入的区间被其它区间包含的情况。

时间复杂度： $O(n \log n \cdot \alpha(n))$

空间复杂度： $O(n)$

69 Codeforces 321D：Ciel and Flipboard

【大意】

有一个 n 行 n 列的板子，每个格子上有一个数。n 是一个奇数，设 $l = \frac{n+1}{2}$。每次可以选择一个 l 行 l 列的子矩阵，并将其中的所有元素乘以 -1。求最大的总和。$n \leq 33$。

【解法】

我们先考虑哪些情况是可能的。从第 l 行和第 l 列把矩阵分为四块：左上，右上，左下，右下。（左边包括第 l 列，上边包括第 l 行）

首先子矩阵的左上角一定要在前 l 行 l 列，不然就超出边界了。然后一个格子最多只会被作为子矩阵的左上角一次。我们设 $x_{i,j}$ 为格子 (i,j) 被作为子矩阵左上角的次数，$v_{i,j}$ 为格子 (i,j) 最终的状态（0 为未取相反数，1 为取了相反数）。为了方便我们再对 x 做一个二维前缀和得到 s。于是一般来说：

$$v_{i,j} = s_{i,j} \oplus s_{i-l,j} \oplus s_{i,j-l} \oplus s_{i-l,j-l}$$

但是其实没有 “一般”。在大多数情况下这个式子都越界了。真实情况是：

- 左上：$v_{i,j} = s_{i,j}$
- 右上：$v_{i,j} = s_{i,l-1} \oplus s_{i,j-l}$
- 左下：$v_{i,j} = s_{i,l-1} \oplus s_{i-l,j}$
- 右下：$v_{i,j} = s_{l-1,l-1} \oplus s_{i-l,l-1} \oplus s_{l-1,j-l} \oplus s_{i-l,j-l}$。

仔细观察这些式子，看起来每个 s 对答案的贡献几乎就是独立的。我们可以枚举矩阵左上的最后一行，然后对于每一行单独处理。每次处理一行时枚举最后一列的 s，然后每个格子就可以单独处理算贡献了。

时间复杂度：$O(2^n \cdot n^2)$

空间复杂度：$O(n^2)$

70 Codeforces 323B：Tournament-graph

【大意】

构造一个结点数为 n 的竞赛图，使得任意两点间距离不超过 2。

【解法】

如果能构造出大小为 n 的，那么就能很容易构造出大小为 $n+2$ 的。找出原图中结点 1 连向的所有结点组成的集合 A 和所有连向 1 的结点组成的集合 B。易知除 1 以外的所有结点要么在 A 里面要么在 B 里面。当新增 $n+1$ 和 $n+2$ 时，我们新增边：

1. $1 \to n+1$
2. $n+1 \to n+2$
3. $n+1 \to 1$

4. $n+1 \to a, a \in A$

5. $a \to n+2, a \in A$

6. $n+2 \to b, b \in B$

7. $b \to n+1, b \in B$

然后 $n=3$ 显然有解，这样奇数都能构造出来。接着发现 $n=4$ 无解，于是尝试手玩 $n=6$，发现可以玩出来，这样大于等于 6 的偶数都能构造出来。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

71 Codeforces 323C：Two permutations

【大意】

你有两个 1 到 n 的排列 p 和 q。有 m 个询问，每次询问在 p 中位置在 $[l_1, r_1]$，在 q 中位置在 $[l_2, r_2]$ 中的数的数量。强制在线。

【解法】

显然我们可以做一个函数式线段树的前缀和维护 p 的前缀中的数在 b 序列中每个位置出现的次数。预处理 $O(n \log n)$，查询只需要查询左右端点然后相减即可。

时间复杂度：$O((n+m)\log n)$

空间复杂度：$O(n \log n)$

72 Codeforces 325C：Monsters and Diamonds

【大意】

有 n 只怪物，给出每个怪物的若干个分裂规则。每次可以选择一个怪物和它的分裂规则，让它按分裂规则分裂。每次分裂会变成一些怪物和至少一个钻石。总分裂规则个数为 m，所有分裂规则的总怪物和钻石数为 l。问对于每个怪物拆分成的钻石个数的最大值与最小值。无法全拆成钻石输出 -1，能拆成无穷多个钻石输出 -2。

【解法】

最小值比较好求，我们只要把 Dijkstra 算法稍微改改就能求出来了。如果一个怪物不可达说明最小值是 -1。对于最大值，我们从每个怪物出发记忆化搜索，如果有一个规则分裂出了最小值是 -1 的怪物，那么我们显然不能使用。如果有个规则分裂出了已经在 dfs 栈中的怪物，说明可以获得无穷多的宝石。否则把怪物的总宝石数加起来更新答案。

时间复杂度：$O((n+l)\log n)$

空间复杂度：$O(n+m+l)$

73 Codeforces 325D：Reclamation

【大意】

有一个 $n \times m$ 的地图，把左边界和右边界粘起来使得形成一个圆柱，有 q 个操作，每次操作要挖去其中的格子，如果某次操作导致无法从圆柱顶部走到底部则不做（格子四连通），问最后有多少次操作是成功的。

【解法】

这题还蛮巧妙的。没挖的格子四连通总是与挖了的格子八连通有关。在这道题上如果从顶部走不到底部，则说明圆柱上有一个环状的被挖去的格子带。严谨地说，有一个八连通的格子带绕圆柱至少一圈。

那么怎么判断是否存在这样的格子带？我们把原图横着再复制一遍拼在一起，如果有格子带那么格子带上的格子一定能从左边的该格子出发走到右边的该格子（只能八连通地走挖掉的格子，可以在拼起来的图的右边界处跳到左边界或从左边界跳到右边界）。而我们发现这是也是个充分的条件。所以我们用并查集维护被挖的格子就好了，每次只用判断新加进来的那个格子是否能从左图的该格子走到右图的该格子。

时间复杂度：$O(nm + q\alpha(q))$

空间复杂度：$O(nm)$

74 Codeforces 329D：The Evil Temple and the Moving Rocks

【大意】

有一个 $n \times n$ 的正方形房间中，四周都是围墙。

相邻的一个房间里有无数块具有魔力的石头。这些石头都属于下列四种类型之一：

- “^”：这种类型的石块将会向上移动；
- “<”：这种类型的石块将会向左移动；
- “>”：这种类型的石块将会向右移动；
- “v”：这种类型的石块将会向下移动；

你首先需要在一些格子中放上某种类型的石块（每个格子最多只能放一块石头）。然后选择其中一块石头激活它。被激活的石块将会一直朝着它的方向移动，直到撞到了其他石块或者撞到了房间四周的围墙（如果在它的方向上紧挨着就有其他石块，它将不会有任何移动）。之后这块石头将停止运动。如果它撞到了围墙，则游戏结束。否则，它所撞击到的石块将被激活，且这一过程将会持续发生。倘若石块在撞击到围墙或者其他石块之前至少移动了一步，则该次撞击将会发出一记清脆的响声。当石块撞击发出的响声达到或者超过 x 次就获得胜利。

请构造一组方案。$x \leq (\frac{n}{2})^3 - (\frac{n}{2})^2$。

【解法】

这样就行了：

solution

```
>>>>>>>>.>.>.v
^<.<.<.<<<<<<<
>>>>>>>>.>.>.v
^<.<.<.<<<<<<<
>>>>>>>>.>.>.v
^<.<.<.<<<<<<<
>>>>>>>>.>.>.v
^<.<.<.<<<<<<<
>>>>>>>>.>.>.v
^<.<.<.<<<<<<<
>>>>>>>>.>.>.v
^<.<.<.<<<<<<<
>>>>>>>>.>.>.v
^<.<.<.<<<<<<<
```

上一层撞完了会激活下一层，而每层都会产生至少 $(\frac{n}{2})^2 - (\frac{n}{2})$ 个撞击声。

时间复杂度：$O(n^2)$

空间复杂度：$O(1)$

75 Codeforces 329E：Evil

【大意】

给出平面上 n 个点，点之间的距离为曼哈顿距离，求哈密尔顿回路。

【解法】

简直就是高难度动作的贪心题啊，一不留神就会犯错。

首先要做一个变换，假想每个点都稍微抖动了非常小的一点点，这样是不会太影响答案的。所以我们可以只考虑 x, y 坐标分别互不相等的情况。

先来研究下一维情况。我们考虑一个点对答案的贡献，要么是 $-2x$ 要么是 0 要么是 $2x$。这意味着要对答案有贡献必须得让路经在这个点处拐弯而不是直接穿过。而拐弯显然不是随便就能拐的，两种方向的拐弯必须配对。所以如果是偶数那么答案就是后 $\frac{n}{2}$ 个点的 x 的两倍减去前 $\frac{n}{2}$ 个点的 x 的两倍。要是是奇数的话，我们只好让最中间的那个点被直接穿过了，于是无视中间那个点变成偶数的情况。

再来研究二维。我们先考虑 n 为偶数的情况。我们一定能找到一条竖线使得上半部分的点数等于下半部分，一定能找到一条横线使得左半部分的点数等于右半部分。这样平面被分成了四块。易知左下和右上点数相等，左上和右下点数相等。最理想的情况自然是左下和右上配对，左上和右下配对，这样所有点都是拐弯点，自然也就是最优解。但是很不幸的是，如果左下右上和左上右下都存在点，那么这个并不是哈密尔顿回路 —— 因为它压根就不连通。所以对于不连通的情况，我们必须至少用 2 条边连接左下右上和右上左下。进一步分析知也就只需要 2 条边就能连通，并且肯定是下面两种情况之一：

- 左上连左下，右上连右下。这种情况有两个点从 y 方向被直接穿过。
- 左上连右上，左下连右下。这种情况有两个点从 x 方向被直接穿过。

所以我们可以把 x, y 分别从小到大排序，然后在 $x_{n/2-1} - x_{n/2}$ 与 $y_{n/2-1} - y_{n/2}$ 中取最大值就可以了。

于是来考虑 n 为奇数的情况。同样我们用一条竖线一条横线把平面分割成四块。不过由于是奇数可能不会恰好等分，所以我们把横线和竖线都取到中位数的位置。我们只用考虑平面上的四个区域都存在点的情况，否则可以很轻松地通过往返走达到上界。显然我们要分别讨论 x 的中位数的那个点是不是 y，即讨论这两条线的交点是不是恰好是个点。假设不是个点，那么横线上恰好有个点，竖线上恰好有个点。不妨假设是横线的左半截和竖线的右半截有个点，那么我们可以知道左下和右上点数相同，而左上的点数比右下少一个。我们可以这样走：从左下出发，在左下右上间往返走遍所有点然后停在右上，接着走到横线左半截的那个点，然后走到右下，在左上和右下间往返最后停在右下，然后走到竖线的上半截那个点，再走到初始点。我们发现这样几乎在所有点都拐了弯，不过横线上的点没有 y 方向的拐弯，竖线上的点没有 x 方向的拐弯，但是本来一维情况的最优解这两个点就没在这里拐弯。所以这样自然就是最优解了。

现在我们只剩下了 n 是奇数且横竖线交叉点恰好是个点的情况。这种情况下一定有至少两条边连接交叉点。进一步分析知，肯定是下面情况之一：

- 交叉点连对角区域。这种情况下交叉点被直接穿过了，于是可以无视交叉点转化为偶数情况的问题。
- 交叉点连相邻区域。不妨设是左上和左下，那么这种情况下路径一定形如：从交叉点出发，走到左下，在左下和右上间往返最后停在右上，然后走到右下，这一步有一个点从 x 方向被直接穿过。然后在右下和左上间往返最后停在左上，然后走到交叉点。这个回路中，交叉点从 y 方向被直接穿过。

这两种情况都可以很轻易列出共 6 个式子，在这中间取最大值就好了。不过列出式子可以发现交叉点连对角区域肯定没有交叉点连相邻区域优，所以其实就只在 4 个式子中取最大值就好了。

所以综合考虑上面所有情况就能解决此题。至于时间嘛……我比较懒就写了个快排 $O(n \log n)$ 保平安了，其实可以用求序列第 k 小的那个 $O(n)$ 算法的，这样时间复杂度就是 $O(n)$ 了。

时间复杂度： $O(n \log n)$

空间复杂度： $O(n)$

76 Codeforces 331C3：The Great Julya Calendar

【大意】

一个整数 n，每次可以减去十进制表示中的一个数位的值，求变成 0 的最小步数。

【解法】

首先可以通过数学归纳法证明每次贪心地删最大的那个数位是最优的。然后我们考虑最高位和其余位，每次删最大的数位时，要么最大的数位是最高位，要么是其余位中的一位。所以我们记下前缀中最大的数位进行 dfs，每次把其余位提取出来递归下去，返回执行次数和最后多减了多少。接下来只剩最高位，先减去上一轮多减的，然后也递归下去。这个地方进行记忆化，用长度，最高位，前缀最大值，多减的值为状态就行了。

用 b 表示是几进制。

时间复杂度： $O(\log_b^2 n \cdot b^3)$

空间复杂度： $O(\log_b n \cdot b^3)$

77 Codeforces 331E2：Deja Vu

【大意】

给你一个 n 个结点的有向图，每条边上有一个长度不超过 n 的结点序列（可能为空）。如果对于一条路径，路径的结点序列和把边上的结点序列按顺序连接起来后形成的序列一样就被称为令人惊奇的。对于长度 1 到 $2n$ 分别输出有多少条令人惊奇的路径。$n \leq 50$。

【解法】

这题好厉害的。

首先我们要换一个舒服点的姿势思考问题。假设已经知道一条长度为 l 的令人惊奇的路径，我们找一张 $l \times l$ 的方格纸。然后在 $(0,0)$ 放一只蚂蚁。我们从路径的起点走到终点，每经过一个结点蚂蚁就往右走 1 格，每经过一个结点序列长度为 s 的边蚂蚁就网上走 l 格。

我们知道，一开始蚂蚁一定是 $(0,0) \to (1,0)$，结束时一定是 $(l-1,l) \to (l,l)$。这意味着蚂蚁一定某时刻穿过了对角线。显然，蚂蚁要么横着走要么竖着走，所以一定是竖着穿过了对角线。这就意味着这条路径上存在一条边 $v \to u$ 使得这条边上的结点序列中包含 v,u 这两个结点且是按顺序紧邻的，而穿过对角线就是发生在走过这条边的时候。

假设我们已经知道了这条边以及 v,u 出现的位置，由于已知的边的结点序列比走过的结点序列多，自然我们可以向左推测出一段蚂蚁的路径，向右推测出一段蚂蚁的路径。而推测不出来当且仅当已知的边的序列和结点序列一样多。这就意味着蚂蚁在对角线上。

于是我们大概知道是怎么做了——这只蚂蚁肯定会不断地在对角线上下走，而只要它不在对角线上我们就一定能推理出一段在对角线上截至的路径。所以很自然地我们可以把这条路径按对角线上的点分割开，我们就可以看见三种路径：

A. 初始时从对角线往右中途一直在对角线下方最后向上走回到对角线。

B. 初始时从对角线往上中途一直在对角线上方最后向右走回到对角线。

C. 初始时从对角线往右中途向上穿过对角线最后向右走回到对角线。

只要分别 dp 出数量然后把路径拼起来就好。

我们再考虑下怎么定位这三种路径。

A. 这种路径在原图中是一种终点被无视的有头没尾的路径。不过由于最后是往上走的，说明我们可以枚举这条路径的最后一条边，然后逆着推理出整条路径。

B. 这种路径在原图中是一种起点被无视的没头有尾的路径。不过由于开始是往上走的，说明我们可以枚举这条路径的第一条边，然后顺着推理出整条路径。

C. 这种路径在原图中是有头有尾的路径。不过由于它竖着穿过了对角线，我们可以枚举那条穿过对角线的边以及穿过的具体的序列位置，然后顺着推一下逆着推一下得到整条路径。

于是我们可以在 $O(n^3)$ 内求出每条 A 型和 B 型路径，在 $O(n^4)$ 内求出每条 C 型路径。

接下来我们考虑怎么组装这些路径。考虑到 A 是有头没尾，B 是没头有尾，C 是有头有尾，那么显然可以组装出 AA, BB, AC, CB。没尾和没头显然不能放一起，不过如果之间存在结点序列为空的边，有尾和有头还是勉强能的拼在一起的。

于是用正则表达式表示的话，一条路径一定形如 A*CB*(ZA*CB*)*（其中 Z 表示结点序列为空的边）。这样我们就可以轻松用 dp 解决了。

首先对于每个点对每个长度求出 A, B, C。然后求出 A*C, A*CB*, ZA*CB*。注意到我们最后只要统计长度为某个数的路径条数而不关心起点和终点，所以我们可以只记录长度和终点，在 $O(n^4)$ 时间内 dp 出 A*CB*(ZA*CB*)*。这样我们就解决了此题。

时间复杂度：$O(n^4)$

空间复杂度：$O(n^3)$

78 Codeforces 332D：Theft of Blueprints

【大意】

给出一个 n 个点的带权无向图，满足对于任意一个大小为 k 的顶点集合 S，恰好有一个点 v_S 与 S 每一个点都有边，定义 S 的权值为 S 中每个点与 v_S 的边权之和。求随机选一个大小为 k 的集合的期望权值。

【解法】

显然出题人玩脱了。枚举 v_S 然后在与 v_S 相邻的点中选 k 个的话一定是 v_S 跟那些点都相邻，于是算算概率乘以边的权值加到答案里就行了。

唔，这样完全没有体现出这题的神奇之处。其实满足这样的图只有三种：$k = 1$，$k = 2$，$k = n - 1$。出题人本意貌似是要我们推出这个然后搞搞，但是貌似被我水过去了。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

79 Codeforces 332E：Binary Key

【大意】

有一个长度为 n 的字符串 p，对于一个长度为 l 的 01 串，先把这个 01 串复制无穷多份按顺序拼接起来，然后把 p 中在 01 串里对应位置值为 1 的位置的字符打出来形成一个长度为 m 的字符串 s。给出 p, s, l，求字典序最小的 01 串。$n \leq 10^6$，$l \leq 2000$，$m \leq 200$。

【解法】

枚举 01 串中 1 的个素 t，那么我们就可以把周期给折叠起来，p 按周期 l 折叠形成 a，s 按周期 t 折叠形成 b，每个位置上堆了大约 $\frac{n}{l}$ 个字符。于是转化为在字符串序列 a 中找一个位置的字典序最大的子序列 b。这个直接从末尾开始贪心能取就取就行了。

时间复杂度：$O(mn)$

空间复杂度：$O(n + l)$

80 Codeforces 333C：Lucky Tickets

【大意】

构造 m 个长度为 8 的数字串使得插入插入加号、减号、乘号、括号后能得到结果 n。$n \leq 10^4, m \leq 3 \times 10^5$。

【解法】

暴力枚举每个 4 位数字串能得到的所有可能结果 r，然后前 4 位给这个数字串，后 4 位是 $n+r$。如果 $n+r$ 不是不超过 4 位的整数就不管。可以发现即使舍掉了超出范围的数，方案数也仍然是非常多的，于是就能 AC 了。

时间复杂度：$O(m)$

空间复杂度：$O(m)$

81 Codeforces 335D：Rectangles and Square

【大意】

有 n 个互不重叠的矩形，要选出一个子集使得矩形并是个正方形。$n \leq 10^5$，坐标范围为 $L \leq 3000$，坐标均位于 $[0, L]$。

【解法】

左上角一定是某个矩形的左上角，所以枚举左上角然后暴力枚举边长。判断是否合法的方式就是先做个前缀和就能知道每个子矩形里有多少个空格子，然后对每个点求出向右和向下走不穿过矩形至多能走多远，然后就能 $O(1)$ 判断了。

时间复杂度：$O(nL + L^2)$

空间复杂度：$O(n + L^2)$

82 Codeforces 335F：Buy One, Get One Free

【大意】

有 n 个物品价格分别为 $a_1, \ldots, a_n$，你每买一个物品都可以选择一个价格严格低于你买的物品的价格的物品作为赠品。求出拿走所有物品的最小花费。

【解法】

把所有物品从小到大排好序然后按价格归类，$f[i][j]$ 表示前 i 类物品全部买走且有 j 个免费拿走但是还未配对的物品。裸 DP 的话光空间就超了，但是我们可以通过递推式发现固定 i 变化 j 时，奇数部分和偶数部分分别是凸的，通过数学归纳法可以严谨地证明这一点。然后我们只用一个 treap 来维护 DP 的过程就可以了。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

83 Codeforces 338D：GCD Table

【大意】

有一个 n 行 m 列的表格 G，$G(i, j) = \gcd(i, j)$。给你一个长度为 l 的正整数数列 a，问是否在 G 的某行的连续位置出现。

【解法】

这题关键在于无视掉 gcd 的 “最大”，只当是公约数，求出需要满足的条件然后再来加强为最大公约数。

那么本来我们得到一些条件 $\gcd(x, y+k) = a_k$，我们放宽一点：$a_k|x$，$a_k|y+k$。于是最小的 x 就是所有 a_k 的最小公倍数，而最小的 y 可以用中国剩余定理求出。而这样的 (x, y) 的一般形式即为 $(px_{min}, y_{min} + qx_{min})$。我们考虑 “最大” 这个条件，显然应该让 $p = 1$ 防止有更大的公约数。我们再来看这个式子，其实 $(x_{min}, y_{min} + qx_{min}) = (x_{min}, y_{min})$，所以如果最小解是满足题目条件，那么任意的 q 都满足。注意到我们还得让 $x \le n, y \le m$，所以最小解是最有可能满足题目条件的，于是带进去检验即可。

时间复杂度：$O(l \log n)$

空间复杂度：$O(l)$

84 Codeforces 338E：Optimize!

【大意】

有长度为 m 的序列 b 和长度为 n 的序列 a，问序列 a 中有多少个长度为 m 的区间能和序列 b 匹配。两个序列中的元素 x, y 能配对当且仅当 $x + y \ge h$，两个序列能匹配当且仅当存在完备匹配。

【解法】

把序列 b 排序，求出 a 序列每个元素能与多少个序列 b 中的元素配对称为配对数。由于一定是连续的一段所以可以贪心地匹配，那么能匹配当且仅当对于每个 $1 \le i \le m$，都存在至少 $m - i + 1$ 个配对数大于等于 i 的。这个用线段树维护下就好了。

时间复杂度：$O(m + n \log m)$

空间复杂度：$O(n + m)$

85 Codeforces 339E：Three Swaps

【大意】

给出一个 $1 \ldots n$ 这个排列区间翻转 3 次后得到的排列，请输出一组区间翻转的方案，方案不超过 3 步。

【解法】

注意到每次翻转会对原序列进行分割，最后翻转完后会形成不超过 7 段，每段内是连续递增或递减，之前翻转操作总是以段为单位进行。

于是一个暴力的想法是枚举分割再枚举翻转操作。但是显然会 TLE 的。我们可以先枚举翻转操作，然后得到了每段的情况，然后尝试和原序列进行匹配。我们可以先把原序列分成若干截，每截内部连续递增或递减，然后匹配时把搜到的段能合并成连续递增或递减的就合并起来考虑，唯一讨厌的因素是我们不知道每段是不是只有一个元素，这个时候只要小小地搜索一下就行了。由于只有 3 个区间 7 个段，所以总的搜索量是非常小的。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

86 Codeforces 342D：Xenia and Dominoes

【大意】

一个 $3 \times n$ 的方格图，上面有一些黑格子、白格子和恰好一个画了圆圈的格子。要用 1×2 和 2×1 的多米诺骨牌填满所有白格子且至少有一个多米诺骨牌朝向这个圆圈。求方案数。

【解法】

记录轮廓线上的格子的被覆盖情况以及是否已有朝向圆圈的多米诺骨牌裸上轮廓线状态压缩 DP 就行了。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

87 Codeforces 348E：Pilgrims

【大意】

给出一棵 n 个结点的黑白树，现在要摧毁一个白点。摧毁后有一些黑点会不高兴。一个黑点不高兴当且仅当他不能到达任何一个在摧毁那个白点前离自己最远的黑点。请你最大化不高兴的黑点数。

【解法】

我们把 1 号结点作为根。对于一个黑点 v，如果能炸一个白点 u 让 v 不高兴，那么所有从 v 到离 v 最远的点的路径一定都要经过 u。所以我们把问题转化为求每个黑点到最远黑点的路径的交集。

考虑 dp。计算出 $down[v]$ 表示 v 向下走到黑点的最大距离，再算出 $up[v]$ 表示向上走到黑点的最大距离，就可以求出每个黑点走到黑点的最大距离。考虑下最大值在哪个子树或者哪个父亲取到，就能知道解是否唯一。如果唯一，那么求路径交集的问题就转化为求儿子的或者父亲的路径交集。

于是我们递推计算出 $cntup[v]$ 表示有多少个黑结点的路径交集会从下往上经过 v，然后再递推计算出 $cntdown[v]$ 表示有多少个黑结点的路径交集会从上往下经过 v。然后再计算下这两者和的最大值，就解决了此题。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

88 Codeforces 351D：Jeff and Removing Periods

【大意】

对一个长度为 n 的序列 a，每次可以选择一个下标为等差数列的子序列且要求这个子序列中的元素都相等，然后删掉这个子序列重新排列剩下的元素。美妙度定义为全部删除的最小步数。现在给你一个长度为 m 的序列 b，有 q 个询问，每次选一个区间询问美妙度。

【解法】

所谓的美妙度描述得那么复杂，其实就是在搞笑。当序列中不存在某种元素的下标排列成等差数列，那么答案为序列中不同元素个数 + 1，否则为序列中不同元素个数。

所以这是两个问题：求区间不同元素个数，判断区间内是否有某种元素的下标排成了等差数列。

前者是个经典问题。我们把询问按右端点从小到大排序，然后依次处理。每次到达一个新的右端点位置 i 时，找出元素 b_i 上一次出现的位置将其删掉。这样一来每次我们的问题就变成了区间内有多少个没有被删的数，可以用树状数组解决。

后者也很好处理。考虑一个位置 i，我们找出以 i 为右端点往左延伸最多能延伸多远还能保证 b_i 的下标是个等差数列，这个很好递推求出。然后我们还是把区间按右端点排序，每次到达一个新的右端点 i 时把这个最多延伸多远的距离 l 求出来，然后在一个数组 $occur$ 把区间 $[i-l,i]$ 内的数加 1。当然了，我们要把上一次右端点元素值为 b_i 时进行的这种操作给取消掉。然后每次对于一个区间询问 $[l,r]$，就转化为了问 $occur[l]$ 是否大于 0。这个显然也可以用树状数组优化。

这样就解决了此题。

时间复杂度：$O(n\log n)$

空间复杂度：$O(n)$

89 Codeforces 354D：Transferring Pyramid

【大意】

有一个金字塔，一个金字塔有 n 行，第 i 行有 i 个格子，每一行都相对于上一行左移了半格。一开始都是白色，每次可以花 3 的代价染黑一个格子，也可以花 $2+s$ 的代价染黑一个大小为 s 的子金字塔。子金字塔是某个格子分别向左下和右下走围起来的那一片区域。求染黑给定的 m 个格子的最小代价（不关心其它格子的颜色）。

【解法】

由于最裸的是 $3m$ 的代价一个一个染，所以子金字塔高度必须是 $O(\sqrt{m})$ 的。所以太高的完全不用管直接 3 的代价染掉，其它的 DP 一下就行了。

时间复杂度：$O(n\sqrt{m})$

空间复杂度：$O(n+m)$

90 Codeforces 356E：Xenia and String Problem

【大意】

如果字符串 s 满足以下条件，那么就称它为 Gray 串：

1. $|s|$ 是奇数。
2. 字符 $s[\frac{|s|+1}{2}]$ 在串 s 中只出现了一次。
3. $|s|=1$ 或子串 $s[1\ldots\frac{|s|+1}{2}-1]$ 和子串 $s[\frac{|s|+1}{2}+1\ldots|s|]$ 相同且都是 Gray 串。

定义一个串的美丽值为所有是 Gray 串的子串的长度的平方和。

给你一个由小写英文字母构成的字符串 s，允许修改串 s 中最多一个字符（也可以不修改），来使得串 s 的美丽值最大。

【解法】

其实 Gray 串的长度只有可能是 2^k-1 的形式，所以我们求出 s 的后缀数组，然后求出 $f[i][j]$ 表示 $s[i\dots i+2^j-1)$ 这一段是不是 Gray 串。然后如果不是，考虑是否能通过改一个字符变成 Gray 串，如果可以，那么记录这个修改的贡献。最后找出贡献最多的修改就行了。

时间复杂度：$O(n\log n+n\Sigma)$

空间复杂度：$O(n\log n+n\Sigma)$

91 Codeforces 360D：Levko and Sets

【大意】

有 n 个数 a_i，有 m 个数 b_j。有 n 个集合，第 i 个集合的生成方式为：

1. 初始时只有 1
2. 如果 x 在集合内，那么对于每个 j，如果 $x\cdot a_i^{b_j} \bmod p$ 不存在则加入集合。
3. 重复执行第二步直到集合无更新。

求 n 个集合的并集的大小。

【解法】

首先找个原根 g 然后取个对数再来考虑这个问题，相当于说 0 在集合内，且 x 在集合内则 $(x+b_j\log a_i) \bmod (p-1)$ 也在集合内。当然，$b_j\log a_i$ 肯定都在集合内，于是我们发现这个集合是模意义下的整数环的一个理想，其生成元就是 $b_j\log a_i$ 的最大公约数。所以我们把 b 中元素求个 gcd 得到 b_g，那么这个理想的生成元就是 $\log a_i\cdot b_g$，而大小是 $\frac{p-1}{\gcd(\log a_i\cdot b_g,p-1)}$。但是这里需要求离散对数，而求一次离散对数是很慢的。不过我们可以枚举大小 s，然后判定生成元乘以 s 是否等于 0，也就是判定 $a_i^{sb_g}$ 是否等于 1。s 肯定是 $p-1$ 的约数，找到最小的 s 就行了。并且，$\frac{p-1}{s}$ 也是一个生成元。

于是我们现在知道了 n 个理想，现在要求并集。我们可以通过容斥 + dp 解决。

时间复杂度：$O(m\log b_{max}+\sqrt{p}+n\cdot\tau(p-1)\cdot\log p)$

空间复杂度：$O(n+m)$

92 Codeforces 360E：Levko and Game

【大意】

有 n 个点 m 条边，有两个人分别从 v_a 和 v_b 出发到 v_t 去。还有 s 条权值未确定的边，每条边有个权值范围 $[w_l,w_r]$，第一个人可以确定这些边的权值。问第一个人能否比第二个人先到，或者是否能打成平局。

【解法】

对于一组解，我们可以把第二个人走过的边都调整成上界，答案一定不会变差。而两个人都没走过的边显然调整成上界不会影响答案。所以我们按边的上界求每个点 v 到 v_b 的最短距离 $d_b[v]$，然后假想调整边权后的图，求每个点到 v_t 的距离 d'，那么每个点 v 都要满足 $d_b[v]+d'[v]\geq d_b[v_t]$。于是从 v_t 开始在反向图上做最短路算法，每次到达 v 时考虑边 (u,v,w)，如果 w 固定那么直接求，如果是个区间那么我们考虑 $d_b[u]+d'[u]\geq d_b[v_t]$，即 $d_b[u]+d'[v]+w\geq d_b[v_t]$ 也即 $w\geq d_b[v_t]-d_b[u]-d'[v]$。考虑到 $d'[v]\geq d_b[v_t]-d_b[v]$ 以及 $d_b[v]\leq d_b[u]+w_r$，所以 $d_b[v_t]-d_b[u]-d'[v]\leq w_r$，所以在这个下界和 w_l 间取最小值作为边权就行了。

时间复杂度：$O((n+m)\log n)$

空间复杂度：$O(n+m)$

93 USACO Mar 08：Land Acquisition

【大意】

有 n 个矩形，拿走一些矩形的代价为最大的宽度乘以最大的高度，问取完的最小代价。

【解法】

如果有一个矩形的宽度和高度都小于等于另一个矩形，那么我们显然不用考虑这个矩形。于是剩下的矩形按宽度递增排序，那么高度递减。

于是就可以很简单地用 DP 求出最小代价，然后用斜率优化搞搞就行了。

时间复杂度：$O(n\log n)$

空间复杂度：$O(n)$

94 USACO Nov 08：Toys

【大意】

有 n 天需要提供玩具，第 i 天需要 a_i 个玩具。每天能从商店以 d 美元买玩具，或者通过消毒重用使用过的玩具。有两种消毒方式，第一种方式需要收费 c_1 美元, 需要 m_1 个晚上的时间，第二种方式需要收费 c_2 美元, 需要 m_2 个晚上的时间。求最小花费。

【解法】

这题我第一次见是在网络流 24 题。可以看出很明显的网络流的感觉，于是可以构个图用最小费用最大流解决。但是对于这题不行，时间复杂度太高。

不妨设 $c_1\leq c_2$，假如 $m_1\leq m_2$，那么第二种方式显然是从来都不应该被使用的，于是强行把 m_2 置为 m_1 也不会影响答案。那么现在只用考虑 $c_1\leq c_2, m_1>m_2$ 的情况。

其实买玩具这一行为很棘手，不过我们知道中途买玩具是毫无意义的。假设已经知道要买 n_t 个玩具，我们来考虑消毒的过程。我们把消毒想象成瞬间的过程，但规则是，放了至少 c_1 天花费 m_1 元，放了至少 c_2 天花费 m_2 元。这样很容易得到贪心的策略：初始 n_t 个新玩具。对于每一天，如果当前有剩余的新玩具，就用。如果不够，尽量洗放了至少 c_2 天的玩具。如果还不够，尽量洗放了至少 c_1 天的玩具。如果还不够，说明初始玩具量太少，返回最少钱数 $+\infty$。

于是只要枚举买来的玩具数又可以得到一个正确的算法，但是时间复杂度也太高。但是不怕，我们有最小费用最大流！有一个性质是，每次按最小费用增广时，费用是递增的。实际意义是，假设每次增加一个玩具，那么最少钱数的减少量会一次比一次少。换句话说，这是个凸函数。所以我们三分买来的玩具数，下界为 0，上界为 $S_a = \sum_{k=1}^{n} a_k$ 就可以了。

时间复杂度：$O(n \log S_a)$

空间复杂度：$O(n)$

95 USACO Dec 08：Fence

【大意】

给定 n 个点，问最多选取其中多少个点可以构成一个凸多边形。$n \leq 250$。

【解法】

我们枚举凸多边形上 x 最小的点 s，然后把右边的点按极角排序，凸多边形上的点一定会按极角序出现。

所以 $f[i][j]$ 表示走到 i 号点，前一个点是 j 号点的最长路径，走的时候判一下叉积是否大于 0。这样就得到了 $O(n^4)$ 的算法。

换一下 DP 的顺序，按 j 递增递推，每次要考虑的就是关于 j 的极角序了。我们发现每次要利用关于 s 的极角序比 j 小的 k 的 $f[j][k]$，算关于 s 的极角序比 j 大的 i 的 $f[i][j]$。由于 k 关于 s 的极角序比 j 小所以在同一个半平面里，每次对于一个 i 而言合法的 k 总是一个前缀，所以把 i 和 k 分别按关于 j 的极角序排序然后扫一扫就行了。

当然预处理关于一个点的极角序可以做到 $O(n^3)$，但是我太懒了写的 $O(n^3 \log n)$。

时间复杂度：$O(n^3 \log n)$

空间复杂度：$O(n^2)$

96 USACO Open 08：Cow Neighborhoods

【大意】

给出平面上 n 个点，曼哈顿距离不超过 c 的之间有边，求连通块个数及最大的连通块大小。

【解法】

一个经典的做法是转坐标系把 (x, y) 变为 $(x + y, x - y)$，这样就从两维坐标差的绝对值之和变为了绝对值的最大值。

于是可以按 x 排序依次加入，考虑现在要加入 p，p 要给它右侧的那个 $c \times 2c$ 的矩形内的点连边。而我们发现那个矩形内绝对放不下三个两两距离超过 c 的点。所以可以用一个 y 坐标的平衡树存下左边 x 坐标差不超过 c 所有点，然后在那个矩形内找最上面的和最下面的用并查集把 p 和它们缩起来就行了。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

97 USACO Open 09：Tower of Hay

【大意】

有 n 个干草堆排成一行，每堆有个宽度 a_i。现在要且分成若干段，把每一段的干草拼起来然后逐段堆砌，越左的越低，且宽度要逐层非严格递减。求层数最多是多少层。

【解法】

首先把序列倒转过来，然后做前缀和，然后可以列出一个很裸的 dp：

$$f[i][j] = \max f[j][k] + 1 \qquad (s_i - s_j \le s_j - s_k)$$

然后打个表发现固定了 i 后，$f[i][j]$ 总是递增的，不过严谨地说，应该是忽略所有的 $-\infty$ 即无解情况后，$f[i][j]$ 总是递增的。

我们可以用数学归纳法证明这一点。对于 $i = 0$ 显然成立。假设小于 i 的都满足这个性质，我们来证对于 i 成立。我们设 l_j 表示 $f[j][k] \neq -\infty$ 的最大的 k。我们发现原 dp 方程的意思就是：

$$f[i][j] = \begin{cases} f[j][l_j] + 1 & \text{if } s_i - s_j \le s_j - s_{l_j}, \\ -\infty & \text{else.} \end{cases}$$

当我们固定了 k，我们增大 j，那么 $f[j][k]$ 一定是非严格递增的 —— 底部变大总是没坏处的。这就说明了当 j 增大时，l_j 也是非严格递增的。

我们考虑一个不为 $-\infty$ 的 $f[i][p]$ 和一个不为 $-\infty$ 的 $f[i][q]$ 且 $p > q$。我们知道 $f[i][p] = f[p][l_p] + 1$，$f[i][q] = f[q][l_q] + 1$。我们又知道 $f[p][l_p] \ge f[p][l_q] \ge f[q][l_q]$，所以 $f[i][p] \ge f[i][q]$。这样我们就证明了这个结论对于 i 也成立。

我们可以仅递推 $f[i][l_i]$，加个单调队列优化下，就可以解决此题。当然这个结论形象点说就是一定存在一个层数最高的解使得它也是底部宽度最低的解。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

98 USACO Dec 12：First!

【大意】

有 n 个字符串总长为 L，问每个字符串是否能通过改变字母表顺序让这个字符串成为其中字典序最小的。

【解法】

如果有一个字符串的前缀是另一个字符串那么显然不行。否则，预处理所有字符串组成的 Trie 树，考虑每个结点所对应的 Trie 树路径上每个结点的兄弟，可以得到一系列 “某个字符要小于某个字符” 的关系，于是搞个有向图表示关系然后判断是否是拓扑图。

时间复杂度：$O(L\Sigma + n\Sigma^2)$

空间复杂度：$O(L\Sigma + n)$

99 GCJ 2012 Final D：Twirling Towards Freedom

【大意】

银河是一个二维的平面。太空飞船从原点出发。银河中有 n 颗恒星。每一分钟你可以选择一颗恒星，然后围绕这颗恒星顺时针旋转 90°。你也可以选择停在原地。求 m 分钟后最远能离开原点多远呢？

【解法】

关键要注意到可以用复数描述旋转。顺时针转 90° 就是乘以 $-i$，所以 p 绕 p_0 转 90° 就得到的是 $(p-p_0)\cdot(-i)+p_0=-ip+(1+i)p_0$。

所以如果依次绕 $p_1,p_2,\dots p_l$ 旋转得到的结果就是：

$$\sum_{k=1}^{l}(1+i)p_k\cdot(-i)^{l-k} \tag{1}$$

由于 $(-1)^k$ 的周期为 4，于是我们把这个求和分成四组。所以我们把给定的每个点乘以 $(1+i)$ 然后复制成四组，我们就需要在每组中选对应的个数个向量使得加起来的和向量最大。

这是个经典问题，假如已知和向量方向那么显然选点积最大的，所以相当于从四组中每组一个向量使得乘以对应个数后和向量最大。由于点积最大可能出现在凸包上，所以这四组同时按极角序转一转搞搞然后更新答案就好了。（当然中途允许不走，所以需要枚举 $l=m-3,m-2,m-1,m$。这是因为如果确定了要取哪几个向量之后显然每过一个周期增加的是一个固定的向量，所以如果减少超过 4 步显然毫无意义）

时间复杂度：$O(n\log n)$

空间复杂度：$O(n)$

100 GCJ 2013 Final C：X Marks the Spot

【大意】

有 $4n$ 个点，没有三点共线。要画两条互相垂直的直线使得分成的四块里的点数相等。（不能有点在直线上）

【解法】

如果已经知道直线的倾斜角 α 那么显然应该取两个方向的中位数作为中心，那么位置也确定了。

对于任意一个 α 我们都这样按中位数划分一下得到四块，设左上角的点数为 $f(\alpha)$，那么其它三块分别为 $2n-f(\alpha),2n-f(\alpha),f(\alpha)$。所以原问题就是要解方程 $f(\alpha)=n$。

考虑一个 α，如果有点在直线上会导致该位置的 $f(\alpha)$ 未定义。考虑未定义的 α 的前面一点点和后面一点点。因为没有三点共线所以至多只有 2 个点在同一条直线上。如果一条直线上有两个点，那么显然前后的 f 值相差不超过 1，如果是两条直线上都有两个点，那么相差不超过 2。而当相差为 2 时我们仔细观察发现其实一进一出抵消了，所以其实相差不超过 1。

得到了这个神奇的结论后，只要用二分法解方程就行了。初始二分区间选在 $[0,\frac{\pi}{2}]$，这是因为 $f(0)=2n-f(\frac{\pi}{2})$，所以如果有等于 n 的那么就找到了解，否则左右端点一个大于 n 一个小

于 n，肯定在中途某处有等于 n 的。要注意如果碰上了倒霉的未定义，只要在旁边稍微抖动一下就行了。（以及要注意精度问题）

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$