IOI2015中国国家集训队第一轮作业
试题范做

长沙市雅礼中学 刘研绎

2015 年 1 月 29 日

目录

第一部分　GoogleCodeJam Finals 2014

1　2014A　Checkerboard Matrix

1.1　试题大意

给出一个$2N \cdot 2N$的矩阵，每个元素是0或者1，每一步操作可以交换任意两行或者任意两列，目标是一个01相间的矩阵，求最小交换次数或者告知无解。

$1 \leq N \leq 10^3$

1.2　算法讨论

首先可以判断出一些简单的无解情况，比如某行或者某列01的个数不等之类的。

可以任意交换两行或者两列这是一个非常宽的条件，需要做的是发现这种操作的性质。不妨考虑最终的情况，可以发现本质不同的行实际上只有2种（不妨以第一个元素称为0行和1行），同理列也只有2种，并且我们任取一列就可以推出剩下的每一列属于哪种。这个性质同样适用于被这种操作任意打乱的矩阵上（要求达到目标可以看做将目标打乱之后再复原），这个性质看起来没有什么用但事实上利用这个就可以解决这个问题了。

根据上面的性质，假设已知某一行（这一行的具体位置是无所谓的，因为01相间的矩阵有两种）最终是0行还是1行，就可以通过这一行推出剩下所有矩阵的情况，于是可以计算出哪些行哪些列处于正确的位置上，同时最小交换次数也可以计算出来了。同时也可以判断出是否有解，确定了所处的行是0/1行，列是0/1列就可以推算出当前位置上应该出现的数字是多少。

最后的问题是，我们并不清楚某一行最终是哪一种行，枚举一下第一行属于哪种情况即可。

问题至此圆满解决。

1.3　时空复杂度

时间复杂度：$O(N^2)$

空间复杂度：$O(N^2)$

1.4 备注

我觉得这题难度很大，如果要迅速解决这道题必须保证有清晰的思路，也许是我的思路不够清晰的问题吧。

2 2014B Power Swapper

2.1 试题大意

给出一个2^N的排列，定义一种交换操作：

- 选出一个整数k。
- 将排列从第一个数开始，每2^k分成一段。
- 选择两个不同的段交换。

你希望利用这种操作对排列进行排序（从小到大），对于每一个k只能在一次交换操作中被选出，问有多少种方法能够将排列排序。两种方法不同当且仅当交换序列不同。

$1 \leq N \leq 12$

2.2 算法讨论

不难发现，如果我们假设操作是有序的会给问题的解决带来好处，不妨假设操作依次按k从小到大进行，最后再乘上排列数即可。

如果当前正在进行$k = i$的交换操作，那么对于任意一个从第一个数开始，每2^{i-1}分成的一段来说，他一定是有序的，同理进行完$k = i$的交换操作之后，每2^i分成的一段也是有序的，这提示我们对于第i次操作来说，能够操作的方案是有很大限制的。仔细分析一下便可以得出，可能的操作方案不超过两个。

别忘了$N \leq 12$，我们可以依次搜索每一步操作。

至此问题便圆满解决。

2.3 时空复杂度

时间复杂度：$O(2^{2N})$

空间复杂度：$O(2^N)$

2.4 备注

这个问题的关键在于有序假设之后发现那个性质，之后的问题就迎刃而解了。

3 2014C Symmetric Trees

3.1 试题大意

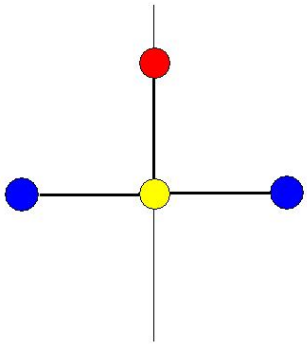
给出一棵N个点的树，每个点有一个颜色c_i，问其是否能够画在一个二维平面上并且轴对称。

更正式的，给出轴对称的要求：

- 所有位置互不相同。
- 如果颜色为C的节点v_i位于坐标(x_i, y_i)，那么一定有一个节点v'_i，颜色也为C，并且位于坐标$(-x_i, y_i)$。
- 如果(v_i, v_j)有边，那么(v'_i, v'_j)也有边。
- 如果边看做连接两点的线段，不能有两条边有交点。

$2 \leq N \leq 10^4$

下图是一个$N = 4$的树的例子。



3.2 算法讨论

假设我们已经知道了最终情况下对称轴上有哪些点，那么我们可以利用树的hash来快速判断左右两边是否相等。

那么现在的问题变成需要在树上找到一条合法的路径，使得将其放在对称轴上之后左右对称。考虑利用树形DP解决这个问题，对于一棵子树，我们只需要知道是否存在一个从根往下的路径能够作为对称轴，即令f_i为以i为根的子树是否存在一个从i往下的路径能够作为对称轴。转移的话便是枚举一个f_x为真的儿子，再判断剩下的子树是否能够轴对称即可。最后我们枚举这条路径上深度最浅的点，在这里把两条链合并即可。

关于树hash，我们可以将对应的括号序列hash起来，也可以直接利用多项式hash那样的方法来做，只是后者错误的概率比较大。

3.3 时空复杂度

时间复杂度：$O(N)$
空间复杂度：$O(N)$

3.4 备注

感觉思路比较简单清晰的题。

4 2014D Paradox Sort

4.1 试题大意

你有N颗糖果，你希望将所有的糖果按某种顺序（即找出一种排列），依次给Vlad。每当你给一颗糖果给他，如果Vlad手中没有糖，他会收下这个糖果，否则他会把你给他的糖果和他手中的糖果进行比较，留下其中更喜欢的一个。但Vlad的喜欢并不是严格的序关系，而是将由一个表格提供。

有一个特定的糖A，你希望再你把所有的糖给Vlad之后他能把这颗糖留在手上。希望你求出字典序最小的给糖果的顺序（一个排列）使得A最终留在Vlad手上。

$1 \leq N \leq 100$

4.2 算法讨论

从D开始就比较厉害了。

不妨先不管字典序问题，思考如何构造出一组解。可以将喜欢的关系转化成一个竞赛图，即对于x和y，如果更喜欢x，则从x向y连一条有向边。那么你需要构造一条路径，使得这个路径覆盖了所有的点（所有的点都可以由这个路径上的某一点再走一条边到达）并且它的起点是A。虽然将问题抽象到了这个程度，但是仍然无法一眼看出应该用怎样的算法。

考虑判断是否有解，我们只需要从A开始，如果不能到达所有的点，就一定无解了。到这里再往下思考就需要仔细观察，有一个不是很好发现的性质：我们需要找的路径就是DFS树的右链（不妨把由DFS的第一个儿子所组成的链叫左链，最后一个儿子所组成的链叫右链），需要对DFS/Tarjan算法比较熟悉才能快速发现这个性质（同时别忘了，这一定是一个完全图）。

剩下的问题就是如何构造字典序最小的解，考虑采用直接枚举的算法。那么问题变成前面一段糖果已经被固定时，是否存在一种顺序合法。既然已经固定，不妨将不必要的点删去，不难发现我们只需保留将已经固定顺序的糖果依次给Vlad后最终他手中拿的那颗糖果（假设是糖果B），同时也可以删去所有B能够到达的点，最终需要使B位于某棵DFS树的右链的底端。看上去构造很复杂但是注意我们已经把所有B能够到达的点都删掉了，事实上我们可以直接把B删掉之后在任意进行一次DFS，直接在得到的右链底端加上B就可以了。（这里要注意一种$B = A$的特殊情况，稍微判断一下就行了，同时还要保证所有点均被遍历到）。

至此问题圆满解决。

4.3 时空复杂度

时间复杂度：$O(N^4)$
空间复杂度：$O(N^4)$

4.4 备注

感觉这个题往后就都是些神题了，做不动啊。这个题主要是得对DFS/Tarjan算法熟悉，那个性质我觉得很难发现。GCJWoldFinal还是太强了。

5 2014E Allergy Testing

5.1 试题大意

YDC打算吃n种药，但其中一种会有过敏反应而YDC并不知道具体是哪一种。YDC打算做一些试吃实验判断出是哪一种药过敏，她会做一些实验。

每一次实验她会选择一些药吃掉，然后等待A天，如果这些药里面没有过敏反应，那么她可以进行下一次实验了。否则需要再等待$B-A$天，等待过敏反应消失之后才能进行下一次实验。

给出n, A, B，问最少需要多少天能够判断出过敏的药是哪一种。

$1 \leq n \leq 10^{15}, 1 \leq A \leq B \leq 10^{12}$

5.2 算法讨论

首先我们需要一些对题目的大致观察。

n种药在实验之前是没有区别的，如果采用二分的策略，每次试吃一半的药，所需的时间是$B \log n$，也就是一个答案的上界。

考虑直接进行DP，令f_i为i种药最少需要f_i天才能判断出具体是哪一个过敏，可以得到转移方程：

$$f_i = \min_j \{\max\{f_j + A, f_{i-j} + B\}\}$$

很显然这个DP毫无前途，n巨大的范围几乎已经枪毙了这个算法。考虑其实我们只需知道当前有n种药需要尝试时，应该选出多少药来进行试吃，之后就能缩小问题规模了，而对于两种试吃结果，我们希望同时最小化之后所需是时间，不妨尝试让DP本身进行选择。考虑将DP的状态和DP值互换，令g_i表示我们有i天，最多能够鉴别出多少种药中的过敏药物，那么转移方程就很简单了：

$$g_i = \begin{cases} 1 & i < A \\ g_{i-A} + 1 & A \leq i < B \\ g_{i-A} + g_{i-B} & B \leq i \end{cases}$$

有了这个DP方程之后我们就可以解决小数据了。但大数据仍然是个问题。借助二分答案，我们需要做的就是快速计算出第$k(k \leq 10^{18})$项的值。

这种递推式肯定没有什么快速求值的办法，有一种思路是根据答案不超过$B\log n$来矩阵乘法，但并不优美。

假设我们二分出的答案为m。我们仔细观察递推式，不难发现需要计算的实际上是：对于每个$1 \le i < B$，每次可以$+A$或者$+B$，有多少种方法可以得到m。不妨考虑固定起点i，注意到答案的上界不超过$B\log n$，那么B出现的次数不超过$\log n$次，可以强行枚举B出现的次数，接下来的事情就是摆放$+A, +B$的位置，这其实是一个组合数。

不妨令选取了x个A，y个B，那么我们要计算：

$$\sum_{x,y} \binom{x+y}{x}$$

其中x, y满足$m - B < xA + yB \le m$。

固定y，我们可以快速算出x的上下界，那么

$$\sum_{y} \sum_{x=xmin}^{xmax} \binom{x+y}{y}$$

$$\Rightarrow \sum_{y} \binom{xmax+y+1}{y+1} - \binom{xmin+y}{y+1}$$

问题至此圆满解决。

5.3 时空复杂度

时间复杂度：$O(\log b \log^2 n)$

空间复杂度：$O(1)$

5.4 备注

这题相当炫酷，太帅了，最后这步转化非常漂亮，真是不知道tourist是怎么直接切掉的，总之就是厉害。

6 2014F ARAM

6.1 试题大意

你正在玩一种游戏，这个游戏里有n个英雄，每一局你会得到一个随机的英雄，你将使用这个英雄进行游戏。当然不同的英雄获胜的概率可能不

一样，因此每个英雄有一个获胜概率p_i。如果你选到一个你不喜欢的英雄，你不一定非得使用它，这个游戏包含一个“重新选择”功能。

具体来说，你在进行第一局游戏之前，你会得到R美刀，之后进行每一局游戏，你都可以花G美刀进行一次重新选择，即再进行一次随机分配，并且只要你仍然拥有G美刀，你就可以再重新选择一次。进行完这局游戏之后你会得到1美刀，但是任何时候你的美刀不会超过R（即使你在拥有R美刀的时候获得了1美刀，你仍然只会拥有R美刀）。

举个例子，如果$R = 2, G = 1$，你在第一局游戏中没有使用重新选择，那么第二局开始时你将仍然只有2美刀。在第二局你使用2次重新选择，那么第三局开始时你将拥有1美刀。

你将进行10^{100}局游戏，问最优策略下获胜的最大期望，如果你的答案的相对或者绝对误差与标准答案不超过10^{-10}，你将被算作正确。。

$1 \le n \le 1000, 1 \le G \le 20, 1 \le R \le G \cdot 20$

6.2 算法讨论

10^{100}是一个非常大的数字，不妨将其考虑为无限。

那么我们所作出的是否“重新选择”的决策，肯定只和当前拥有的美刀数以及当前选择到了怎样的英雄相关，与还剩多少局游戏是无关的（可以看做做出每个决策之后仍然有无限局游戏）。假设固定美刀数i，那么决策肯定是如果当前得到的英雄获胜概率大于g_i，就不进行重新选择了，否则就重新选择一次。也就是说我们需要计算的是对于每一种当前拥有的美刀数i所对应的g_i。

这样思考后发现，这对解决问题似乎没有直接作用，仍然不知道如何下手。看起来直接DP都是困难重重，虽然可以看做每一局游戏之后都要进行无限局，但是如果不和剩下的局数相关，当前拥有的美刀数是一个没有意义的状态。假设将之后仍然剩下的局数加入状态，似乎也无法直接用$f_{i,j}$进行DP，不如用最暴力的方法，将每个英雄使用的次数全部记录下来，不妨令第k个英雄被期望使用了$c_{i,j,k}$局游戏，那么获胜的概率为$\frac{\sum_k c_{i,j,k} \cdot p_k}{i} \to \frac{\sum_k c_{i,j,k} \cdot p_k}{\sum_k c_{i,j,k}}$。仔细观察后不难发现，这非常类似（其实就是）一个分数规划问题，我们遇到的困难和分数规划时设计状态的困难一致，那么我们二分答案P，之后使用每个英雄有一个收益$p_i - P$，只需要最大化期望收益之和就行了。

于是我们可以直接暴力DP了，现在的问题需要改变状态设计的方法，必须不与剩下的局数相关才行。考虑我们当前有的美刀数为i，假设我们不使用“重新选择”，那么下一回合时将有$i+1$美刀，如果使用了重新选择，那么不仅当前美刀数变为了$i-G$，而且如果某一时刻我们拥有了$i+1$美刀，这期间所有游戏的期望收益之和一定要比不使用“重新选择”更大，否则没有必要进行这一步操作，那么我们似乎找到了新的思路。

不妨令f_i表示我们从拥有i美刀到拥有$i+1$美刀进行的所有游戏的期望收益之和最大是多少，这样我们就设计出了一个与剩下的回合数不相干的状态，考虑他的转移方程。

当$i<G$时我们没有选择的机会：

$$f_i=\frac{1}{n}\sum_{i=1}^{n}p_i-P$$

当$G\le i<R$时，此时我们需要决策g_i，不妨将p排序，那么令j为当选到最差的j个英雄时我们将重新选择，不妨枚举j，那么：

$$f_i=\max_j\{\frac{j}{n}\sum_{k=0}^{G}f_{i-1+k}+\frac{1}{n}\sum_{k=j+1}^{n}(p_i-P)\}$$

特别的，当$i=R$的时候，因为拥有的美刀数不会超过R，所以：

$$f_i=\max_j\{\frac{j}{n}\sum_{k=0}^{G-1}f_{i-1+k}+\frac{1}{n}\sum_{k=j+1}^{n}(p_i-P)\}$$

那么我们二分答案，当f_R大于0时认为P不够大，当f_R小于0时认为P太大了就行了。

至此问题圆满解决。

6.3 时空复杂度

时间复杂度：$O(R\cdot n)$

空间复杂度：$O(n)$

6.4 备注

这题太神了，其中二分答案和最后状态的设计非常的精彩，是解决本题的关键。小数据部分估计是没有设计出最后的状态而是暴力进行很多次DP得到的吧。

第二部分　GoogleCodeJam Finals 2013

这年的题和14年的比简直是小巫见大巫，感觉都不是一个档次上的。

7　2013A　Graduation Requirements

7.1　试题大意

有一个交通环岛，有n条路与环岛相连，顺时针标号为1到n。一般车辆都是逆时针绕环岛行驶，但你想在环岛上逆行。你知道一共有m台车会进入环岛，并且知道每台车进入环岛的时间和入口，以及出口（不会有车绕环岛一周以上），你不希望与其中任何一台车相遇，不论是在环岛上还是进出环岛时（你不能与某车同时进入环岛，并且某条路上有车从环岛上下来时你也不能从这条路进入环岛，出环岛同理）。另外环岛只在0时刻至x时刻开放，你希望知道在环岛上的最长逆行时间（你一旦从环岛上下来就不允许上去了）。所有的车（包括你的）的速度均为1，即每个单位时间往前移动一个路口，你和车辆都只能在整数时刻进入或者离开环岛。

$3 \leq n \leq 10^{10}, 1 \leq x \leq 10^{10}, 0 \leq m \leq 1000$

7.2　算法讨论

注意到虽然任意一台车不会绕环岛一周以上，但是你是可以的。

不妨抽象一下模型，用图象表示车的运动轨迹。对于一台车，如果他在时刻t出现在路口x，那么就在平面的(x,t)上画一个点，将同一台车画出的点连起来就得到了一个这台车的运动轨迹，如果有车从n号路口开至1号路口，不妨将其拆成两半。可以发现每台车的轨迹都是一个斜率为1的直线。注意到我们可以绕环岛很多圈，那么我们将得到的图象复制x/n遍，再将这些图象从左到右拼在一起，得到一个$x \cdot x$的图象，我们要做的就是在这个图象里找一条最长的斜率为-1的线段，让他和原有的线段都不相交。

考虑转化后的问题，不妨将图象旋转45°，这样问题会变成：给出N条水平的线段，要求画一条竖直的线段，使其不和任意一个水平线段相交。对于这个问题，可以发现一定有一个最长的竖直线段和某个水平的线段相切（当然在本题里相切是不允许的）。那么我们可以得到一个$O(N^2)$的解法，枚举每条线段的左右两端再计算过这个点的竖直线段最长能够延伸多

少就行了。

但注意到，这里的N已经不是原题中的线段个数了，实际上$N = m \cdot x/n$，可以达到一个很大的数量级。

如果已经得出了答案，并且将答案所对应的线段画在图像上，再将图象“复制”回去，得到一个“循环图象”，即左右相接，不难发现一定存在一条最优解对应的线段经过当前存在的线段的两端，也就是说我们没有必要在复制出来的图上枚举，直接在这张“循环图象”上枚举就行了。

这样线段数是m，复杂度就可以做到$O(m^2)$了，需要注意的是他是一个$[1, n+1)$的循环图象，虽然没有$n+1$号点但是$(n, n+1)$这一段的图象还是必须画出来的（考虑你出现在路口1，同时有车出现在n，然后你们相向而行的情况）。

问题至此圆满解决。

7.3 时空复杂度

时间复杂度：$O(m^2)$
空间复杂度：$O(m)$

7.4 备注

写起来还是比较恶心，我写了比较久才把代码写出来。感觉思路还是非常有趣的，一定要注意最后所说的特殊情况。当然不要学习主人公的做法，在马路上逆行是作死行为。

8 2013B Drummer

8.1 试题大意

有很多鼓手来报名参加你组建的乐队，你决定给他们进行一次面试。对于一个鼓手，你让他敲了n次节奏，敲击的时间分别为$b_0, b_1, \ldots, b_{n-1}$，这便是这个鼓手的节拍序列。但一个完美的节拍序列应该是形如$a_i = a_0 + i \cdot d$的等差数列，但是对于人类来说这太难了，所以用一个数E来判断这个节拍序列的完美程度。对于一个节拍序列b，

$$E = \min_{a_0, d} \left\{ \min\{e | \forall_{0 \le i < n}\ a_0 + i \cdot d - e \le b \le a_0 + i \cdot d + e\} \right\}$$

你希望计算出这个鼓手的完美程度是多少，注意E, a_0, k可以是实数。

$1 \leq n \leq 5 \cdot 10^4$

8.2 算法讨论

考虑把一个节拍序列画在图象上，用(i, b_i)来表示第i个拍子，那么一条完美的节拍序列对应的一条直线。

那么我们要做的就是找出与当前鼓手最相符的完美序列，不妨另其为$y = kx + b$，这样有两个量需要我们确定，分别是k, b。不妨假定b已经确定，此时E的值显然是可以二分的，二分E之后对于每个鼓手打的拍子(i, b_i)，可以确定k的上下界，只要仍有可以取值的k，那么E便是可行的。但是我们似乎无法发现b的更多性质，换一种思路考虑固定k。

固定k之后，由于我们并不方便直接确定b，不妨对当前存在的所有点做一个斜率等于k的直线，这些直线与y轴会有很多交点，很显然我们需要用$[b - E, b + E]$覆盖所有点，那么b, E的值就直接确定了。

到这一步之后我们观察问题，实际上我们可以将问题转化为：将所有鼓手打的拍子转化成点之后求他们的凸包，我们得到了一个多边形，而在无穷远处有一个点光源，可以认为她发出的光都是平行光，我们要使多边形在y轴上的投影尽可能的小。不妨先考虑一条直线在y轴上的投影与平行光和y轴的夹角的关系，很显然这是一个凸函数，而整个多边形的投影其实是在所有包含在多边形内部的线段投影的最大值，也就是很多凸函数的最值，这个函数肯定也是凸的。那么我们可以直接三分斜率暴力计算了（其实在凸包上进行类似旋转卡壳的算法应该也是可以的）。

至此问题圆满解决。

8.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

8.4 备注

这是最简单的一道题，感觉还是很有算法题的样子。最后的模型转化非常炫酷，但我过掉这道题的时候并没有想这么多。

9 2013C X Marks the Spot

9.1 试题大意

平面上$4N$个点，你需要用两条互相垂直的直线将他们分成N份，每份N个点。保证没有三点共线。

任意输出一种方案即可，你需要输出他们的交点和任意一个在他们上的点。

$1 \leq N \leq 2500, -10^6 \leq x_i, y_i \leq 10^6$

9.2 算法讨论

这题有一个非常需要重视的地方，虽然$N \leq 2500$但你应该会对$4N$的数据量进行计算，而$n = 4N = 10000$这是不能用$O(n^2)$这样的复杂度通过的。但是如果只是codejam的话你可以考虑试一试$O(n^2)$算法。

不妨先将问题简化，单独考虑一条直线将所有点左右两等分，而对于一条直线，需要的信息就只有他的极角。我们将所有点按照在当前直线法线上的投影排序，找一个中位数就能确定直线所在的位置，不妨称所有点在当前直线法线上投影的顺序成为法线序。而有用的极角区间只有不超过n^2个，对于同一个极角区间，n个点的法线序相同（考虑每对点都会将原有的极角区间增加两个，比如对于一对点a, b，在某一半极角区间内a在b之前，另外一半a在b之后）。

考虑按照极角序枚举极角alp，我们需要维护对于alp的法线序A和对于$alp + \pi/2$的法线序B，并且询问同时在A, B前半部分的点的个数是否等于N，由于我们按照极角序枚举极角，因此每次的操作要交换A或B中相邻的两个（因为没有三点共线，因此一定只有两个点交换顺序）。维护A序和B序可以直接用数组维护，询问的话可以对于B序维护一个树状数组，表示对应的点在A序中是否在前半部分。这样我们得到一个$O(n^2 \log n)$的算法，但并不能通过此题。

需要更优秀的算法。上面这个方法主要问题在于需要枚举全部$O(n^2)$级别的极角区间，即使维护做得足够优秀也几乎无法出解，考虑优化这一枚举的部分。注意到每次只会交换相邻的两个这是一个非常厉害的条件。假设我们随机一个极角alp，并且得出A序和B序，不妨令同时在A, B前半部分的点为x个，那么将alp旋转至$alp + \pi/2$，此时B序变成了原来的A序，B序变为原来的A序的翻转，现在同时在A, B前半部分的点数为$2N - x$个。

仔细观察这个将alp旋转至$alp+\pi/2$的过程，可以发现过程中同时在A,B前半部分的点数是一个递增的过程。可以利用仅交换相邻的两个点的顺序证明这个结论。

于是对极角区间二分就可以解决这个问题了。剩下的问题就是对于固定极角求出A,B序判断该往哪边二分，这个排序就行了，其实可以用求第k大的分治算法解决。

至此问题圆满解决。

9.3 时空复杂度

时间复杂度：$O(n\log n)$

空间复杂度：$O(n)$

9.4 备注

得到的$O(n^2\log n)$的算法肯定是不能再清澄上通过了，不过codejam是以提交答案的形式，大数据似乎可以在4,5分钟之内出解。

其实感觉这个单调性并不显然，发现起来还是需要仔细观察，另外这题卡精度，好不容易过掉gcj的数据却发现清澄上的数据怎么卡精度都过不了，只好作罢。

10 2013D Can't Stop

10.1 试题大意

给出一个长度为n的序列，每个位置上有D个数。你可以选出k个数，最大化一段连续的位置使每个位置中的D个数至少有一个出现在你选择的k个数中的一个，使这一段连续的位置最长。

$1\le n\le 10^5, D\le 4, k\le 3$，出现的数不超过$10^5$。

10.2 算法讨论

要注意到$D\le 4, k\le 3$，这种数据范围相当特殊。

考虑一种最暴力的算法，我们枚举答案区间开始的位置，每次往后暴力延伸，如果当前位置的D个数均未被选出，那么枚举一个数选出来，继续下去。这个算法的复杂度大概是$O(n^2D^k)$，关键出在n^2太大了。

如果按照顺序枚举起点，那么终点肯定是单调的，如果我们可以快速判断一段是否可以作为答案，就可以去掉那个n^2了。如果已知我们选了具体哪k个数，可以利用容斥原理计算个数，但是复杂度相当高（大概是$O(nD^{2k}\log^2 n)$），应该考虑其他算法。

可以暴力延伸这一点非常特殊，其实我们并不需要知道具体的起点，只需要知道答案序列中的任意一点就可以进行同样的操作。这样我们便可以使用分治算法，每次从当前段的中点往两端延伸，采取与暴力相同的方式即可，这样子可以将复杂度优化至$O(n\log nD^k)$。

上面的做法已经可以通过本题，但还有更优秀的做法。我们将分治算法稍微改进一下，改成'BFS'形式的，即一层一层按分治区间从大到小的计算。不妨令当前的最优答案长度为len，如果分治区间的大小小于len就停止计算，这样可以去掉一个log，时间复杂度为$O(nD^k)$

10.3 时空复杂度

时间复杂度：$O(nD^k)$
空间复杂度：$O(nD)$

10.4 备注

如果是gcj，那个复杂度相当高的算法应当是可以通过的，因为大数据包里只有6组极限数据，每一个跑10s也不就1分钟。不过是清澄那就不可能过了。

这题确实非常简单但是我思考了相当久的时间，一直觉得分治并没有什么特殊的作用，我错了。

11 2013E Let Me Tell You a Story

11.1 试题大意

给出一个长度为n的序列a，每次可以删去任意一个位置上的数，直到剩下的序列是一个不升序列便停止删除。问有多少种删除方式，对10007取模（两种删除方式不同当且仅当每一次删去的位置不同）。

$1 \le n \le 8000$

11.2 算法讨论

我们直接关注被删除出来的不升序列，那么枚举所有a的不升子序列，把删除得到他们的合法方案数加起来就能得到答案了。

删除方案不合法的情况是如果某次删除结束后a已经成为了不升序列，但仍要进行删除操作。我们来考虑对于不合法的常规处理思路：保证整个过程合法，减去保证合法的方案，减去任意放的方案。

直接保证整个过程合法难度比较高，不太好直接搞。减去已经保证合法的方案这种方法的优势在于不用担心减掉了重复的东西，但这里很难算出有哪些状态需要被减去，也不太方便。不妨考虑减去任意放的方案，显然在这个问题里我们只要枚举一个不在当前序列中的数，并且加入这个序列后序列仍是不升的，我们只需要减掉任意删除得到加入这个数后的方案就行了。但问题是我们不能记录当前的序列里选取了具体哪些数，我们换一种角度思考。对于一个不升子序列，令任意删除得到他的方案数为x，他的长度为len，那么x需要被它本身加上1次，被它的子序列减去len次，对答案的贡献为$x \cdot (len - 1)$。这样计算每个子序列对答案的贡献就行了。

至此问题圆满解决。

11.3 时空复杂度

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n)$

11.4 备注

这题考场上的各位都被50point的分值吓住了吧，其实这题挺简单常规的，和2014的最后一题ARAM完全没得比。

如何统计合法的方案数也是经典问题，应该非常好想。

第三部分　codeforces试题范做

12　263E　Rhombus

12.1　试题大意

给出一个$n \times m$的表格，第i行第j列有一个非负整数$c_{i,j}$，另外你得到了一个非负整数k。

你需要找出一对(a,b)满足：$k \le a \le n-k+1$，$k \le b \le m-k+1$，且在所有满足上述条件的(a,b)中函数$f(a,b)=\sum_{i=1}^{n}\sum_{j=1}^{m} a_{i,j} \times max(0,k-|i-a|-|j-y|)$最大。

$1 \le n,m \le 1000$

12.2　算法讨论

考虑最暴力的算法，枚举所有可能的(a,b)，再暴力枚举$a_{i,j}$的系数，最后利用前缀和统计答案。看起来这是$O(n^3)$肯定会TLE，但稍微分析一下就看发现实际复杂度是$O((n-2k) \times (n-2k) \times k)$，当$n$取到1000时这个值不超过$10^8$，在codeforces机器速度较快的情况下是可以通过本题的。

当然$O(n^2)$算法并不难想，很容易用多个前缀和数组递推得到答案，但实现过程并不简单。这里分享一种只需要两个前缀和数组就能计算出答案的方法。考虑$|i-a|+|j-y|$这一部分其实相当于曼哈顿距离，我们可以利用分类讨论将绝对值符号拆掉，再利用这一特点设计算法。

令$f_{i,j}=\sum_{t=0}^{t<k} c_{i-t,j-t}$, $g_{i,j}=\sum_{t=0}^{t<k} c_{i-t,j+t}$

则$f(a,b)=g_{a+k-1,b}+g_{a-1,b-k}-f_{a+k-1,b-1}-f_{a-1,b+k-1}$

对于$f_{i,j},g_{i,j}$利用前缀和即可快速求出，之后直接枚举$(a,b)O(1)$计算即可，实现非常方便。

12.3　时空复杂度

时间复杂度：$O(n^3) \to O(n^2)$

空间复杂度：$O(n^2)$

12.4 备注

暴力竟然能够，真是无力吐槽了，出题人也不卡一下。这题Div2-E风格明显，思路不难但想要写出精短的代还是很有难度的。

13 293B Distinct Paths

13.1 试题大意

给出一个$n \times m$的网格，有些格子已经染上了k种颜色的某一种。你需要给未染色的格子染上k种颜色中的一种，使得从$(1,1)$到(n,m)的任何路径上都不存在相同的颜色，路径只能沿着有边相邻的格子往右或者往下走。求有多少种满足条件的方案。

输出答案对10^9+7取模。

$1 \leq n,m \leq 1000, 1 \leq k \leq 10$

13.2 算法讨论

很显然当$n+m-1>k$时，一定是没有方案的，因此答案等于0。那么数据范围就缩小了，有$n+m-1 \leq k$这个限制了。

最初始的想法是轮廓线+最小表示，不过这题已经有被染色的格子因此变得相当麻烦。但这启发我们在最小表示下可能的状态不会太多，那么就使用最小表示法+暴力搜索，最后乘上一个排列数再把所有方案数加起来就能得到答案了。

13.3 时空复杂度

时间复杂度：$O(S \times k)$,S为可能的状态数。

空间复杂度：$O(n^2)$

13.4 备注

在轮廓线算法上纠结了相当久QAQ。。。没有办法的情况下就只能爆搜了咯。不怎么有趣的题。

14 235E Number Challenge

14.1 试题大意

令$d(n)$表示n的约数个数。

给出a, b, c，要求计算$\sum_{i\le a, j\le b, k\le c} d(ijk)$

$1 \le a, b, c \le 2000$

14.2 算法讨论

考虑一个数的约数个数可以通过素因子次幂的成绩计算。

不妨利用动态规划算法来计算，令$f_{a,b,c,p}$表示所有素因子都不超过p的情况下，$\sum_{i\le a, j\le b, k\le c} d(ijk)$的值是多少，$p$是一个素数。利用枚举$p$在$a, b, c$中的次幂数来转移，具体的：

$$f_{a,b,c,p} = \sum_{x=0}^{p^x\le a} \sum_{y=0}^{p^y\le b} \sum_{z=0}^{p^z\le c} f_{a/p^x, b/p^y, c/p^z, p-1} \times (x+y+z+1)$$

注意到当$p^2 > max(a, b, c)$时，所有大于p的素数在a, b, c中要么出现一次，要么不出现，也就是没有本质区别，我们可以不用继续dp，直接分类讨论他们在a, b, c中的出现情况，结合容斥原理直接计算答案即可。

另外，a/i也只有$\sqrt{a}$种取值，并且$\lfloor \frac{\lfloor \frac{a}{b} \rfloor}{c} \rfloor = \lfloor \frac{a}{b\times c} \rfloor$，因此状态数只有$O(n^2), n = max(a, b, c)$，再乘上转移复杂度，总时间复杂度是$O(n^2 \log n)$。

还有一种基于一个神奇结论的反演算法，复杂度也是$O(n^2 \log n)$，在这里就不提了。

14.3 时空复杂度

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n^2)$

14.4 备注

不愧是丽洁的题，还是挺有难度的，rng58猜的那个结论也是相当神奇。

15 325E The Red Button

15.1 试题大意

给出一个n个点$2n$条边的有向图，顶点标号为$0,1,...,n-1$，标号为i的点存在连向标号为$2i,2i+1$的边，在这个图上求一个从0出发的哈密尔顿回路，输出这组方案，无解输出-1。

$1 \leq n \leq 10^5$

15.2 算法讨论

又是cf上喜闻乐见的构造题。

首先当n为奇数时，考虑0和$n-1$，他们都只能从$(n-1)/2$过来，因此一定不存在哈密尔顿回路。

接下来我们只有考虑从n为偶数的情况，有一个十分有趣的性质，为了方便下面所说的数都是指这个数模n得到的数。考虑i和$i+n/2$能够到达的点是相同的，也就是这两个点没有本质区别，具体考虑时并不需要知道是哪一个点。那么我们将i和$i+n/2$合并为同一个点，这个点连出两条边，一条连向$2i$和$2i+n/2$合并成的点，表示下一步我们走到了$2i$，另一条连向$2i+1$和$2i+1+n/2$合并成的点，表示下一步我们走到了$2i+1$，我们需要在新的图上求一条欧拉回路。剩下的事情就很简单了。

还有一些细节问题实现的时候注意即可。

15.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

15.4 备注

一道我非常喜欢的构造题。虽然大多数时候构造哈密尔顿路都会转化到构造欧拉路。

16 241B Friends

16.1 试题大意

给出n个数，$a_1, a_2, ..., a_n$，每两个数都有一个异或和$a_i \otimes a_j$，求所有异或和中前k大的和。

$1 \leq n \leq 5 \times 10^4, 0 \leq k \leq n \times (n-1)/2, 0 \leq a_i \leq 10^9$

16.2 算法讨论

k的级别让我们不可能枚举k个异或和，因此需要更好的算法。直接找出前k大的异或和是比较麻烦的事情，不妨考虑问题的反面，如果已经知道了第k大的异或和的结果，如何计算k是多少。这是一个很经典的问题，建立一颗trie就能在$O(n \log n)$的时间内计算出来。既然可以快速计算出这个问题，那么在加上一个二分就能够算出原问题的答案了，需要$O(n \log^2 n)$的复杂度。

继续思考发现其实我们不需要二分，我们将二分改成倍增算法将得到一个复杂度为$O(n \log n)$。

至于统计前k大的和，可以在trie上的每一个节点存下这颗子树里面第i位为1的有多少，最后统计即可。很可惜这一步复杂度是$O(n \log^2 n)$

16.3 时空复杂度

时间复杂度：$O(n \log^2 n)$

空间复杂度：$O(n \log^2 n)$

16.4 备注

嗯我比较喜欢的题目之一，也算一道好题吧。

这题太丧心病狂了吧，时限竟然从6s被改成了200ms，复杂度明明一样怎么卡常数都过不了，最后只好参考ydc的代码把常数卡过。表示感谢> _ <。

17 260E Dividing Kingdom

17.1 试题大意

给出平面上n个点，要求画4条不同的直线，两条垂直于x轴，另两条垂直于y轴，并且这四条线将平面分成9个部分，满足存在一种方案将这9个部分分给9个人，使得第i个人得到a_i个点。画的线当然不能恰好穿过一个点。

$1 \le n \le 10^5, \sum_{i=1}^{9} a_i = n$

17.2 算法讨论

不妨假设分出来的9个区域包含的顶点数从左往右，从下往上依次是：$b_1, b_2, ..., b_9$，他们是$\{a_i\}$的一种排列。

我们只要确定b_1, b_2, b_3，就能确定两条竖直的线和一条靠下的水平的线（先用$b_1 + b_2 + b_3$确定靠下的水平线，再确定竖直的线）。不难发现，再枚举剩下任意一个b_i的大小就能确定所有线了，复杂度大概是$O(\binom{9}{6} \times 6n + n \log n)$。

上面的过程我们发现，在没有别的条件的情况下，只要确定一条线左侧（或下侧）有多少顶点就能确定这条线了。另一种方法也就出现了，不妨枚举所有$\{a_i\}$的排列，已知排列后可以确定直线的坐标，再利用树状数组套单调队列快速查询每一个部分是否满足条件即可，复杂度$O(9! \log^2 n + n \log^2 n)$。

17.3 时空复杂度

时间复杂度：$O(\binom{9}{6} \times 6n + n \log n)$ 或$O(9! \log^2 n + n \log^2 n)$

空间复杂度：$O(n)$ 或$O(n \log n)$

17.4 备注

比较明显的Div2-E风格吧。不过这次还不算难写。

18 317E Princess and Her Shadow

18.1 试题大意

一位公主想要追回她的影子。在一个二维平面内，公主在(v_x, v_y)，她的影子在(s_x, s_y)，并且平面上还有很多树，分别是(tx_i, ty_i)，公主和影子都不能走进有树的格子。公主每次可以往上下左右移动一步，但影子是影子，她会和公主移动同样的一步，除非目标格子是一颗树，此时影子就会在原地不动。要求构造出一个公主追到影子的方案，如果不可能则输出-1。

所有坐标在$[-100, 100]$之间，方案的步数不能超过10^6。但平面是无限的，公主和影子都可以走出坐标的范围。

18.2 算法讨论

如果公主和影子之间不连通，那么毫无疑问就无解了咯。

否则公主一定会去追影子，不妨假设每次公主都会走最短路。如果公主和影子不在无限平面内（即被封起来了），那么公主肯定能够追到影子。追不到的话就说明公主和影子都进入了无限域里面，这样问题也十分好解决。

- 如果公主在影子的正左方，那么就找到最靠左的一颗树，将影子移动到这颗树的左边，让公主往右走即可。
- 如果公主在影子的左上方，先找一颗最靠上的树，将影子移动到这颗树的上边，让公主往下走直到位于影子的正左方，再按上面所说的移动即可。

注意具体实现时不需要走一步求一次，只需要求出到(s_x, s_y)的最短路，当公主到达这里之后再复制影子走过的路径即可。当公主和影子坐标的绝对值都大于100的时候就可以认为他们走到了无限域了。

18.3 时空复杂度

时间复杂度：$O(D^2)$
空间复杂度：$O(D^2)$

18.4 备注

比较有趣的构造题，我代码写得很复杂，应该有更简单的做法。

19 325D Reclamation

19.1 试题大意

给出一个$r \times c$的环形网格（即对于所有的i，$(i,1)$和(i,c)也是相邻的），如果存在一条从顶部到底部路径并且路径上的格子都存在则称之为合法。现在依次删去m个格子，如果删去这个格子之后仍合法，则删掉这个格子，否则不操作。求有多少次删除成功进行。

$1 \leq r,c \leq 3000, 1 \leq m \leq 3 \times 10^5$

19.2 算法讨论

直接判断连通性是比较复杂的事情，不如我们通过顶部和底部是否被割开，也就是是否有一个八连通的连通块形成了一个“环”。比较麻烦的是这是一个环形网格，不过影响不大，只要将整个网格复制一份到右边就行了，这样判断是否成“环”只需要判断是否存在一个(i,j)和$(i,j+c)$是否八连通即可。维护连通性十分简单，每次只会加入两个点，因此对这两个点判断就行了，只需要用并查集即可。

19.3 时空复杂度

时间复杂度：$O(n^2)$
空间复杂度：$O(n^2)$

19.4 备注

不算太难的题，不过一直在纠结复制一份是不是不够。

20 335E Counting Skyscrapers

20.1 试题大意

有n栋摩天大楼修建在一列，n是$[2,314!]$中的一个随机数。每栋楼的高

度也是随机选取的，对于所有的正整数i，有2^i的概率选取高度为i，大楼的楼层标号为$0, 1, ..., i-1$。如果两栋楼的第i层之间没有其他楼房，则有一个走廊连接两栋楼的第i层。Alice和Bob想统计有多少大楼，Alice一栋一栋地数，每经过一栋楼则将计数器+1；Bob则为了追求速度，每次会选取这栋楼最高的一个走廊往后走，如果这个走廊高度为i，则会将计数器增加2^i，但由于恐高，他拒绝通过高度大于h的走廊，他们计数器在第一栋楼的初值均是1。

已知Alice或Bob的计数器为n，求另一个人计数器的期望。

$2 \le n \le 3 \times 10^4, 0 \le h \le 30$

20.2 算法讨论

题目很好玩，这就是一个skiplist嘛。

首先考虑已知Bob的计数器，求Alice的。假设Bob通过了一条高度为i的走廊，那么他会在之后一栋高度不小于i的楼房停止，这种楼房出现的概率是2^{-i+1}，因此期望要通过2^{i-1}栋楼房，此时Bob的计数器会增加2^{i-1}。也就是不论Bob怎么走，通过的大楼数的期望总是等于当前的计数器。那么Alice的计数器的期望也就是大楼数的期望就是n了。

再来考虑已知大楼的个数，求Bob计数器的期望。很容易想到一个$O(n \log^2 n)$的动态规划，令$f(i, j)$表示前i层最后一层高度为j，Bob计数器的期望。转移方程：

$$f(i,j) = \begin{cases} (f(k,l) + 2^{l-1}) \times (1 - 2^{-l+1})^{i-k-1} \times 2^{-j+1} & l \le j \\ (f(k,l) + 2^{j-1}) \times (1 - 2^{-j+1})^{i-k-1} \times 2^{-j+1} & l > j \end{cases}$$

利用经典的优化方法，设两个前缀和数组优化转移，很容易将上面这个方程优化成$O(n \log^2 n)$。

20.3 时空复杂度

时间复杂度：$O(n \log^2 n)$

空间复杂度：$O(n \log n)$

20.4 备注

官方题解里面有一种$O(n \log n)$的做法，挺厉害的。

21 360D Levko and Sets

21.1 试题大意

有两个数组$a_1, a_2, ..., a_n$，$b_1, b_2, ..., b_m$，有n个集合，第i个集合生成方式：

1. 初始存在一个元素：1。
2. 任取集合中元素c，对于所有的$j(1 \le j \le m)$如果$c \times a_i^{b_j} \bmod p$不在集合中，则加入这个数。
3. 重复操作2直到没有新的数可以加入

你需要求出有多少数至少在一个集合中。

$1 \le n \le 10^4, 1 \le m \le 10^5, 1 \le a_i < p, 1 \le b \le 10^9$，p是质数。

21.2 算法讨论

注意到所有的操作都是乘法，不妨利用原根将所有的数转化原根的若干次幂。这样下来就将所有的乘法转化为熟悉的加法，并且是在$\bmod (p-1)$下进行。

不难发现集合的生成过程其实等价于$\vec{c} \cdot \vec{b} \times a_i \bmod (p-1)$能得到多少不同的数（$\vec{c}$是任选的一组系数向量）。根据扩展欧几里得相关知识可以得到其等价于$k \times gcd(b_1, b_2, ..., b_m, p-1) \times a_i \bmod (p-1)$能得到多少数（k是任选的系数）。

不妨令$d = gcd(b_1, b_2, ..., b_m, p-1)$，那么集合可以写成$\{s_j | s_j = k \times d \times a_i \bmod (p-1)\}$，这相当于$(d \cdot a_i) \times x - (p-1) \times y = w$所有有解的$w$构成的集合，我们知道只要满足$w = k \times gcd(d \cdot a_i, p-1)$那么就有解，也就是说统计有多少$w \in [0, p-1)$且$w$是某个$gcd(d \cdot a_i, p-1)$的倍数。

接下来的问题利用容斥原理解决即可。虽然约数个数是$\sqrt{p}$级别，但在10^9只能约数个数最多的数仅有10^3左右个。

21.3 时空复杂度

时间复杂度：O(约数个数^2)

空间复杂度：$O(\sqrt{p})$

21.4 备注

中国人round的数论题，不是很难。

22 339E Three Swaps

22.1 试题大意

初始有一个n个数的排列，现在进行不超过3次区间翻转操作，给出操作之后的序列，求操作过程。

$1 \leq n \leq 1000$，数据保证一定有解。

22.2 算法讨论

数据保证有解是很好的条件，但3次区间旋转情况还是太复杂，直接构造难度比较大。

我们可以直观的感受到，3次区间翻转将整个序列分成不超过7段，我们很有可能通过搜索构造出方案。当然直接找出具体是哪7段是比较困难的，可以考虑搜索加上一定的剪枝，就可以通过本题。很容易看出一个满足条件的寻找边界的方法即判断如果$|a_i - a_j| = 1$，那么i, j均有可能成为边界。这个剪枝的优化效果相当好，满足这个的方案已经不多了。

22.3 时空复杂度

时间复杂度：$O(?)$ 视剪枝强弱而定

空间复杂度：$O(n)$

22.4 备注

不怎么好玩的搜索题，不过确实有一些有趣的性质在里面吧，不知道有没有简单的构造方法。

23 286E Ladies' Shop

23.1 试题大意

有n个袋子，袋子的体积是a_i，希望你找到k个物品，每个物品的体积

是p_i，满足：

1. 对于任何一个袋子，存在一种放物品的方案使得袋子被装满（即所有物品的体积和等于a_i），一个物品可以使用多次。

2. 对于任意一个体积和不超过m的物品集合，存在一个袋子能够恰好装满这个集合。一个物品可以在集合中出现多次。

要求求出一组物品的方案$\{p_i\}$，使得k尽可能小。如果无解，输出'NO'。

$1 \leq n, m \leq 10^6$
$1 \leq a_1 < a_2 < ... < a_n \leq m, 1 \leq p_1 < p_2 < ... < p_k$

23.2 算法讨论

根据第二个条件，可以得出在不超过m的情况下，给出的袋子体积$\{a_i\}$就是所有物品有可能组成的体积，不妨考虑最小的那个袋子，毫无疑问必须有一个物品恰好那么大。其他袋子的情况类似，如果它不能由两个袋子合起来得到（两个就够了），那么也要有一个对应的物品。不难发现这就是一个多项式乘法的过程，我们用FFT优化这个过程就行了。

23.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n)$

23.4 备注

我的实数转换竟然比策爷的慢10倍。。。吓傻了。。。只好写数论转换才能过。。。

24 332D Theft of Blueprints

24.1 试题大意

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，恰好有一个点与S每一个点都有边。令这个点为$v(S)$，并且对S进行

操作的代价是S中每个点与$v(S)$的边权之和。现在求对于一个大小为k的子集操作代价的期望。

$1 \leq k < n \leq 2000$

24.2 算法讨论

这是一道比较有趣的简单题。

毫无疑问题目给出的无向图的特殊性质是解决本题的关键。如果直接枚举一个集合，那么复杂度是$\binom{n}{k}$级别的，无法接受。不妨考虑换一个枚举角度，先枚举$v(S)$，此时题目中的性质就可以用上了：任选直接与$v(S)$相连的k个点一定是一个可行的集合S。而题目保证S与$v(S)$是一一对应的关系，因此在枚举了所有可能的$v(S)$也就是枚举所有点之后，一定可以枚举到所有大小为k的子集。

考虑每条边对答案的贡献，即这条边的代价与被选中的概率的乘积，再将所有边对答案的贡献加起来即得到答案。令枚举的$v(S)$为节点x，其度数为$deg(x)$，对于从x连出的一条边$< x, y >$，令其边权为$c(x, y)$，不难发现包含边$< x, y >$的方案有$\binom{deg(x)-1}{k-1}$，而总方案数为$\binom{n}{k}$，因此其对答案的贡献为

$$\binom{deg(x)-1}{k-1} \Big/ \binom{n}{k} \times c(x, y)$$

注：$< x, y >$和$< y, x >$是不同的边。

那么依次枚举所有的边将贡献加起来就能得到答案，这种做法复杂度是$O(n^2)$的，不过缺点是计算时会产生很大的精度误差，自行调整精度后也是可以通过此题的。

但上述做法其实并没有完全运用到题目的特殊条件，这个条件还有更强的性质。

引理一 对于任意一对节点(x, y)，和x, y都有边相连的节点一定恰好有$k-1$个。

考虑任意一对不同的节点(x, y)，令$v_1, v_2, ..., v_l$为同时与x, y相邻的所有节点。如果$l \geq k$，那么对于集合$\{v_1, v_2, ..., v_k\}$已经存在两个点(x, y)与所有的点都相邻，矛盾。那么现在$l \leq k-1$，假设$l \leq k-2$，令集合$S = \{x, y, v_1, v_2, ..., v_l\}$，显然$|S| \leq k$。若$S$的元素个数不及$k$个，不妨在不属于$S$的节点中任选几个将$S$补充至$k$个元素，根据题目提供的条件此时一定

存在一个$v(S)$使得$v(S)$与S中的所有点均相连，即与x,y均相连，但$v(S)$又不属于集合S，即未出现在$v_1,v_2,...,v_l$中，矛盾。因此$l=k-1$。

引理二 该图存在一个大小为$k+1$的子图是完全图。

考虑一个集合$T=\{v_1,v_2,...,v_k,v_{k+1}\}$，其中$v_1,v_2,...,v_k$是任意选取的$k$个不同的点，$v_{k+1}$是与他们都有边的点。依次替换$v_1,v_2,...,v_{k-1}$，假设当前在对$v_i$进行替换，另$S=\{v_1,...,v_{i-1},v_{i+1},...,v_{k+1}\}$，如果$v_i$与当前$S$内所有顶点均相邻，那么$v_i$已经满足要求，否则就将$v_i$替换为$v(S)$即可，接着继续对$v_{i+1}$进行操作。当替换完最后一个元素时，集合$T$构成了一个大小为$k+1$的完全图。

定理 k的值只可能等于1，2，$n-1$。

当$k=1$，该图是一个完备匹配。

当$k=2$，该图是有一个$n-1$个点构成的完备匹配加上一个与剩下所有点都有边的图。

当$k=n-1$，该图是一个完全图。

对于$3\le k\le n-2$，根据引理二，不妨令$S=\{v_1,v_2,...,v_{k+1}\}$为该图的完全子图。考虑$S$中的任意两个节点$x,y$，那么$x,y$和$S$中余下的$k-1$个点都相邻，根据引理一，不会有其他的点与$x,y$同时相邻了，即同时与$x,y$相邻的点均在S 中。任取一个不在$S$集合中的节点$u$，一定存在一个与$u,x,y$都相邻的节点$w$（不一定唯一，可以考虑将集合$\{u,x,y\}$补充至一个大小为$k$的集合），并且$w$一定属于$S$，同时也存在一个$u,x,w$都相邻的点$p$。那么对于$w,p$这两个点，$u$和他们都相邻但$u$不属于$S$，矛盾。因此不存在$3\le k\le n-2$的情况。

根据定理，具体实现时只需要对$k=1$，$k=2$，$k=n-1$三种情况特判即可。复杂度是与输入同阶的$O(n^2)$（此题输入的边数达到了n^2级别）。因为只需要做一步整除计算因此没有精度误差。

至此，问题便圆满解决。

24.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

24.4 备注

来自自己写的详细题解。

25 335D Rectangles and Square

25.1 试题大意

给你n个顶点在整点，边平行于坐标轴的矩形，矩形之间不会相交，可能相切。求是否存在一个子集，使得子集里的矩形形成一个完整的正方形（没有空隙）。

$1 \leq n \leq 10^5, 1 \leq$坐标范围$k \leq 3000$

25.2 算法讨论

考虑正方形左下角的位置，固定这个位置之后，如果确定了正方形的边长，需要上方，右方，对角线上三个点同时存在，如果我们每次选最少的一个方向暴力扫，可以证明时间复杂度均摊下来是$O(n\sqrt{n})$级别的，可以通过此题。

当然还有更优美的做法。可以预处理每个顶点往左右上下最多能够延伸多远，之后将顶点按$x-y$排序，这样我们能够将同一对角线上的点放在一起考虑，同时还可以快速判断出是否可以形成一个正方形的轮廓。有一个很重要的性质：如果一个已经形成轮廓正方形完全包含另一个正方形，那么我们只需要考虑那个被包含的正方形是否被完全填满即可。那么我们便可以用一个单调栈维护可以与当前右上角形成正方形的左下角，并且考虑最近的一个就行了，至于是否被填满，因为已经确定轮廓，只要里面被矩形包含的面积等于当前正方形面积即可，可以用扫描线和线段树统计，这样是$O(n \log n)$的，当然可以使用前缀和，这样是$O(k^2)$的。

25.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

25.4 备注

确实是一道很好的题目，cf上的好题也只有这些非常规round里才比较多吧。根号的算法都想了很久。

26 261D Maxim and Increasing Subsequence

26.1 试题大意

给出一个数列$\{b_n\}$，将他复制t遍之后得到$\{a_n\}$，求$\{a_n\}$的最长上升子序列的长度。

$1 \leq n, maxb \leq 10^5, 1 \leq t \leq 10^9, n \times maxb \leq 2 \cdot 10^7, 1 \leq b_i \leq maxb$

26.2 算法讨论

$n \times maxb \leq 2 \cdot 10^7$这么特殊的条件一看就要用吧。考虑上升子序列长度最长也就$maxb$，复制t遍其实是没有必要的，复制$maxb$遍就可以了。令$N = |a| = n \times maxb$，那么$O(N)$的时间复杂度也是可以接受的了。

既然是上升子序列长度不妨结合经典做法，一是直接用树状数组维护关键字，另一种做法是维护单调栈并且进行二分，其实我们把这两种做法的log都改成暴力单调扫一遍就行了，均摊下来复杂度仍是$O(n \times maxb)$。

26.3 时空复杂度

时间复杂度：$O(n \times maxb)$

空间复杂度：$O(n)$

26.4 备注

竟然有$n \times maxb \leq 2 \cdot 10^7$这种奇怪的条件。当时做的时候没看到直接导致想了一下午没想出来。

27 235C Cyclical Quest

27.1 试题大意

给出一个字符串s，n组询问，每次询问有多少子串与x_i循环同构。
$1 \le n \le 10^5, 1 \le |s| \le 10^6, 1 \le \sum |x_i| \le 10^6$，字符均为小写字母。

27.2 算法讨论

嗯字符串题，并且是统计子串类型的，很容易想到利用后缀自动机解决。

不妨考虑回答一组询问，因为我们可以$O(|x_i|)$地回答每组询问，自然会想的找出所有与x_i循环同构的字符串并依依询问出现次数，其实我们可以一边询问一边找出循环同构的字符串，考虑下一个循环同构的字符串可以看成在左边删掉一个字符，再在右边加上它，利用SAM的parent tree和trans link可以很容易的从上一个串在state转移到下一个串的state，直接询问state→num统计即可。

27.3 时空复杂度

时间复杂度：$O(n \cdot |\sum|)$ $|\sum|$是字符集大小。
空间复杂度：$O(n \cdot |\sum|)$

27.4 备注

丽洁的SAM题。可以用SA做吧，不过稍微慢一点。

28 311E Biologist

28.1 试题大意

给出n个变量，每个可以取值0/1，第i个变量的初值为x_i，改变他的值需要花费v_i，有m组要求，每个要求需要某个子集的变量全为0或全为1，如果满足该要求可以收益w_i，否则不但没有收益，还有花费g_i。请你最大化收益。

28.2 算法讨论

变量取值是0/1这么明显的最小割模型肯定就是用最小割建模。考虑事件并不是直接的二元关系，不妨给每个事件建一个节点单独考虑。而事件和变量的关系是只要有一个变量不满足条件事件就不会发生，符号最小割建模的要求。于是利用最小割就可解决此题。

28.3 时空复杂度

时间复杂度：$O(maxflow(n+m, n+m))$
空间复杂度：$O(maxflow(n+m, n+m))$

28.4 备注

一道最小割模型的题，还不算难吧。这种抽象事件和最大权匹配结合的模型让我建了好一会。

29 306D Polygon

29.1 试题大意

要求构造一个n个点的多边形，各内角相等，各边长不相等。

$3 \leq n \leq 100$，边长在$[1, 1000]$内可以是实数，点的坐标绝对值不能超过10^9。

29.2 算法讨论

嗯条件比较宽松的构造题。

应该不论怎么构造都可以吧，如果坐标可以是实数的话。为了避免精度问题不妨将边长设为$1, 2, ..., n-1$，最后一条边直接连上就行了。但如果依次是$1-n$最后一条边的边长肯定超过了1000，可以将边长设计成这样：$1, n-1, 2, ...$，但设计成这样之后还是会出问题，可以调整成$2, 2n, 3, 2n-1, ...$之类的。稍作调整就可以通过了。

29.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

29.4 备注

喜闻乐见的构造题，不过还算能想吧。总而言之就是乱搞搞就过了。

30 249E Endless Matrix

30.1 试题大意

给你一个无限的表格，前6行6列大概长这样：

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

再给出n组询问，每组询问一个子矩形的和对10^9取模的结果。

30.2 算法讨论

询问子矩形的和很容易想到转化成询问$(1,1)$到(x,y)的和，在利用区间加减法求得答案即可。这样转化之后公式就很好推了，不过麻烦的是比如说要用到除法，以及运算结果会超过64bit的范围。用到除法的话可以转化为对$6\cdot 10^9$取模，这样就能直接除了。另外一个问题可以用$7k+$的乘法解决，也能用快速乘代替。

30.3 时空复杂度

时间复杂度：$O(n)$
空间复杂度：$O(n)$

30.4 备注

竟然有这么简单的Div1E，难以置信。可能是中国人熟知的小技巧木有普及。

31 295D Greg and Caves

31.1 试题大意

给你一个$n \times m$的网格，每个格子可以染黑色或白色，统计满足图案是一个洞的方案数。具体的：

1. 存在一个l, r，$[l, r]$之间的行都恰好有2个黑格子。其他行没有黑格子。
2. 存在一个$t, l \leq t \leq r$，使得对于所有的$l \leq i \leq j \leq t$，满足i行黑格子之间的列是j行的子集，相似的，对于所有的$t \leq i \leq j \leq r$，满足j行黑格子之间的列是i行的子集。

31.2 算法讨论

先枚举t行两个黑格子之间列的宽度，观察这个洞边缘要求的特殊性质，可以发现等价于一条路径从t行出发，走到l行之后又走回来，并且至少在l行经过两个格子。我们可以做一个翻折，将走回来的那部分按l行对称过去，至少在l行经过两个格子的话我们简单的将列数减1即可，不难发现就相当于从左上角走到右下角的方案数，用组合数统计即可。

接下来就用一个前缀和数组统计答案。比较头疼的是去重，因为如果有多个宽度和t行相同的行我们会统计多遍。不妨只统计第一个t行所在的位置，直接统计这个稍微有点复杂，不过仔细处理一下即可。也可以用补集转化，如果正着做很麻烦那么就减掉重复的，这样会简单很多。

31.3 时空复杂度

时间复杂度：$O(n^2)$
空间复杂度：$O(n^2)$

31.4 备注

比较有趣的计数题吧。应该只能算是简单计数。我耗时太久了，没有直接想到补集转化还多设了一个DP。

32 264E Roadside Trees

32.1 试题大意

在一条街上从西往东共有n个位置来种树。树每个月长1m，在每个月的开始你需要进行一个询问，这个询问是下面一些的一种：

1. 在位置p种一棵高度为h的树。
2. 砍断当前存在的从西往东第x棵树，砍倒的树会占据所在的位置，以后无法在上面种树了。

当你执行完每个询问之后，需要回答最长上升子序列的长度。
保证任何时刻不会有两棵树高度相同。
$1 \leq n \leq 10^5, 1 \leq h, x \leq 10$

32.2 算法讨论

每个月树的高度会增加一，看上去需要类似区间加的东西，实际上没有必要，我们只需要将所有树的高度统一到第一个月就行了。

直接维护支持修改的最长上升子序列长度是一件很麻烦的事情，似乎不太靠谱。不过这题数据范围十分特殊，x, h都在10以内，看起来就像是暴力题。同样的不妨采用类比的方法，考虑我们如何优化n^2的最长上升子序列，维护一个类似的栈。我们倒过来做最长下降子序列，考虑x不超过10，也就是说删掉第x棵后只有最西边x个有影响，考虑h不超过10，不难发现每当其加入时是当前前10小的，也就是说只有高度小于他的10个有影响。但这样我们要快速维护每个位置的栈的信息并且支持更新，考虑给每个栈的位置都维护一个set，把这个位置的更新情况都存下来，这样我们在set上二分就能将对应的栈找到。而每次操作受影响的位置都不超过10个，暴力即可。

32.3 时空复杂度

时间复杂度：$O(n \log n^2 + 10n \log n)$
空间复杂度：$O(n)$

32.4 备注

嗯很有趣的题，总之化归类比是好算法。开上n个set真是暴力极了，看见status里面不少线段树做法。

33 286D Tourists

33.1 试题大意

有两条旅游路线，以$(1,0)$和$(-1,0)$为起点，沿着y轴正方向前进，同时在从某些时刻起在y轴正半轴上会出现n个墙，第i个墙从t_i时刻起出现，范围从$(0,l_i)$到$(0,r_i)$，如果两个人的连线穿过一堵墙，那么他们就互相看不见了。有m对旅人，第i对会从q_i时刻开始，一个从$(1,0)$出发，另一个从$(-1,0)$出发，沿着旅游路线行走，速度均为$1m/s$，问他们互相看不见的时间有多长。

$1 \leq n,m \leq 10^5$

33.2 算法讨论

脑补一下就会发现，单独一段墙的出现对一对旅人的影响是，给一段时间出发的旅人互相看不见的时间增加一个和时间相关的一次函数，给接下来大于某个时刻的旅人增加墙的长度。看上去这要写一个线段树，其实我们利用差分就可以解决这个问题。

墙重叠的问题就更简单了，我们将点离散化之后每次一段一段地取就行了。

接下来直接统计答案就行了。

33.3 时空复杂度

时间复杂度：$O(n\log n)$

空间复杂度：$O(n)$

33.4 备注

这题好歹是D啊，未免也太不够意思了吧，竟然是差分加排序就够了的题。

34 258D Little Elephant and Broken Sorting

34.1 试题大意

给出一个长度为n的排列，现在有m次操作，每次操作会交换两个元素，但都只有1/2的可能性发生。问最终排列里逆序对的期望个数。

$1 \leq n, m \leq 1000$

34.2 算法讨论

这题我做得不好。虽然把期望拆开了但一直关注某两个数最后在哪，但其实上需要统计的东西是一个条件概率，总之就是不work。不难发现，其实某个位置上的数具体是多少其实我们并不care，因为我们可以n^2的维护各个位置上的大小关系，那也就可以维护$f_{i,j}$表示第i个位置上的数大于第j个位置上的数的概率，这样我们找到了一种靠谱的拆分期望的办法，我们只需对每个操作暴力$O(n)$的维护这两个位置与其余位置的关系即可。最后将期望线性叠加就行了。

34.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

34.4 备注

唉这题其实并不难，不知道怎么自己就想了好久，一直关注与具体数字上，其实关注位置就行了。

35 273D Dima and Figure

35.1 试题大意

给出一张$n \times m$的表格，每个格子可以染黑，要求所有染黑的格子四连通并且对于任意两个黑格子$(x_i, y_i), (x_j, y_j)$，最短距离（每次可以走到相邻的黑格子）为$|x_i - x_j| + |y_i - y_j|$。问染黑的方案数对$10^9 + 7$取模。

$1 \leq n, m \leq 150$

35.2 算法讨论

思考一下不难发现要求的黑色图案其实是一个“凸包”，具体的就是对于左右边界均是先范围扩大再范围减小，那么直接设一个DP，$f_{i,j,k,0/1,0/1}$表示第i行为黑色的范围是j,k，左右边界的范围是增大还是缩小，对应的方案数是多少。不难发现转移其实是一个矩形区间，可以利用前缀和优化到$O(1)$，注意去重即可。

35.3 时空复杂度

时间复杂度：$O(n^3)$
空间复杂度：$O(n^3)$

35.4 备注

简单DP题，这种题我也没有一眼，真是不知道该说什么。

36 283E Cow Tennis Tournament

36.1 试题大意

有n个人，每个人拥有一个唯一的能力值k_i，两两都会进行一场比赛，两个人进行比赛一定是能力值大的一方获胜。现在举办方觉得比赛太无聊了，决定做m次操作，每次操作会取反所有双方能力值均在$[a_i, b_i]$内的比赛结果。问最终由多少(p, q, r)，使得p战胜q，q战胜r，r战胜p。

$1 \leq n, m \leq 10^5$

36.2 算法讨论

这种三元关系很难直接统计，不妨考虑补集转化。不难发现所有非法三角形都有一个人同时赢了剩下的人和同时输了剩下的人，我们只需单独对每个人统计同时赢了多少人和输了多少人即可，这就相当于统计度数。我们不妨维护一个线段树，在依次扫描所有人，将所有与其相关的区间都进行修改，最后利用补集转化统计答案就行了。

36.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n)$

36.4 备注

作为一道Div1-E还是简单了点吧。。。三角形统计上补集转化是常用思路。

37 280E Sequence Transformation

37.1 试题大意

给出一个不降序列$x_1, x_2, ..., x_n$，均在$[1, q]$之间。另外给出a, b，要求构造一个序列$y_1, y_2, ..., y_n$满足$a \leq y_{i+1} - y_i \leq b$，并且最小化$\sum (x_i - y_i)^2$。

CF原范围：$1 \leq n \leq 6000$
清澄范围：$1 \leq n \leq 3 \times 10^5$

37.2 算法讨论

直接考虑令$dp_{i,j}$表示y_i取值为j的最小代价，当然这样是不靠谱的，因为后一维状态是无限的，那么考虑用一个函数来表示这个DP，之后这就work了。那么我们每次需要做的就是先找到最小值，再把最小值左边的图象平移$b - a$，之后再给整体加上一个二次函数即可。不难发现这就是很多分段函数拼凑在一起，每一段都是一个二次函数。考虑凸函数与凸函数的和仍是凸函数就不会有其他问题了。

直接暴力维护每段函数就能过通过CF数据了，不过出题人调皮了一下，非把数据范围弄到了$30W$的级别，感觉其实没什么必要了，不过这样就得用平衡树维护了。比较方便的是直接维护导数，记录下每个位置的最优区间，答案最后统计就行了。比较麻烦的地方是几个tag的维护要注意思考一下。把思路理清仔细思考一下应该不会有问题。

37.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

37.4 备注

出题人还是有点丧心病狂的。。。不能多说。。。我不仅会写平衡树而且会卡常数么，但更逗的是我竟然把在平衡树上二分弄成$O(\log^2 n)$的了，弄到一直不敢写怕常数卡不过。。。

38 305D Olya and Graph

38.1 试题大意

给出一张有向无环图，要求在上面添上一些边，使得：

1. 如果$(x,y), x<y$有边，一定是从x连向y。
2. 从任意一个点i，一定能到$i+1, i+2, ..., n$。
3. 任意两个点之间只有一条边。
4. 对于所有$i<j, j-i\leq k$，i,j间的最短路等于$j-i$。
5. 对于所有$i<j, j-i>k$，i,j间的最短路等于$j-i$或者$j-i-k$。

求删边方案数对10^9+7取模。
$1\leq k\leq n\leq 10^5$

38.2 算法讨论

题目的限制相当强（当然了这是道Div2D），所以统计起来也相当方便。首先所有i到$i+1$的边肯定都是要存在的，那么就分是否存在跨度为k的边分类讨论就行了。对于每种情况可以先决定一条最靠前的跨度为k的边，再用2^{cnt}次方统计答案即可。

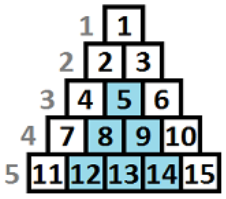
38.3 时空复杂度

时间复杂度：$O(n)$
空间复杂度：$O(n)$

39 354D Transferring Pyramid

39.1 试题大意

给出下面一种三角形：



每次你可以选中一个格子，代价是3，也可以选中一个子三角形（图中蓝色格子就是选中5的子三角形），代价是子三角形的大小+2，给出一个高度为n的三角形，其中有m个点被标记，要求你用上面两种操作选中所有被标记的点，使得代价最小（一个点当然可以被选多次）。

$1 \leq n, m \leq 10^5$

39.2 算法讨论

如果我们直接选择单独选中所有点，那么代价是$3m$，也就是说我们答案的下界就是$3m$了。考虑选子三角形的操作，如果这个三角形大小超过$3m$就肯定没有必要选他了，而如果他的高度为k，他的大小为$k \times (k-1)/2$，那么我们所选的子三角形高度就不超过$\sqrt{6m}$了。也就是可以$O(n\sqrt{n})$暴力DP了。

令f_i表示第i斜列之前的点被选中了的最小代价，那么枚举当前选中了多大的子三角形，为了优化转移可以统计一个数组g_i表示第i反斜列（注意和f数组的方向是反的，相当于一个斜率是1另一个是-1）都被选中的最小代价，这样就可以转移了。对于高度超过$\sqrt{n}$的点我们直接选中即可。

39.3 时空复杂度

时间复杂度：$O(n\sqrt{n})$

空间复杂度：$O(n)$

39.4 备注

还是挺有趣的题，本来还没什么人过的，即使发现了根号的性质也是要思考一下的。

40 261E Maxim and Calculator

40.1 试题大意

现在有两个数a,b，有两种操作可以选择，一是令$b=b+1$，另一个是令$a=a\cdot b$，初始时$a=1,b=0$，问进行不超过p步操作之后，$[l,r]$间有多少数能够出现在a中。

$1\le l\le r\le 10^9, 1\le p\le 100$

40.2 算法讨论

这是个奇怪的题目，$p\le 100$就表示你不能生产出含有100以上质因子的数，脑补一下就会发现其实这样的数还是相当少的，动手写个暴力跑出来大概只有3×10^6级别，那么就可以考虑复杂度和答案有关的算法了。

考虑采用类似最短路的算法，我们只需要知道那些能在$\le p$步内到达的点有哪些就行了。具体的我们可以用一个$O(answer\cdot p)$的DP，维护一个双向链表再加上一个队列转移即可。

40.3 时空复杂度

时间复杂度：$O(answer\cdot p)$

空间复杂度：$O(answer)$

40.4 备注

真是无语。竟然有这种题，不过也还算好吧，虽然做法奇葩。。。关键是我一直以为需要$O(answer)$的时间复杂度才能过，然后逗了一段时间。

41 253E Printer

41.1 试题大意

给出n个任务，每个任务有优先级p，起始时间s，以及持续时间t，每次会选择优先级最高的任务进行，现在有一个任务的优先级不明，但知道他的结束时间，求他可能的一个优先级以及所有任务的结束时间。

$1 \leq n \leq 5 \cdot 10^4$

41.2 算法讨论

考虑知道所有优先级模拟整个过程可以借助一个优先队列就能暴力了，复杂度是$O(n \log n)$的，显而易见地，优先级越高完成的时间越快，借助这个单调性就可以二分答案了。

41.3 时空复杂度

时间复杂度：$O(n \log^2 n)$

空间复杂度：$O(n)$

41.4 备注

其实此题有$O(n \log n)$算法，也不是很难想。

42 309E Sheep

42.1 试题大意

给你n个区间，第i个区间范围是$[l_i, r_i]$，要求你给区间重新排列顺序，使得有交集的区间中，距离最远的距离尽可能的小。

$1 \leq n \leq 2000$

42.2 算法讨论

考虑最大值最小那么就二分答案，于是我们得到了一个限制条件：所有有交集的区间距离不超过D。

我们记l_i为初始第i个区间能放置的位置的最大值，这是由已经放置的区间和D决定的。

考虑第一个放的区间一定是右端点最小的区间，因为如果他放在后面的话，不妨将其提到第一个来，这样不会有约束条件减弱。但如果每次都选当前右端点最小的放置是会有bug的，因为它的限制还和与之相交的区间有关。也就是说我们会遇到某个区间必须放在当前的位置，但右端点不是最小的。考虑解决这个困境，如果我们能在保证之后的区间一定有合法的放置位置的条件下贪心地放置区间这应该就靠谱了。这里我们借助Hall定理，如果前i个位置恰好有i个区间需要放置，那么可以在这i个区间中选择一个右端点最小的。这样就保证了之后也一定存在合法的放置方案，同时约数条件尽可能的宽松。

至此本题便圆满解决。

42.3 时空复杂度

时间复杂度：$O(n^2)$
空间复杂度：$O(n^2)$

42.4 备注

这题还是有难度的，想了相当久才想出来，二分答案之后那个结合hall定理的贪心给人眼前一亮的感觉。

43 316E3 Summer Homework

43.1 试题大意

给出一个序列$\{a_n\}$，有下面三种操作：

1. 将a_x赋值为v。
2. 询问$\sum_{x=0}^{r-l}(f_x \cdot a_{l+x})$，$f_i$是斐波拉契序列第$i$项。
3. 将$a_l, ..., a_r$均加上d。

43.2 算法讨论

具体来说，本题就是一个单点修改，区间加，区间查的fib数有关的问题，这样经典的问题不难想到用线段树维护fib数的转移矩阵，只需要令斐波拉契数$f_0 = f_1 = a$推得的f_n便是$fib_n * a$，这样就支持单点改和区间查了，区间加只需加上一个tag 就能解决，可以预处理一些东西方便计算。

43.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n)$

43.4 备注

很经典的问题，利用线段树维护转移矩阵，不知道我现在来做是不是可能会用笨办法。

44 243C Colorado Potato Beetle

44.1 试题大意

你在一个无限的网格图上，给出你的行走过程：你走了n步，每次会向上下左右某一个方向走d_i步，你会在你经过的地方撒上农药。从无限远的地方会有害虫侵袭，如果一个格子被感染，那么与之有公共边的并且未喷洒农药的格子也会感染。问最终未被感染的格子有多少。

$1 \leq n \leq 1000, 1 \leq d_i \leq 10^6$

44.2 算法讨论

这不裸的BFS么，离散化一下然后暴力。除了写起来麻烦一点没有其他问题了吧。

44.3 时空复杂度

时间复杂度：$O(n^2)$
空间复杂度：$O(n^2)$

45 316G3 Good Substrings

45.1 试题大意

统计字符串s中有多少本质不同的子串是好的。一个子串是好的定义为满足所有n个规则，其中每个规则定义为一个三元组(p, l, r)，即p是一个字符串，如果某个字符串在p中出现的次数在$l..r$之间，那么就满足这个规则。

$0 \leq n \leq 10$, $maxlength \leq 50000$

45.2 算法讨论

所有的限制条件都是串的出现次数，这几乎就是让我们用SAM来做这题。我们不妨将s看做一个规则，即所选出的串需要在s中出现$1..len(s)$次，统计有多少满足所有规则的字符串。我们将所有规则包含的字符串建成一个SAM，统计在限制条件中每个状态的出现次数就行了。

45.3 时空复杂度

时间复杂度：$O(n \cdot |\sum|)$
空间复杂度：$O(n \cdot |\sum|)$

45.4 备注

比较有趣的字符串题，不过做起来还算简单。出题人竟然卡空间，这些人真是什么心态？

46 251E Tree and Table

46.1 试题大意

给出一颗$2 \cdot n$的树，要求将其放置在一个$2 \cdot n$的棋盘上，使得棋盘上的格子与树的节点一一对应，并且如果树上两个节点有边那么对应的格子也相邻。求方案数对$10^9 + 7$取模的结果。

$1 \leq n \leq 10^5$

46.2 算法讨论

很显然如果有点度大于3，那么就无解了。

考虑棋盘左端点的情况，要么是两个度为1要么是一条边相连的两点，再考虑从左往右地放置，不难发现$2 \cdot n$的表格有很好的性质，那就是整体来说“一条链”，那么可以设$f_{(x,y)}$表示把(x,y)这条边靠y这一侧的点全部放入棋盘，合法的方案数是多少。然后按情况讨论转移就行了。

46.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

46.4 备注

这题细节相当多而且麻烦，需要注意很多情况，WA了相当多次也是多么痛的领悟。

47 256D Liars and Serge

47.1 试题大意

有n个人，他们一部分说真话，另一部分会说假话。现在你想知道他们中有多少人说真话，于是你询问了所有人这个问题。他们每个人都知道其余的人谁说真话，谁说假话，于是说真话的人会告诉你说真话的人的确切个数，而说假话的会在$1..n$中任意回答一个错误答案。问有多少种$a_1, a_2, ..., a_n$，a_i为第i个人的回答，使得可以判断出恰好有k个人明显说谎，输出答案对777777777取模之后的结果。

$1 \leq k \leq n \leq 2^8$，保证n是2的次幂。

47.2 算法讨论

如果满足$a_i = b$的i恰好有b个，那么可以推出这b个人可能在说真话，那么很容易写出一个n^4的Dp，用$f_{i,j,k}$表示当前DP到所有人的回答不超过i个，并且之前确定的人数为j，其中有k个人说谎，枚举回答i的人数，利用组合

数很容易推出转移方程。不过问题是会TLE，注意到还有一个最重要的条件我们没用，保证n是2的次幂，那么TLE 的部分我们打表就行了。

47.3 时空复杂度

时间复杂度：$O(n^4)$
空间复杂度：$O(n^4)$

47.4 备注

标解竟然就是打表，无力吐槽。。。

48 303D Rotatable Number

48.1 试题大意

定义Rotatable Number，设这个数长度为n，如果他旋转之后能得到的数也可以通过乘以$1, 2, ..., n$得到，那么称这个数为一个Rotatable Number。希望你找到一个最大的b满足$1 < b < x$，使得b进制下存在一个长度为n的正Rotatable Number。

$1 \leq n \leq 5 \cdot 10^6, 2 \leq x \leq 10^9$

48.2 算法讨论

这题有点类似数学结论题，总之要多玩玩。考虑题目中给出的142857，这个10进制下6位的Rotatable Number，他乘以1到6均会得到他的旋转之后得到的数，那如果乘以7，那么有趣的事情发生了：

$$142857 \times 7 = 999999 = 10^6 - 1$$

于是我们可以猜一个结论，如果$b^n - 1$能被$(n + 1)$整除，那么$(b^n - 1)/(n + 1)$就是一个Rotatable Number，可惜这个结论并不正确，可以发现如果存在$1 \leq x < n$满足$(b^x - 1)/(n + 1)$，就不会存在Rotatable Number。到此问题已经非常明显，只需找出$n + 1$的一个原根就行了。

具体证明可以考虑类似循环小数。

48.3 时空复杂度

时间复杂度：$O(? \cdot \log n + \sqrt{n})$，视原根疏密而定。

空间复杂度：$O(\log n)$

48.4 备注

比较有意思的题目，挺好玩的，可能是我孤陋寡闻之前不知道这个性质。

49 338D GCD Table

49.1 试题大意

有一个n行m列的表格G，$G_{i,j} = gcd(i,j)$，给出一个正整数序列$a_1, a_2, ..., a_k$，问这个序列是否在表格某行的连续位置出现。

$1 \leq n, m, a_i \leq 10^{12}$

49.2 算法讨论

首先可以发现肯定出现在$t = lcm(a_1, a_2, ..., a_k)$行，因为出现的行的编号肯定是$t$的倍数，而如果大于$t$只会增加限制条件，没有必要，所以直接令行号等于$t$。

很自然的，会想到是否能将k列的限制条件统一到某一列上，会发现任意统一到某一列的限制条件都是一系列的同余方程，我们利用中国剩余定理就能解决这个问题。

之后便是判断是否在范围内即可。

49.3 时空复杂度

时间复杂度：$O(\sqrt{t})$

空间复杂度：$O(\sqrt{t})$

49.4 备注

近乎是裸的CRT，有点没想到会出这样的题。

50 266E More Queries to Array

50.1 试题大意

你有一个数组$a_1, a_2, ..., a_n$，你需要回答m个两种询问。

1. 将$l..r$全部赋值为x
2. 询问$\sum_{i=l}^{r} a_i \cdot (i-l+1)^k$对$10^9+7$取模的结果。

$1 \le n, m \le 10^5, 0 \le k \le 5, 0 \le x \le 10^9$

50.2 算法讨论

区间问题肯定是上线段树了，我们只需要考虑合并过程。令一个线段树节点左右端点为l, r，如果我们维护$\sum_{i=l}^{r} a_i \cdot (i-l+1)^k$，那么合并的过程其实就是将右儿子中的$a_i \cdot (i-mid)^k \to (i-l+1)^k$，这个很容易利用二项式定理实现，但这样需要$k^2$的复杂度合并，不如我们换一种思路，直接维护$\sum_{i=l}^{r} a_i \cdot i^k$，这样我们在需要输出答案时利用二项式定理计算一下就行了。

50.3 时空复杂度

时间复杂度：$O(n \log n \cdot k)$
空间复杂度：$O(n)$

50.4 备注

很好的思路，我们换一种维护的思路就可以将复杂度降下来，非常有趣。

51 269E String Theory

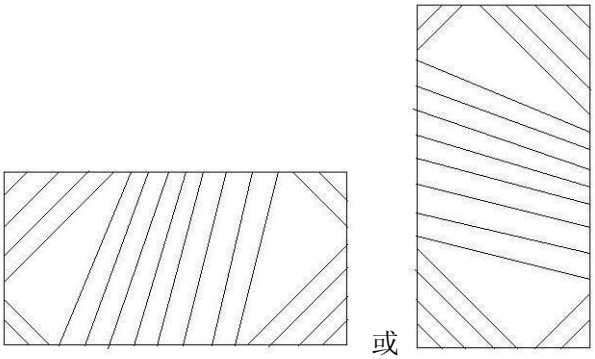
51.1 试题大意

有一个$n \cdot m$的网格图，每个靠边界的网格上的靠边界的边都可以连出去一条弦。我们可以重新排列行和列的顺序，使得弦两两不相交，有这么张参考图片：

题意有点复杂，可以参考英文题面。http://codeforces.com/problemset/problem/269/E

51.2 算法讨论

我们可以脑补一下最终的情况，肯定是下面两种情况之一：



不难发现最终情况非常容易计算，我们只需统计各个方向的弦的个数就行了。接下来就是判断当前方案是否能够通过重排行列顺序达到最终方案，如果我们将本来存在的弦看做边，将两条弦的端点看做点，将在同一行或者同一列的端点连上边，这样就会构成由很多环组成的图，我们只需利用hash判断这两个图是否同构顺便处理对应关系即可。

51.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n)$

51.4 备注

挺好玩的题目，也不是很难。图片是随手用mspaint画的，见谅咯。。。另外这题有一个很炫酷的名字，不过和题意和那个一点关系都没有。

52 332E Binary Key

52.1 试题大意

给出两个字符串p, q，称p为容器，q为钥匙，并且q为一个0,1串，密码是这样生成的：将q复制若干次直到长度不小于$|p|$，然后再将两个串串首对齐，p中对应1的位置依次选出来得到密码。现在给出密码串s和容器p，问长度为k的钥匙串字典序最小的是什么。

$1 \leq |p|, k \leq 2000, 1 \leq s \leq 200$

52.2 算法讨论

考虑我们已知了密码的长度以及钥匙串的长度，我们可以知道钥匙串与容器串的匹配情况，有些部分会匹配$|p|/k$次，有些会匹配$|p|/k+1$次，我们只需要枚举每部分有多少个1，就可以从后往前贪心的匹配，就是能匹配就配就行了，然后选择一个字典序最小的输出就行。

52.3 时空复杂度

时间复杂度：$O(k|p|)$
空间复杂度：$O(|p|)$

52.4 备注

还不算很难的题，一开始总关注于暴力DP，其实枚举之后倒过来贪心就行了，真是智商不够。

53 331E2 Deja Vu

53.1 试题大意

给出一个n个点m条边的有向图，保证任意两点间只有一条边，每条边

上有一个顶点序列，需要统计有多少路径使得边上的顶点序列依次拼起来能得到这条路径经过的实际的顶点序列，只需输出长度为1..2n范围内每个长度对应的路径个数即可。

$1 \leq n \leq 50, 1 \leq m \leq n \cdot (n-1)/2$，顶点序列总长度在$10^5$以内。

53.2 算法讨论

这个题还是挺有意思的，记得以前就遇到过然后一直不会做，最近来做的时候仔细想想发现还挺好做的。

这题的关键就是你可以发现一条路径一定能被分成若干有特殊性质的小段路径，不妨将依次经过的边上的顶点序列看做指示。如果当前仍然存在指示，并且这是一条合法路径（长度不超过2n）的话，那么他一定能不断按照指示继续往前扩展直到某个顶点指示恰好也到这个顶点为止，可以发现此时之前的路径对之后的没有任何影响；如果当前不存在指示，那么我们可能经过几条没有指示的边然后又获得新的指示，这是一种比较难考虑的方案，我们不妨考虑反过来做，即从“获得新的指示”的边反向往前寻找。我们的目的是找到所有这样类似的极小路径，然后将他们拼成长的路径。

考虑一条极小路径，如果他一定在某一时刻指示会“追平”或者“超越”当前的路径，不妨令这个情况发生在边(x, y)，那么(x, y)的指示上一定仅存在一个连续的$'xy'$，我们只需找到这个位置并且向前向后扩展路径就能得到一条极小路径，但仔细思考后发现我们还有细节问题，在一条极小路径之后拼接需要一条指示里没有起点的路径，同理往前拼接需要没有终点，那么我们可以统计出所有合法的4种路径，即指示中包含/不包含起点，包含/不包含终点，寻找方法与上述类似，可以发现他们总共只有不超过2m条。

统计长度为1..2n的路径数我们就可以DP了，令$f_{i,0/1,j}$表示当前已经走了i步，起点肯定已经被包含，终点是否被包含，并且路径的终点在j的方案数，我们只需要枚举i然后每次枚举用那条极小路径转移就行了，复杂度是$O(nm)$的。

53.3 时空复杂度

时间复杂度：$O(nm)$
空间复杂度：$O(nm)$

53.4 备注

非常有意思的题，还是挺有难度的，不过这次做起来很顺手让我很是意外，作为一名沙茶过了这题感到非常高兴。这背景还真贴切，既视感。

54 356E Xenia and String Problem

54.1 试题大意

如果称串s是一个格雷串，需要满足：

1. $|s|$是奇数。
2. $s_{\frac{|s|+1}{2}}$在串s中只出现一次。
3. 要么$|s|=1$，要么子串$s[1..\frac{|s|+1}{2}-1]$和$s[\frac{|s|+1}{2}+1..|s|]$是相同的而且均为格雷串。

如果一个串的某个子串$s[l..r]$为格雷串，将贡献$(r-l+1)^2$给答案。现在给你一个字符串s，允许你修改一个字符，最大化答案。

$1 \le |s| \le 10^5$，串仅有小写英文字母构成。

54.2 算法讨论

这题有不少性质，具体来说，首先由于长度限制，可能成为格雷串的子串个数就是$O(n \log n)$的，并且将所有合法的格雷串所有字符遍历一遍也是$O(n \log n)$的（因为不会有两个长度相同的格雷串相交）。这个解题带来的很大便利。

因为只需要修改一个字符，因此考虑修改每个字符的收益和代价，因为有左右子串相等的条件，可以使用hash来快速判断相等。首先考虑收益，我们只需暴力考虑每个可能成为格雷串的子串就行了，而这其中又有非常严格的限制条件，继续分情况，分别判断作为串中心和不作为中心。作为中心很容易判断，不作为中心又因为左右子串相等，也就是总有个模板，我们用manacher找出左右不同的第一个字符，修改这个字符再用hash判就行了。再考虑删除，注意这时候我们可以遍历所有的字符，和上面类似的分情况讨论即可。

之后再单独对每个位置统计答案就行。

54.3 时空复杂度

时间复杂度：$O(n\log n)$
空间复杂度：$O(n\log n)$

54.4 备注

只要观察到性质这题就好做了。

55 273E Dima and Game

55.1 试题大意

Dima在纸上写了n对数(l_i, r_i)，然后Dima和Anya轮流操作，可以干这么件事：

1. 选择一个$i(1 \le i \le n)$，并且$r_i - l_i > 2$。
2. 将第i对数替换为$(l_i + \lfloor \frac{r_i-l_i}{3} \rfloor, l_i + 2 \cdot \lfloor \frac{r_i-l_i}{3} \rfloor)$或$(l_i, r_i - \lfloor \frac{r_i-l_i}{3} \rfloor)$其中之一。

不能操作者输。

问满足每对数在$1 \le l_i \le r_i \le p$的n的数并且先手必胜的方案数对10^9+7取模。

$1 \le n \le 1000, 1 \le p \le 10^9$

55.2 算法讨论

可以看出一对数基本之后$r_i - l_i$也就是长度有关，和l_i, r_i具体是多少无关。

既然是博弈问题，那就考虑sg函数，n对数显然是独立的子游戏，单独考虑每对数的sg值异或起来就行了，现在的关键问题是需要求出长度为i的sg函数之后再进行统计。但p过大，又推不出sg函数的性质，只好打表找找规律咯。经打表发现sg总是成段（连续一段的sg都相同）出现的，而且段数看起来很少，于是用两个队列维护相同的段就能把sg函数都求出来。接下来的事情就是简单DP了。

55.3 时空复杂度

时间复杂度：$O(s+n)$，s为相同的段数。
空间复杂度：$O(s+n)$

55.4 备注

sg函数找规律题，不太喜欢。

56 333C Lucky Tickets

56.1 试题大意

找出m个八位数，允许有前导0，对于每个数允许在这个数的每个数字左右添上$+,-,\times,/$和括号，使得计算结果等于k。
$1 \le m \le 3 \cdot 10^5, 1 \le k \le 10^4$

56.2 算法讨论

考虑m的范围还是比较大的，直接搜索肯定有问题并且非常麻烦，但如果有太多情况不考虑又可能凑不齐m个，发现k不超过10^4，不妨枚举前4位的填数以及符号情况，再利用后4位借助加减法来调整，这样能得到比较多的解，也就可以通过本题了。

56.3 时空复杂度

时间复杂度：$O(m)$
空间复杂度：$O(m)$

56.4 备注

感觉这道构造题还是有点难度的。虽然思路很多但是好写的方法不多，而且这个方法也不是很好想。

57 240F TorCoder

57.1 试题大意

给出一个字符串s，要求你执行m个操作，每个操作给出l, r，要求将$l..r$之间的字母重新排列使之得到一个回文串，如果有多种方案则选择字典序最小的。输出最终的字符串。

$1 \leq n, m \leq 10^5$，均为小写拉丁字母。

57.2 算法讨论

要你写成回文串，脑补一下就能发现大概类似aaaaabbbbvbbbbaaaaa这样子的连续相同字母构成，而且你只需知道这一段各个字母的个数就能排出最后的样子，那么就写个线段树，区间修改+区间查就行了。

57.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n \log n)$

57.4 备注

两年前的题吧，虽然思路不难但放CF上还是要稍微写一会的。

58 305E Playing with String

58.1 试题大意

又到了游戏时间，每一步可以做这些操作：

1. 每次可以从可用的字符串中选出一个吗，令其为t，初始时只有一个字符串可选。
2. 从t中选出一个i，使得$t_{i-1} = t_{i+1}$。
3. 将t切成三段，分别为$t_1 t_2 ... t_{i-1}, t_i, t_{i+1} ... t_{|t|}$，放入可选的字符串集合。

问是否先手必胜，如果是找出最小的第一步位置在哪。

$1 \leq |t| \leq 5000$

58.2 算法讨论

如果直接以区间作为子游戏来求sg函数的话是会TLE的，我们可以仔细思考一下，不难发现题目中的条件其实并不强，我们只需要知道他相邻的两个，也就是说只会影响到边界。这样我们就不用具体知道这个区间内有些什么字母，只需弄清楚有多少个满足可以分裂的位置就行了。再利用以可以分裂的位置个数来作为子游戏，题目就迎刃而解了。找最小第一步用常规手段就行了。

58.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

58.4 备注

稍微要思考一下的sg函数，还有点意思。

59 235D Graph Game

59.1 试题大意

类似点分治地，提出随机图的点分治，大概有以下几步：

1. 对于当前的图G，令答案加上$|G|$。
2. 随机选取一个在G中的点x。
3. 将x从图中删掉，G变为若干连通块。
4. 递归处理各个连通块。

给出一个n个点n条边的图，对该图执行上述过程答案的期望。

$1 \leq n \leq 3000$

59.2 算法讨论

注意到给的图是n个点n条边，那么我们先考虑树的情况。不妨利用期望的线性叠加的性质来解决问题，首先可以把答案统计的方法转换一下，我们将答案拆分成若干个事件的贡献，即如果删除点x时，点y和点x连通，那么给答案贡献1。对于一对顶点(x,y)，一定会在删掉某个点之后两个点不连通，这个点可能在x到y的路径上等概率出现（因为整个过程都是在等概率随机，因此肯定是等概率出现），因此对答案的贡献是$\frac{1}{dist(x,y)}$。

再来考虑带环树的情况，利用同样的方法，可以发现如果(x,y)之间的路径经过环，那么情况就会有不同，但不会有太多差异。令x与环之间的点和y与环之间的点为A，环上一侧路径为B，另一侧为C。当x删除时要保持与y连通，要么是A,B均完整，要么A,C均完整，那么对答案的贡献为$\frac{1}{A+B}+\frac{1}{A+C}$，不过我们重复统计了$A,B,C$均完整的情况，减去$\frac{1}{A+B+C}$即可。

59.3 时空复杂度

时间复杂度：$O(n^2)$
空间复杂度：$O(n^2)$

59.4 备注

这是个很厉害的题，我也非常喜欢这题，毕竟出题人是丽洁嘛，还是非常可怕的。我毕竟还是太弱了，这题做过好多好多遍还是不怎么会证明，另外还想起了一句话：证明有的时候就是盯着结论反复看直到他变得显然。

60 342D Xenia and Dominoes

60.1 试题大意

给出一个$3\cdot n$的棋盘，你需要在棋盘上放置一些多米诺骨牌（$1\times 2, 2\times 1$），上面有一些'X'代表不能放置东西，有一个'O'代表放置完之后这里要是一个空格子。并且要求放置完之后能够有一个骨牌能够移动进空格子（类似华容道的移动）。问放置的方案数。

$3\le n\le 10^4$

60.2 算法讨论

对于最后能够移动，我们可以事先将能够移动进去的骨牌放好，但是要注意这是需要容斥原理去重的（考试时没注意到这个直接滚粗了）。接下来就是十分简单的状压DP转移了，可以写个DFS预处理一下，不是很好写但思路非常简单。

60.3 时空复杂度

时间复杂度：$O(n \cdot 2^4)$
空间复杂度：$O(n \cdot 2^4)$

60.4 备注

这样的爆搜题最讨厌了，不过少考虑一种情况也只能怪自己咯。

61 319E Ping-Pong

61.1 试题大意

有一大堆线段，线段a能到线段b的条件是a的两个端点至少有一个被b严格包含。现在给出两种询问：

1. 给出x, y，加入一条新线段(x, y)，保证他的长度比之前所有加入的线段都长。
2. 给出a, b，问第a条线段是否能到达第b条线段。

初始时没有线段。
$1 \leq n \leq 10^5$

61.2 算法讨论

这也是一道很好玩的题目，可以发现直接建图的话是一个有向图，这比较难办，我们希望把他变为无向图，这样维护一个并查集就行了。考虑线段长度严格递增这个条件，他其实保证了如果新加入的线段的端点落在旧的线段里面，那么这两个线段是可以互达的。这样我们就可以用并查集把满足这个关系的线段全部并起来，就只用考虑有向边的问题，而唯一的

有向边就是被完全包含的情况，只需要判断这两个范围是否存在完全包含的情况就行了。那么我们要找包含一个点的所有线段，并最后把他们并起来，这个用线段树就可以轻松解决问题。

61.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n \log n)$

61.4 备注

有一种特殊情况比较恶心，被出在清澄的数据里面了，就是两个互相相加的线段和一个与他们的并一样长的线段会有特殊情况。

62 301C Yaroslav and Algorithm

62.1 试题大意

有一个奇怪的算法。

1. 以一个字符串为输入，令这个串为a。
2. 她由一些命令组成，i号命令形如$s_i >> w_i$或$s_i <> w_i$，其中s_i, w_i均为长度不超过7 的字符串（可以是空串），由数字和'?'组成。
3. 她每次找到一个最小的i满足s_i是a的子串，如果没有那么算法终止。
4. 她会将在a中第一个出现的s_i替换成w_i，如果是$' >>'$那么算法继续，否则终止。
5. 她终止时会输出a。

希望你提供一个命令，使得对于给出的n个数，执行这个算法之后能够得到每个数加1。

你提供的命令条数不超过50。

62.2 算法讨论

形式上比较有趣的构造题，考虑我们做加法的过程：找到最后一位，给他加1，再考虑进位。同样的道理，我们可以用字符'?'来标记当前所在的位置。首先要生成一个'?'，那么一定是空串变过来的，因此肯定在第一个，我们再用一系列形如'?i >> i?'将'?'移动到最后一个，再将最后一个加1，再利用'?'表示进位的位置，但两个问号会重复，我们将位置的标记设成'??'即可，稍微注意下细节就能通过了。

62.3 时空复杂度

时间复杂度：$O(1)$
空间复杂度：$O(1)$

62.4 备注

有趣的构造题，比下面这个题有趣多了。

63 309D Tennis Rachets

63.1 试题大意

给出等边三角形，每条边上有n个等距离排列的洞，靠近端点的m个洞是不能使用的。问从可以使用的洞中选出三个并且组成钝角三角形的个数。

$1 \leq n \leq 32000$

63.2 算法讨论

直接n^2暴力twopoint扫描一下就行。

63.3 时空复杂度

时间复杂度：$O(n^2)$
空间复杂度：$O(n^2)$

63.4 备注

这这这这这题，竟竟竟竟然是暴力，32000的数据范围啊！我真是没什么话说，跑去向丽洁吐槽反而被吐槽回来：现代计算机跑得比你想象的快，几亿的复杂度都是靠谱的=。=。

64 301E Yaroslav and Arrangements

64.1 试题大意

称一个r个整数的序列a是好的，当且仅当他满足$|a_1-a_2|=1,|a_2-a_3|=1,...,|a_{r-1}-a_r|=1,|a_r-a_1|=1$，并且$a_1=\min_{i=1}^{r} a_i$。

称一个序列$b_1,b_2,...,b_r$是大的，当且仅当他满足：

1. $b_i \le b_{i+1}$。
2. $1 \le r \le n, 1 \le b_i \le m$
3. 如果我们能重新排列b，至少能得到1个，至多能得到k个不同的好的序列。

给出n,m,k，统计有多少大的序列，输出答案对10^9+7取模。

$1 \le n,m,k \le 100$

64.2 算法讨论

我们直接来考虑b序列是相当麻烦的事情，不妨先分析a序列的性质。不如将序列a的图象画出来，脑补一下就会发现他有一点类似“山脉”，考虑还有一个a_1是最小的条件，那么满足$+1,-1$个数相等并且最小值不小于0就能满足合法的要求了。再来考虑他能变换出多少好的序列，如果观察两两的差值不难发现这是一个括号序列，那么问题就好办了，那么我们只需将需要设的状态，即当前最大值，一共填了几个数，还有多少括号可以扩展，以及方案数设出来DP就行了。

64.3 时空复杂度

时间复杂度：$O(n^4)$

空间复杂度：$O(n^4)$

64.4 备注

这题竟然是CF有名的沙茶题专家Sereja的Round，能有这种难度的题实在是让人大吃一惊。

65 309B Context Advertising

65.1 试题大意

给出一个单词序列，你需要从中选出一段子段使得他能够写在广告牌上，顺序不能交换。广告牌是一个$r \times c$的，在一行中相邻的两个单词需要用空格隔开，一个单词不能被拆成两行。问能选取的子段最长是多少。

$1 \leq n, r, c \leq 10^6, r \times c \leq 10^6$

65.2 算法讨论

考虑已经确定下了区间，直接贪心地放置字符串就能判断是否合法。那么我们不妨枚举区间的起点，贪心地放置直到放不下，可以用一个递推推出每个单词作为行首最多能放的单词数，再加上一个倍增就能解决问题了。

65.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n \log n)$

65.4 备注

我的做法并没有很好的利用到$r \times c \leq 10^6$这个条件，肯定有更优的做法。

66 331D3 Escaping on Beaveractor

66.1 试题大意

在平面上有一个$b \times b$的正方形，左下角位于$(0, 0)$。平面上有n个平行于坐标轴的箭头，箭头是一些不会相交不会相切的线段，每个箭头有一个

方向，保证箭头和箭头的方向是对应的。现在有一个点，如果这个点碰到箭头就会沿着箭头的方向走，直到走出边界或者碰到下一个箭头。有m个询问，每次询问在(x_i, y_i)放置一个初始方向为d_i的点，经过t_i时刻之后的位置，每一个时刻走一个单位。

$1 \leq n, b, m \leq 10^5, t \leq 10^{12}$

66.2 算法讨论

如果将箭头抽象成一个点，与沿着该箭头能到的下一个箭头连上一条有向边，不难发现这是一个n个点n条边的有向图，也就是很多环套树形成的东西。我们把每个环套树都给找出来，同时给每个起点记录一下他下一步在哪，之后就是一个树上倍增，到环上时直接计算就行了。

说起来倒是轻松，把环套树找出来就要用到扫描线加平衡树，还有各种特殊情况难写死了，哼的。

66.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n \log n)$

66.4 备注

这个题挺麻烦的，CF上为数不多的代码题。有一些小技巧可以省掉一些代码，比如说扫描线的时候带一下方向就能避免写4份，在环上的情况也可以直接用倍增算出来，重新特判比较麻烦。

67 277D Google Code Jam

67.1 试题大意

现在你去打GCJ，比赛有n道题，持续t分钟，每道题都有小数据和大数据。做出小数据之后才能做大数据，做出小数据有sS_i分，解决大数据有sL_i分，但小数据需要tS_i分钟才能解决，大数据需要tL_i分钟。并且大数据有pF_i的出错概率（只能提交一次）。求最大期望的得分，在保证得分最大的同时求最小期望罚时。

GCJ的罚时是最后一个AC的代码的提交时间。

$1 \leq n \leq 1000, 1 \leq t \leq 1560$

67.2 算法讨论

如果只有第一问的话非常简单，直接背包就好了。稍微难一点的是加上了罚时，那么比赛的策略一定是先做完所有小数据，再来解决大数据，但大数据的解决顺序成了关键问题。不难发现交换相邻的两个大数据的解决顺序是不影响前后的，那么我们将其列出式子来就能得到将解决大数据的顺序排序的方法。但考虑DP时我们不可能知道当前需要解决的问题，我们不妨保持阶段顺序不变，换一种计算期望的方式就行了。那么接下来就是设$f_{i,j}$为前i个问题，耗时j分钟的最大期望得分，$g_{i,j}$为保证得到$f_{i,j}$情况下的最小罚时，写写转移方程即可。

67.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

67.4 备注

这题略有点卡精度，加上longdouble就行了。

我首先以为直接按出错概率从小到大排序就是解决顺序，后来发现错的时候已经写了DP，之后我直接按照转移方程推应有的顺序，怎么推都不对（毕竟智商堪忧），忘记考虑其实转移的时候他们的时间是不相等的，需要从原问题出发来推才能推得正确的式子。

68 341E Candies Game

68.1 试题大意

一个奇怪的游戏，初始时有n个箱子，第i个箱子里面有a_i颗糖，游戏的目标是把所有糖移动到2个箱子里，其他$n-2$个箱子都没有东西。你每步操作可以选择两个不同的箱子$i, j, a_i \leq a_j$，然后从箱子j往箱子i里移动a_i颗糖。你的任务是给出一个合法的操作序列，或者判断出无解。

$3 \leq n \leq 1000, \sum a_i \leq 10^6$，操作步数不能超过$10^6$。

68.2 算法讨论

这是一道非常帅气的构造题，太迷人了。考虑一种特殊情况，如果我们有一个1和2^n-1，那么我们就能无脑操作将其中一个变为0了。接下来的部分就是这题的精彩之处了，如果我们有一个1和另外一个任选的数呢？别忘了我们还可以借助其他的数，我们保留最大的数使得他不被选出，同样的我们每次都将1这个数乘2，不同的是当我们选出的数对应位置上不为1时，我们借助最大的数乘2，这样这种情况就解决了。考虑更一般的情况，当没有数是1 时，很容易想到拿最小的数出来，然后如法炮制上面的过程，但问题是这样并不确保会有数变为0。仔细思考之后可以发现，实际上我们在对$\lfloor \frac{b}{a} \rfloor$做二进制的分析，那么留下来的数实际上是$b - b/a \times a = b \bmod a$，可以发现最小的数在不断变小，我们重复这个过程就能把剩下所有的数清0。可以看出这样是无论如何都有解的。

68.3 时空复杂度

时间复杂度：不是特别容易计算，大概$O(n \log^2 n)$

空间复杂度：$O(n)$

68.4 备注

最开始做这题的时候我还打表打了很久，反正不会做也找不到规律，据说这是从数学竞赛书上抄的（还是IMO原题不太记得了），有点凶。不过这种从特殊情况开始考虑是做构造题的常规手段。

69 323B Tournament-Graph

69.1 试题大意

给出一个团，要求你给他的边定向，使得得到一个竞赛图，并且任意两点通过不超过两条边就能达到。

$1 \leq n \leq 1000$

69.2 算法讨论

这是一道构造题，不过构造的方法很多。

一种方法是可以发现如果n可行，那么我们可以构造出$n+2$，即增加两个点，一个连向所有点，一个由所有点连向他，然后两点之间再连一条边就行了。

一种方法是如果n是奇数，那么我们可以取出一个点，把剩下的点分为两半，然后分别由该点连向他或者被连入就行了，如果是偶数，那么单独拿出一个点，剩下的按奇数构造，再将这个点添上去就行了。

还有一种类似贪心的思想，我们考虑最终的图他的度数是平均的，那么我就希望他的度数尽可能的平均，每加入一个点就将连向之前度数少的一半，由度数多的一半连进来就行了。

69.3 时空复杂度

时间复杂度：$O(n^2)$
空间复杂度：$O(n^2)$

69.4 备注

一道构造题，最后那个挺乱搞的贪心算法是我的算法，和前面两个一比智商差异就立刻显现出来了。

70 248E Piglet's Birthday

70.1 试题大意

你有很多罐蜂蜜，装在n个箱子里面，第i个箱子里有a_i罐蜂蜜，你现在会做m次操作，每次操作在箱子u_i里随机选出k_i罐蜂蜜，并且把每一罐选出来的蜂蜜都品尝一下，然后在放进箱子v_i中。求每一步操作之后没有一个没被品尝的蜂蜜罐的箱子数的期望。

$1 \le n \le 10^5, 0 \le a_i \le 100, 1 \le q \le 10^5, 1 \le k_i \le 5$

70.2 算法讨论

考虑初始每个箱子里蜂蜜罐不超过100，然后每次品尝的罐子不超过5，就意味着我们可以暴力。被品尝过的蜂蜜罐实际上只当充数，不妨暴力记录，令第i个箱子里还剩j个未被品尝的罐子的概率为$p_{i,j}$，我们枚举有多少

未被品尝的罐子被选中了，然后利用组合数计算概率就行了。相当的暴力。统计答案就利用期望的线性叠加，把所有箱子的$p_{i,0}$加起来就是答案了咯。

70.3 时空复杂度

时间复杂度：$O(n \cdot |a_i| \cdot k)$
空间复杂度：$O(n \cdot |a_i|)$

70.4 备注

一开始想得相当复杂，还以为是难题呢，没想到这么暴力。

71 325C Monsters and Diamonds

71.1 试题大意

你找到了一本怪物书，书上告诉你如果给怪物喂食派的话怪物会吐出一些钻石。你非常高兴，但你发现喂食之后怪物可能会分裂成很多个怪物，也可能就不见了。现在有n种怪物，你知道了m种怪物分裂的方法以及选择这种方法会吐出多少钻石，并且保证每个怪物至少有一种方法。你最终不想要怪物了，所以你会在没有怪物的时候停止。你想知道对于每一只怪物，如果你初始只有这一个，你停止时最多和最少能获得多少钻石。你有无限多的派噢。

$1 \leq n \leq m \leq 10^5$，保证所有分裂方法能够得到的怪物总数不超过$10^5$。

71.2 算法讨论

我们直接设f_i, g_i为如果我们有第i个怪物，能得到的最大最小钻石数，那么如果是一个拓扑图的话我们就可以直接DP了。但可能有环是这个题比较有趣的地方，这有什么影响呢，是不是仍然可以按照拓扑图那样弄一个标记数组直接DP，当然不work。直接记忆化搜索求的问题是会搜索到正在计算的点，那么最大值是没有问题的，这时候就直接无限了。关键是计算最小值时这里就不知道如何决策了，我们考虑在拓扑图上DP还可以求出拓扑序之后倒着做一遍，如法炮制，我们从没有怪物的状态往前推，每次选一个权值最小的拓展就行了。

71.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n)$

71.4 备注

两种方法套在一起用，一拿到这题我就无脑码了个记忆化然后才发现有bug，真是智商捉急，果然应该思考清楚之后再动手。

72 266D BerDonald

72.1 试题大意

给出一个n个点m条边的带权无向图，求出一个位置，使得该位置到最远的点距离最短（可以在边上），输出最短距离。

$1 \leq n \leq 200$

72.2 算法讨论

思路很简单，我们枚举需要找的位置在边(x, y)，那么对于另外任何一个点i，仅考虑这个点的情况便是位置在某个端点是最优的。那么接下来就是多个点的信息需要并在一起，考虑$i \to x \to y \to i$这条“链”的中点（中点一定会在边上，路径都走最短路，不用考虑路径重叠的问题），对于这个点的计算我们只需将最终的位置减去它到中点的距离再加上这条链的一半就行了，这个可以排序之后DP一下，注意此时就要保证最终的位置一定在边上。

72.3 时空复杂度

时间复杂度：$O(n^3 \log n)$
空间复杂度：$O(n^3 \log n)$

72.4 备注

这么简单的题我竟然做了1个多小时，一直拖着不想写，太懒了。

73 311C Fetch the Treasure

73.1 试题大意

你在一个长长的数轴的位置1，初始时你每次可以往前跳k步。给出了m个下面三种操作。初始时有n个宝藏，第i个位于a_i，价值c_i。

1. 增加一个x，此时你可以往前跳x步，如果当前你能往前跳$a_1, ..., a_r$步，那么你能到$1 + \sum_{i=1}^{r} v_i \cdot a_i$，$v_i$是正整数。
2. 让第x个宝藏的价值减少k。
3. 询问当前能到达的最大的宝藏价值，然后就将他的价值清空。

$1 \le n, m \le 10^5, 1 \le k \le 10^4, 1 \le x \le 10^{12}$，操作1不超过20个。

73.2 算法讨论

这题有个至关重要的条件就是$1 \le k \le 10^4$，在其他数据范围都很大的时候，这里成了解题的关键。一个很常规的思路，只问是否能够达到的话，可以将数轴上的点分为k类，对于第i类点满足$v \bmod k = i$，我们只需要找到最小的一个v使得v能被走到，那么该类里比v大的数就均能达到了。我们只需要在每次加入一个数时做一遍最短路就能算出这个值。接下来问题就好办了，我们把能够到达的宝藏维护成一个堆，每次就是堆的操作了。

73.3 时空复杂度

时间复杂度：$O(n \cdot k \cdot \log n)$
空间复杂度：$O(k + n)$

73.4 备注

这确实是一种挺常见的思路，不过我还是想了一会，一开始一直纠结是不是2^{20}之类的复杂度。

74 316D3 PE Lesson

74.1 试题大意

给出一个n个数的排列，每个位置要么可以交换1次，要么可以交换2次，问通过任意次交换能够得到多少不同的排列。

$1 \leq n \leq 10^6$

74.2 算法讨论

如果所有位置都被允许交换两次，那么方案数就是$n!$了，再考虑只能交换一次的，不难发现如果将其复原的话会经过一条链，所有这样的链互不影响，而对于一条链我们并不关心具体是怎么交换的，只用管最终的结果就行了。那么就有思路，令$f_{i,j}$表示当前有i个只能交换一次，j个可以交换两次，DP就行不过会TLE。

假设初始时有n个只能交换一次的，那么考虑最后一个人，要么他自己形成一条链，要么和另外一个人，因此方案数f_i有$f_i = f_{i-1} + f_{i-2} * (i-1)$。再考虑现在又$m$个能交换两次的人，我们把这$m$个人插入进去，如果他插入一个长度为$n$链，那么他只要不放在原位置就肯定合法，因此方案数为n，另外他可以新加入形成一条链，因此方案数为$f_n \cdot (n+m)!/n!$。用这个就能通过本题了。

74.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

74.4 备注

感觉还是不简单的题目，关键就是注意到不需要考虑他们的交换顺序，因为最后统计的排列个数。

75 323C Two permutations

75.1 试题大意

给出两个长度为n排列，m个询问形如(l_1, r_1, l_2, r_2)，问有多少个数在第一个排列里出现在$l_1..r_1$，第二个出现在$l_2..r_2$，强制在线。

$1 \leq n \leq 10^6, 1 \leq m \leq 10^5$

75.2 算法讨论

直接用主席树嘛，还是函数式线段树？不管了，就是说考虑给每个第一个排列的前缀建一个线段树，线段树里存储这个前缀出现在第二个排列中的位置，询问用区间减法即可回答。

75.3 时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n \log n)$

75.4 备注

CF上会出这种题真是比较意外，直接考裸的数据结构不是偏袒中国人么。

76 343E Pumping Stations

76.1 试题大意

给出一个n个点m条边的无向图，每条边有一定的流量限制，要求选出一个顶点的排列$v_1, v_2, ..., v_n$使得v_1与v_2，v_2与v_3，...，v_{n-1}与v_n的最小割之和最大。输出最大值并给出方案。

$1 \leq n \leq 100, 1 \leq m \leq 2000$

76.2 算法讨论

这题直接用最小割树就行了。

最小割树是说将一个无向图等价成一棵树，顶点不变，在无向图中的两点的最小割等于在树上两点的最小割。有了这棵树那么这题的答案就显然了，就是所有树边权值之和嘛。我们考虑树上最小的一条边，不妨令最终的答案序列为v，如果它被经过多次，那么在所有经过这条边的相邻两点断开，这样就会将序列分为完全在某一侧的很多段，再把在相邻一侧的段合并起来，就得到一个只经过最小的边一次而且答案不会劣的方案。

考虑构造方案，我们任意选一个点作为起点，在所有没有被选择的点中找一个到该点路径最小值最大的点，这样下来就能构造出答案。

76.3 时空复杂度

时间复杂度：$O(n \cdot maxflow(n,m))$
空间复杂度：$O(m)$

76.4 备注

CF上竟然考最小割树而且还是Div2E，比上一题更加不可思议。。。看到代码里有一份非常精短的最小割树写法，十分膜拜。

77 267C Berland Traffic

77.1 试题大意

现有一个n个点m条边的网络。每条边有一个容量限制，对于一条边$< x, y >$，设其流量为x，容量为c，如果x大于0则是从x流向y，反之就是从y流向x，但x的绝对值不能超过c且可以不是整数。网络中节点1是源点，节点n是汇点，对于除$1, n$外的所有点，流入的流量等于流出的流量。这个网络有一个奇怪的性质，对于任意一对节点(x, y)，x到y的路径上流量之和不会随着选择不同的路径而改变（有可能有小于0的流量，流量的符号取决于x到y路径上这条边的方向）。

求满足上述条件且流过该网络的流量尽可能大的方案以及最大的流量是多少。通过网络的流量是指从节点1流入或者从节点n流出的流量。

$2 \le n \le 100, 1 \le m \le 5000, 0 \le c_i \le 10000$

77.2 算法讨论

这是一道基于网络模型的题目。

考虑任意节点路径流量之和不变这个条件，其实这给了很强的限制。毫无疑问这肯定是解题的关键了。如果根据一般网络的步骤利用边的流量直接考虑问题是非常困难的，需要借助题目条件进行转化。假设当前已经得出了通过网络的流量最大的一组方案，从节点1出发，无论怎么走，到达每个节点时经过路径的流量之和一定是定值，这个值只和最终到达的点有关，不妨令到达$1,2,...,n$的流量之和分别为$x_1,x_2,...,x_n$。可以发现，所有边的流量都可以用$(x_1,x_2,...,x_n)$表示，具体的，对于一条边$< u,v >$，其流量为$x_v - x_u$。

在得到边的流量表示后，就可以列出满足网络这个性质的关系式了，具体的：

每个点（节点$1,n$除外）流入等于流出：

$$\sum_{<u,v>\in e} x_v - x_u = 0$$

每条边的容量限制：

$$-c \le x_u - x_v \le c$$

要求最大化目标函数：

$$\sum_{<u,n>\in e} x_n - x_u$$

注：可以直接令$x_1 = 0$从而去掉x_1这一未知数。

得到了一个线性规划的模型，尝试用单纯形算法直接解决。可以发现虽然能够很方便的找到一组初始可行解，但是对于任意一种可行解，单纯形的主元操作均不可完成，因此单纯形算法是不适用的，采用其他线性规划算法也不是解决本题的上策。

可以发现把等式条件联立起来似乎是可以通过解方程来确定未知数的，不过这里离解出未知数仍少了一个方程，一般的方法是将其中一个未知数看做常数来解，但这样的问题是不能确定方程是否有唯一解而变得不靠谱。

仔细观察限制条件，不难发现限制条件的等式部分组成的线性方程组的系数矩阵其实是去掉两行一列的Kirchhoff矩阵(Laplacian matrix)。根

据matrix-tree 定理（Kirchhoff's theorem），Kirchhoff 矩阵的行列式为0，删掉任意第i行第i列之后的行列式为其生成树个数，而只要一个图连通其生成树个数一定不为0，也就是说假设这个矩阵是一个线性方程组的系数矩阵，那么这个线性方程组可以解出一组唯一解。但现有的系数矩阵是去掉两行一列的Kirchhoff矩阵，并且如果将最大化的目标函数当做一个方程会得到一个去掉第1行第1列的Kirchhoff矩阵。那么不妨另$w = \sum_{<u,n>\in e} x_n - x_u$，也就是要最大化的目标函数值，在原有方程组中加上一组：

$$\sum_{<u,n>\in e} x_n - x_u - w = 0$$

这样方程组的系数矩阵就变成了去掉第1行第1列的Kirchhoff矩阵，是可以解出唯一解的，也就是说解可以写成$x_i = a_i \times w$的形式。再来考虑每条边的容量限制，此时目标便很明显了：将w带入到每条边的容量限制去，可以得出w的取值范围，直接取一个最大的w，再将w带回表示出每条边的流量，输出方案即可。

看似问题解决了，其实忽略了当整个图不连通时，矩阵行列式的值仍然是等于0。这其实很好处理，当$1, n$不连通时，w显然最大是等于0的，当$1, n$连通但有其他部分不连通，可以单独取出节点1所在的连通块，也可以直接跳过那些不与$1, n$连通的点。

至此问题便圆满解决，复杂度$O(n^3)$。

77.3 时空复杂度

时间复杂度：$O(n^3)$
空间复杂度：$O(n^2)$

77.4 备注

来自自己写的详细题解。

78 240E Road Repairs

78.1 试题大意

给出一个带权有向图，边权仅为0/1，一个以1为源点的最小树形图。
$1 \le n, m \le 10^5$

78.2 算法讨论

直接套用最小树形图的算法好像是可以AC的。不过还要算方案，会比较麻烦。

不过这题有更好的性质，边权仅为0/1。考虑直接简化最小树形图算法，首先将所有边权为0的边缩强连通分量，对于每个分量只需要一条入边即可，如果有为0的入边直接选，之后在DFS一遍看看要选哪些1的入边就行了。

78.3 时空复杂度

时间复杂度：$O(n)$

空间复杂度：$O(n)$

78.4 备注

最小树形图，CF上又多了一个奇怪的算法，不过能有这些算法也不错。

79 329E Evil

79.1 试题大意

给出n个点的坐标，求一条最长的Manhattan距离哈密尔顿回路。

$1 \leq n \leq 10^5$，坐标范围10^9。

79.2 算法讨论

一个结论题。

考虑我们最大能够得到的答案。我们将最终的绝对值拆开，不难发现对于x坐标一定有n个正号和n个负号，y坐标也是同理。考虑我们能够得到的最大答案就是将x坐标每个值写两遍然后再排序，那么我们能够得到的最大答案就是前n个数都是正号，后n个都是负号，y坐标也如此。我们能够同时得到这两个最大值么，当然在有些情况下是可以的。

接着我们考虑如何得到这个上界，不妨利用x,y坐标的中线将整个区域划分成左上，左下，右上，右下A, B, C, D四个区域。那么最简单的情况就

是只有A,D或B,C里面有点，这样直接连就能达到最大值。否则我们就要像$B\to C,C\to B,A\to D,D\to A$这样，但注意到$B\to A$这一步很可能没法达到最大值，这时如果中线上有2个点即可借助中线上的点。否则要么中线上仅有一个点，要么n是个偶数，可以证明这两种情况下上界是没法到达的，但是我们可以构造出仅次于上界的解，也就是最优解了。

实际实现时分类讨论一下就可以解决了。

79.3 时空复杂度

时间复杂度：$O(n)$
空间复杂度：$O(n)$

79.4 备注

直接思考如何求得最优值几乎是不可能的，既然没有固定算法就只好找找有什么客观结论，从可以逼近上界这个方面思考确实是一个很好的思路。CF上也是有很厉害的结论题的。

80 303E Random Ranking

80.1 试题大意

有n个数，每个数在$[l_i,r_i]$中随机取值（可以是实数），从小到大排序后问每一个数排在每一个位置的概率。

$1\le n\le 80,1\le l,r\le 10^9$

80.2 算法讨论

考虑如果我们知道了某个数的最终取值，我们可以算出这个数排第几位的期望。但现在每个数的取值是在一个区间内随机，如果可以离散化那么可以根据期望的线性叠加枚举每个取值最后加起来，不过不能离散化我们可以发现这其实就是若干一次函数的和，那么加起来就变成积分起来，这样我们有了一个初步的思路，可以算出关于第i个数取值的一次函数（是很多分段函数），然后把它们在$[l_i,r_i]$区间内的函数积分起来就能算出答案，复杂度是$O(n^5)$，考虑到每个数的分段函数其实差别不大，可以利用分治优化至$O(n^4\log n)$。

不过很奇怪，我写上面这个方法怎么都有精度问题，只好写了个$O(n^5)$的暴力DP再卡卡常数才能AC，因为考虑到其实不用维护分段函数了，我们暴力离散化之后每一个区间看做一个整体，这样还是可以DP的，从结果上来说精度也是好一些的。

80.3 时空复杂度

时间复杂度：$O(n^5)$
空间复杂度：$O(n^5)$

80.4 备注

比较有趣的题目，还是花了很久的时间想，不过有意思的是这题在比赛testing之后发现finaltest是错的，出现了精度误差导致概率变成负数了，被tourist发现了，不过pretest是对的而且考试的时候没人过，所有本来要变成unrated 的还是rated了。

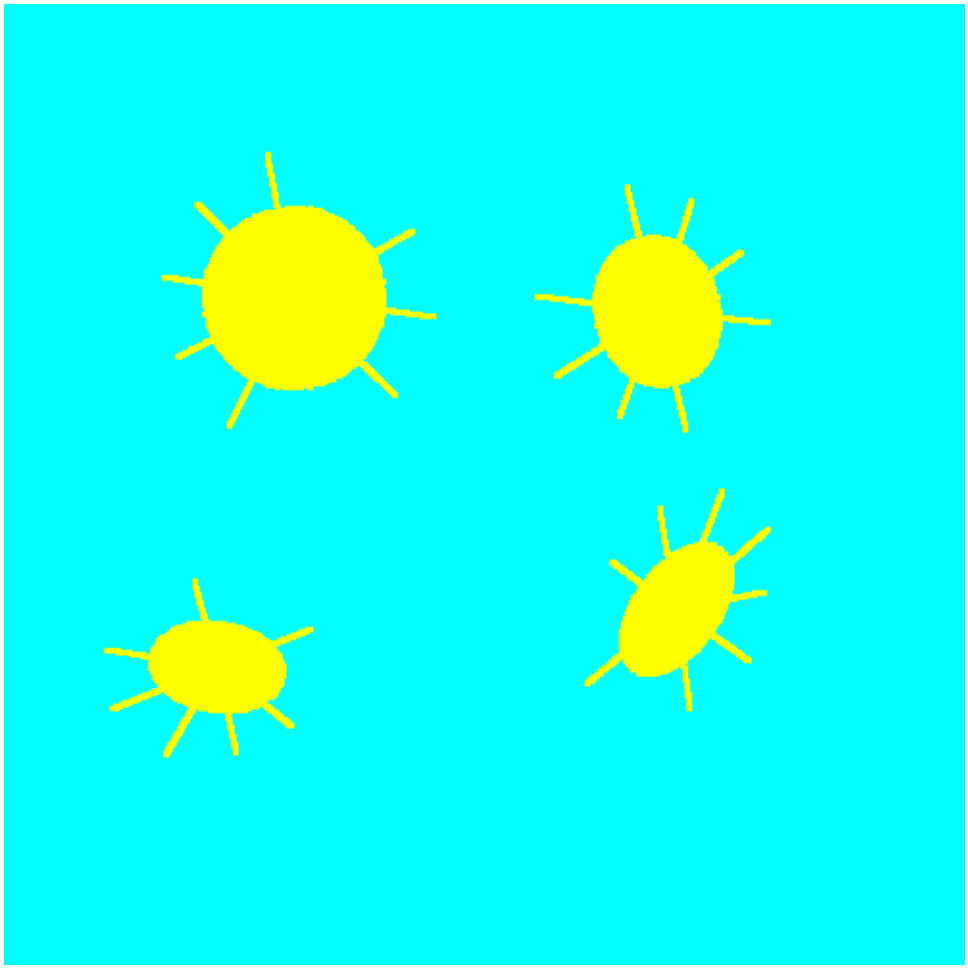
81 316F3 Suns and Rays

81.1 试题大意

在一张图上画了很多个太阳（用1表示有颜色，0表示白色），太阳是一些椭圆，轴的长度在40到200之间，太阳周围有一些光线，光线的宽度等于3个像素点，长度在10到30之间，所有太阳不相交。问有多少个太阳和每个太阳有多少条光线。

$1 \leq h, w \leq 1600$

大概像这样：



81.2 算法讨论

这题的算法还是挺多的。考虑光线的宽度为3，而太阳最短的轴也有40，那么我们只需要高斯模糊（可能不是高斯模糊吧，就是说取周围九个格子中，如果黑格子不超过多少个个就把这个格子变成白色的）几次光线就会消失。再用高斯模糊的逆操作将每个太阳的大小变回来，之后对比新的图和原图，如果原图中为1 的点在新的图里面为0了，那么这里就很有可能是一条光线，调调参数就可以AC 这题了。这确实是一个很有趣而且很厉害的思路。

81.3 时空复杂度

时间复杂度：$O(n^2)$
空间复杂度：$O(n^2)$

81.4 备注

做图形识别题就是反复调参数，WA好多好多遍。。。

82 360E Levko and Game

82.1 试题大意

有一个n个点$m+k$条边的有向图，其中k条边的边权可以由你定，范围在l_i, r_i，其他的边的边权已经定好了。你希望让s_1到f的最短路比从s_2到f的最短路更短，如果可以则要输出方案。

$1 \leq n, m \leq 10^4, 1 \leq k \leq 100, 1 \leq l_i, r_i \leq 10^9$

82.2 算法讨论

很容易发现需要我们定边权的边要么是l_i，要么是r_i，不妨先将初始边权定为r_i，之后在考虑哪些边需要变成l_i。那么有个贪心的思路就是，首先按照距s_1的距离由小到大考虑每条待定的边，如果从s_1到当前边的末端比从s_2更远，那么就将这条边设为l_i。最后再做一遍Dijkstra求出最短路比较一下就行。这个做法基本上就已经保证正确了。

82.3 时空复杂度

时间复杂度：$O(nk\log n)$
空间复杂度：$O(nk\log n)$

83 338E Optimize!

83.1 试题大意

给出两个序列a_n, b_m，对于a中每一段长度为m的连续序列，如果存在一种b的排列使得对应位置的数之和均不小于h，则给答案加1。求最终答案。

$1 \le m \le n \le 150000, 1 \le h \le 10^9$

83.2 算法讨论

可以观察到题目的条件是均不小于h，因此有一种贪心的算法，即我们可以将b中最大的元素匹配目前a中最小的元素，第二大的匹配第二小的以此类推，很显然如果希望都不小于h那么这肯定是最优策略。再考虑a中长度为m的相邻的两端，操作仅仅是删除一个数，再添加一个数，考虑用平衡树来维护这步操作。很容易想到的办法是维护全局$a_i + b_i$ 的最小值，然后与h比较大小，这显然是不容易维护的。考虑我们当期已经知道了h，不妨给每个a_i维护要想使得$a_i + b_j \ge h$的最小的j，这样就可以变成区间操作了。于是写个平衡树就能解决这题。

83.3 时空复杂度

时间复杂度：$O(n\log n)$
空间复杂度：$O(n\log n)$

83.4 备注

官方题解好像是根号算法，不过我顺着想下来只想到了平衡树，所以就写了，也不是特别难写吧。

84 285E Positions in Permutations

84.1 试题大意

在一个排列p中，如果$|p_i - i| = 1$，则称位置i是好的。统计大小为n的排列中恰好有k个位置是好的的方案数对$10^9 + 7$取模。

$1 \leq n \leq 1000, 0 \leq k \leq n$

84.2 算法讨论

如果直接DP会发现需要记录的东西非常多，也不怎么好优化转移，虽然不知道可不可做但$O(n^2)$的时间内肯定是没什么希望了。不过可以发现如果仅仅保证至少有k个位置是好的，也就是说保证k个位置是好的其他的随便放是非常容易计算的（我是采用的奇偶分开DP的方法）。那么我们再考虑去重的问题，对于一个恰好有k个好的位置的排列，在用我们的统计方法计算至少有b个好的位置的排列时被重复计算了$\binom{k}{b}$次，减掉就行了。

问题至此便圆满解决。

84.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

84.4 备注

作为Div2的E还想错了一次去重，以为直接减掉就行了，真是图样。

85 238D Tape Programming

85.1 试题大意

有一种奇怪的语言，他的程序是写在一行上的，初始时光标在最左侧，方向向右。程序包括数字和箭头，如果光标遇到了一个数字，则输出这个数字并且将数字减1，如果是0则删除这个数字，如果是箭头光标的方向将改为箭头的方向，如果下一个仍是箭头则将当前的箭头删去。处理完这个位置之后光标向光标的方向移动一步，当光标移动出这一行（在第一个位

置向左，最后一个位置向右）时结束程序。给出一段长度为n的程序，m个询问，每次询问拿出l_i, r_i当做一段程序，每个数字被输出了多少次。

$1 \leq n, m \leq 10^5$，数字是0到9。

85.2 算法讨论

这是一道很有趣的题目。

考虑我们要单独运行的l_i, r_i，如果我们运行这个程序，那么一旦某次我们进入了区间$[l_i, r_i]$，我们就开始运行了这个询问要运行的同样的代码，如果我们从这里出去，那么也就结束了，可以发现在进入这个区间之前的代码对当前运行结果没有任何影响，那么每次询问其实就是我们运行整个代码的一部分过程。那么我们要做的就是运行源程序，如果他结束了但程序仍然没有被完全删除就再运行一遍，直到整个程序为空。之后每次回答询问只用找到对应的一段，我们记录前缀和进行回答即可。

85.3 时空复杂度

时间复杂度：$O(10n)$
空间复杂度：$O(10n)$

85.4 备注

在如果程序没被完全删除就再运行一遍这部分想了很久，一直没有什么思路，不过这个性质也确实是非常漂亮。

86 331C3 The Great Julya Calendar

86.1 试题大意

给出一个数n，你每次可以选出他某一位上的数字，然后将n减掉这个数字。问n最少需要几次操作才能变为0。

$1 \leq n \leq 10^{18}$

86.2 算法讨论

直接的思路肯定是数位DP吧，对于一个前缀你只需要知道最大的数字

是多少就行了。但这样非常麻烦代码也不怎么好写。于是有了一个记忆化搜索的办法，令$f_{n,d}$表示当前需要处理的数为n，n之前可能还有一些数字他们的最大值为d，最少需要几次才能将n减少到小于0，然后对于一个数n，肯定是先减掉个位，再减掉十位，这么以此类推，类似数位DP的方法进行记忆化搜索即可。可以估算一下状态数大概不超过2^{18}级别，应该是可以通过的。

86.3 时空复杂度

时间复杂度：$O(s \log n)$，s为状态总数。

空间复杂度：$O(s)$

86.4 备注

只会用数位DP的方法呀，看了看status里的代码发现竟然有这么短小的写法就学习了一下，真是弱爆了。

87 288E Polo the Penguin and Lucky Numbers

87.1 试题大意

幸运数字是指那些十进制下仅有4,7构成的数，给出两个幸运数字l, r，令$l..r$之间的幸运数字从小到大第i个为a_i，求$a_1 \cdot a_2 + a_2 \cdot a_3 + ... + a_{n-1} \cdot a_n$对$10^9 + 7$取模的结果，保证$l, r$位数相等。

$1 \leq l, r \leq 10^{100000}$

87.2 算法讨论

喜闻乐见的数位DP题。

首先使用区间减法，接下来就只用考虑从44..4到l之间的答案了。再从高位往低位DP，预处理f_i表示长度为i的44..4到77..7之间的答案，g_i为44..4到77..7之间幸运数字的和来辅助f_i的转移，用分配率将乘法拆开就是了。之后的事就是分情况讨论转移了。

87.3 时空复杂度

时间复杂度：$O(\log r)$
空间复杂度：$O(\log r)$

87.4 备注

最开始思考的时候竟然忘记用区间减法了，真是到什么都不会的地步了QAQ。。。而且这种题不应该是Div2E么，怎么变成Div1E了。。。

88 241F Race

88.1 试题大意

给出一个$n \times m$的城市地图，如果一个格子是'#'那么他是一个建筑物，如果是一个数字则表示一个街区，数字就是通过这个街区所需的时间，如果是一个字母则表示路口（一个字母只出现一次），通过路口需要1个单位时间。相邻的街区组成道路，保证道路的宽度为1，并且路口一定在道路的两端。建筑物是不准进入的，可以在其他的格子上行动，并且可以走到往相邻的格子。给出起点，终点和路口序列，你需要从起点出发，按照要求依次通过这些路口，再到达终点，你会采用最短路行动，并且到达终点之后便一直停留，询问时刻k时你的位置。

$1 \leq n, m \leq 100, 1 \leq k \leq 100000$

88.2 算法讨论

可以发现路口之间的路径是唯一的。

求出路口之间的最短路并且记录路径之后暴力模拟就行了，k非常小。最短路用SPFA或者Dijkstra都行吧，直接写就是了。

88.3 时空复杂度

时间复杂度：$O(n \cdot m \cdot 26)$
空间复杂度：$O(n \cdot m)$

88.4 备注

这题主要是卡题意吧，其实是非常简单的题。

89 243D Cubes

89.1 试题大意

给出一个$n \cdot n$的网格，格子(i,j)上放了$c_{i,j}$个长度为1的立方体，问从无限远处以$(v_x, v_y, 0)$的方向看过来，能看到多少方块，一个方块的边界是属于这个方块的。

$1 \le n \le 10^3, 0 \le c_{i,j} \le 10^9$

89.2 算法讨论

考虑这些方块在$\vec{v}$方向上的投影，这样就可以将原本的立体图形转化成平面图形。那么判断在当前图形的最后新加入一列能看到多少方块也非常简单，只需找到对应区间内的最小值，用线段树维护一下就行了。接下来的事就是把每一列按从前往后的顺序加入就行了，虽然有10^6的数据范围不过时限较大还是可以接受的。

89.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n)$

89.4 备注

在投影时可以点积或者叉积来将所有边界设置成整数，方便运算。实际实现线段数的过程中也不需要打tag，我们在每一整段区间记录一下，询问时稍作修改即可。

90 238E Meeting Her

90.1 试题大意

给出一张n个点的有向图，你需要从a到b去，有很多条bus路线，每条

路线起点是s_i，终点是t_i，并且该bus会随机选择一条最短的路径从起点去终点，每一时刻都有一台bus出发。如果有bus经过你所在的节点，那么你可以上车，并在之后选择一个时候在某个节点下车（注意你永远都不知道bus的路线）。如果你能在最坏情况下用有限的时间到b，输出你最少换bus的数目，否则输出-1。

$1 \leq n \leq 100$

90.2 算法讨论

因为只用考虑最坏情况，我们就不把路线说成"随机"这么玄乎，大概是一种你想要我去哪我就偏不去的感觉。那么很有可能会出现你坐不上bus的情况，也就是只能在s_i至t_i的最短路的割点上才能坐上这个bus。之后我们可以考虑逆向来DP，直接令f_i为当前在节点i需要最小的换乘次数，初始时我们知道$f_b = 0$，然后枚举一遍公交路线之后可以递推出一些新的f，每个点会被更新一次，每次更新是$O(km)$的，所以还是可以通过这题的。

90.3 时空复杂度

时间复杂度：$O(nkm)$
空间复杂度：$O(nkm)$

90.4 备注

如果用迭代的写法很有可能在时间上更快，不过复杂度没变。这题清澄上竟然是100ms的时限，你说好好的1s怎么着就改成100ms而且$O(n^4)$的算法跑出来15ms并且也没有更好的算法，单纯的没有必要啊。

91 251D Two Sets

91.1 试题大意

给出n个数$a_1, a_2, ..., a_n$，要求把这n个数分成两个集合，不妨称其为A, B。令A中所有数异或起来为$f(A)$，要求最大化$f(A)+f(B)$。如果有多种情况，就保证上一个条件满足的情况下最小化$f(A)$。特别的，$f(\varnothing) = 0$。

$1 \leq n \leq 10^5, 1 \leq a_i \leq 10^{18}$

91.2 算法讨论

位运算按位考虑是常规思路，那么对于每一位，如果我们能在两个集合里分别凑出一个1，那么达到最大化的目标了。如果该位上有奇数个1，那么无论如何拼凑，$f(A)+f(B)$的这一位上一定是1了，如果是偶数个，不妨令x_i表示第i个数是否选入了B，那么对于每一位我们可以得到一个异或方程，贪心的从高位往低位判断每一位是否可能等于1就行了，最小化$f(A)$也可以用同样的方法。

具体的，如果异或方程有解，那么方案可行。有无解的判断可以不需要每次都重新消元一遍，我们维护前i个方程在线性空间上拓展的基就行了。

91.3 时空复杂度

时间复杂度：$O(n\log n)$
空间复杂度：$O(n)$

91.4 备注

真是奇怪的线性方程，第一次见到n个未知数$\log n$个方程的题。其实两个部分用的方程没有重叠，可以稍作调整让其变成相同的目标，这样就只用写一遍了。

92 351D Jeff and Removing Periods

92.1 试题大意

给出一个序列a，你可以执行下面的操作。

- 选出三个整数$v,t,k(1\le v,t\le n;0\le k;v+tk\le n)$，并且$a_v=a_{v+t},a_{v_t}=a_{v+2t},...,a_{v+t(k-1)}=a_{v+tk}$。
- 删除$a_v,a_{v+t},...,a_{v+tk}$，剩下的元素按顺序排行。
- **你可以重排剩下的序列**，并且重复操作。

一个序列的美丽度是删除整个序列的最小操作次数。给出一个序列$b_1,b_2,...,b_n$，和m个询问，每次询问序列$b_l,...,b_r$的美丽度。

$1 \leq n \leq 10^5, 1 \leq m \leq 10^5$

92.2 算法讨论

考虑可以重排这个序列，所以问题的答案就是：不同的数的个数+ 是否不存在一个数使得其位置形成等差数列。不同的数的个数可以用函数式线段树快速维护，不过后面那个问题就有点问题了。不妨考虑更暴力的做法，因为没有修改操作，因此考虑用莫队算法，然后直接$\sqrt{n}\log n$维护就行了。

92.3 时空复杂度

时间复杂度：$O(n\sqrt{n}\log n)$
空间复杂度：$O(n\sqrt{n}\log n)$

92.4 备注

你可以重排重排重排重排这个序列阿，我都想了炒鸡久的时间怎么都想不出来。加上这个条件瞬间就变傻B题了，不能忍啊。

93 293E Close Vertices

93.1 试题大意

给出一棵树，每条边有边权。询问有多少对点他们之间路径边数不超过l，边权和不超过w。

$1 \leq n \leq 10^5$

93.2 算法讨论

这不就是个裸的树分治么，分治之后将所有点按到当前根节点边权和排序，然后用树状数组维护一下路径边数，twopoints单调扫一遍就行了。

93.3 时空复杂度

时间复杂度：$O(n\log^2 n)$
空间复杂度：$O(n\log^2 n)$

93.4 备注

题目太裸都不知道写些什么。

94 348E Pilgrims

94.1 试题大意

给出一棵树，每个节点要么是黑色的，要么是白色的。一个黑点坏了的条件是他与所有离他最远的黑点都不连通，你需要删除一个白色的节点，使得最多的黑点坏掉。输出你最多能使多少黑点坏掉以及删除白点的方案数。

$1 \leq n \leq 10^5$

94.2 算法讨论

一个比较常规的做法是：首先删除不必要的白点（大概就是不断删除度为1的白点，直到没有），之后可以发现，对于一个黑点，他和离他最远的黑点的路径上一定经过直径的中点，看上去这个条件没有什么用，不妨将直径的中点作为根，便可以发现我们可以进行DP了。脑补一下需要统计的路径只有从子节点进入父节点，这还是非常容易树形DP的。

一个比较奇葩的做法：考虑树分治可以遍历所有的路径，不过这题的要求稍微高一点，需要知道对于每个点经过他的一些路径，那么我们来一个树分治Modified，考虑我们在用重心分治的时候会直接删掉新的部分与重心的连边，我们换一种方法，留下一个点记录所有被删除部分的信息，不过这个点不参与新一轮的分治。这样在分治的时候就可以直接计算了。

94.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n \log n)$

94.4 备注

那个比较逗的思路就是自己想出来的，智商堪忧（摊手）。

95 241E Flights

95.1 试题大意

给出一个n个点m条边有向无环图，初始时边权均为1，你需要将一些边的边权改为2，使得所有从1到n的路径的边权和均相等。

$1 \leq n \leq 1000, 1 \leq m \leq 5000$

95.2 算法讨论

考虑删掉那些不和$1, n$连通的点，那么这实际上是一个不等式模型，可以发现所有剩下的点到n的距离都固定，然后有边相邻的点有不等式限制，所以直接套用差分约束系统就行了，可以用队列大小大于$4(n+m)$判断有无负环。

95.3 时空复杂度

时间复杂度：$O(m)$
空间复杂度：$O(m)$

95.4 备注

为什么不能$n \cdot m$的复杂度做呀，直接设这个点与n的距离然后暴力转移，不过为何没发现有人这么写，弄得我都不敢写，是有什么bug么。

96 257E Greedy Elevator

96.1 试题大意

一栋大楼有m层，有这么一个电梯。初始时在第1层，电梯里是空的而且没有人在等电梯。有n个人会来乘坐电梯，第i个来乘坐电梯的人会在时刻t_i，于第s_i层上电梯（如果电梯不在他就会等着），在第f_i层下电梯。如果在这一时刻，电梯里有人在这一层下，这些人就会迅速下去；电梯所在的楼层有人在等电梯，这些人就会迅速乘上电梯。电梯每一个单位时间可以移动一层，它会按照如下算法移动：

- 如果电梯是空的并且没有人在等电梯，电梯会留在当前楼层。

- 否则，令电梯在第x层，令p_{up}为当前电梯里的并且目标楼层大于x的人和在大于x楼等电梯的人的总和，p_{down}同理。如果$p_{up} \geq p_{down}$，那么电梯会往上移动一层，反之往下移动一层。

你需要统计每个人下电梯的时间。
$1 \leq n \leq 10^5, 2 \leq m \leq 10^5$

96.2 算法讨论

我们只会在有人开始等电梯、上电梯或者下电梯时改变电梯的方向，那么我们把这些当做一个事件的发生，不难发现总共的事件个数是$O(n)$级别的，我们只需维护下一个事件的发生时间就可以按顺序处理事件回答询问了。对于有人开始等电梯，我们将所有人到达时间排个序就行了，上下电梯我们可以维护一个支持插入删除的有序表，根据当前p_{up}, p_{down}的大小定好方向，查询在有序表中上一个或下一个的位置。有个小细节就是不能直接记录$p_{up} - p_{down}$，因为要考虑电梯留在当前层的情况。

96.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n \log n)$

96.4 备注

注意时间可以被构造得很大，因此要用64位整数存储。也有一小部分细节需要注意。

97 268D Wall Bars

97.1 试题大意

要建这么一个奇怪的wall bars，以下是这东西的设计方案。

- 定义一个单位长度，中心是一个高度为n的杆子。
- 在高度为$1, 2, 3, ..., n$的地方恰好有一根水平的杆子伸出去，每根杆子都是四个预先定好的方向中的一个。

- 一个小孩能爬到一个与他现在高度之差不超过h的杆子，并且一旦从地面爬上第一根杆子之后就不允许换方向了，地面高度为0。
- 要求一个小孩从地面至少能爬到高度为$n-h+1, n-h+2, ..., n$其中的一根。

问有多少种方案设计合理。
$1 \leq n \leq 1000, 1 \leq h \leq min(n, 30)$

97.2 算法讨论

注意到h不超过30，这提示我们可以暴力记录这4个方向的之前一根杆子距当前高度的距离。这已经是$\binom{30}{4}$的状态数已经不是很大了，不过注意到转移起来非常麻烦。继续观察状态我们可以得到考虑前一个高度摆放的方向，这根棒子一定只离当前高度差1，唯一需要注意的只是从地面是否可以爬到这根杆子。所以这一维只需要记录0/1就行了。那么就有$f_{i,j\cdot k,a,b,c}$表示当前在高度i，高度$i-1$在方向j上放了一根杆子，并且是否能从地面爬上了的情况为k，剩下三个方向距离当前高度分别为a, b, c，但这样状态数稍微有点大有TLE的可能。接下来可以用一个比较有趣的技巧，我们在DP的时候不需要将每个状态具体对应到哪个方向，只需要体现出四个方向的不同，就能做到排列数的统计，因此j这个状态可以省掉，接下来的复杂度就不用担心了。

97.3 时空复杂度

时间复杂度：$O(n \cdot h^3)$
空间复杂度：$O(h^3)$

97.4 备注

我直接用$\binom{30}{4}$的状态去搞的结果被各种特殊情况整服帖了，老老实实去掉一维状态做。一道做了将近一下午的Div2D也是刻骨铭心啊（可见我现在真是渣）。

98 254D Rats

98.1 试题大意

有一张$n \times m$的网格图，有一些格子是墙壁，一些格子里是老鼠。现在需要把老鼠都杀掉，你可以在两个格子中投放毒气弹，如果一个格子有毒气，那么毒气会在下一秒扩散至和他有一条边相邻且不是墙壁的格子，毒气的持续效果是d秒（就是说从你放置毒气开始计时，第d秒之后所有的毒气全部消失）。如果一个有老鼠的格子里有毒气，那么老鼠将被立刻杀死。给出一组投放毒气的方案或者判断不可能。

$1 \leq n, m \leq 1000, 1 \leq d \leq 8$

98.2 算法讨论

竟然有$d \leq 8$这种条件简直不能忍啊。可以发现有老鼠的格子不超过$2 \times (d^2 + (d+1)^2)$，而可能放置毒气的格子就是每个老鼠周围距离不超过d的格子的并，这个并也不大，仔细思考一下可以发现是$2 \times ((2d)^2 + (2d+1)^2)$，这完全就可以暴力枚举两个可能放置毒气的点然后BFS就行了，也可以枚举老鼠然后用bitset来判断。需要注意的特殊情况是两个毒气的放置地点不允许相同，所以可能有一个毒气要放在没有用的位置，这个特殊情况可真够坑人的。

98.3 时空复杂度

时间复杂度：$O(d^4 + n^2)$
空间复杂度：$O(d^2 + n^2)$

99 269D Maximum Waterfall

99.1 试题大意

平面上有$n+2$条线段，第1条是在高度为t的$(-10^9, 10^9)$，第2条是在高度0的$(-10^9, 10^9)$，其他的线段位于高度为h_i的(l_i, r_i)，如果线段a能够连到线段b当且仅当：

- $h_a < h_b$。

- 线段(l_a, r_a)与(l_b, r_b)相交（包含也算）。
- 令他们相交的部分为(L, R)，在高度(h_a, h_b)的范围内(L, R)中都不能有任何线段。
- 这条边的边权为$R - L$。

求从1到2的一条路径使得最短的边最长，输出这个值。
$1 \leq n \leq 10^5$

99.2 算法讨论

很容易发现这是一个拓扑图，如果我们能够求出这个图来整个算法就只剩一个DP了。现在的关键是怎么建出这个图，这其实也很简单。考虑维护前i条线段构成的并，对于每个点只取最靠下方的线段上的点，那么我们新加入一条线段实际上只是将本来存在的一些线段删除之后加入一条新的，不难发现加入一条线段时最多新生成三个线段，每个线段会被删除一次，所以我们用set维护一个有序表每次暴力插入删除就行了。在删除的同时判断一下是否有边生成即可。细节比较恶心需要注意一下。

99.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n \log n)$

99.4 备注

本来以为要用线段树维护的，后面稍微仔细想想觉得没这个必要，set就行了。

100 346E Doodle Jump

100.1 试题大意

有n个站台，第i个站台的高度为$a \cdot i \bmod p$，如果$h_2 - h_1 \leq h$就能从高度为h_1的地方跳到h_2去。你初始时在地面上，高度为0。问你是否能够跳到最高的站台上去。有t组数据。

$1 \leq n, a, p \leq 10^9, 0 \leq h \leq 10^9, 1 \leq t \leq 10^4$

100.2 算法讨论

就相当于询问站台之间最大的间距是否不超过h，我们来仔细思考一下这个问题。

考虑我们将整个高度为p的部分按照长度a分段，不难发现前$\lfloor \frac{p}{a} \rfloor$段都是基本一样的（如果不一样我们用最后一段代替之前所有的段即可，答案不会改变，即将最后几个装上去的站台拆掉）。那么我们要处理的就是前面一个长度为a的段和之后一个长度为$p \bmod a$的一小段。我们把这两段拼在一起，也就是说$p' = a + a \bmod p$可以发现这种转换之后，站台的仍是每隔a设置一个，和原问题没有区别，答案也一致。虽然此时我们减小了一部分数据，不过还是过不了的。考虑现在a的值并没有改变，我们其实可以反过来安放站台，即从第n个站台每次往后减少a，也就是加上$p \bmod a$，这样一来我们有$a' = p \bmod a$，可以发现每经过两次缩小就至少能把a缩小一半，那么就靠谱了。

100.3 时空复杂度

时间复杂度：$O(t \log p)$
空间复杂度：$O(t \log p)$

100.4 备注

我做数学题能力一向很弱，可以做出这道题还是比较欣慰的。

101 264D Colorful Stones

101.1 试题大意

有两列彩色的石头，分别为a, b。每列石头的第一个石头上都站了一个人，你可以说出一些颜色，如果你说出的颜色和他们某个人脚下石头的颜色一致，那么他就会往前移动一步，站在下一个石头上（一共只有三种颜色的石头）。如果第一个人在第i块石头上，第二个人在第j块石头上，记录这个状态为(i, j)，问有多少种状态可以通过你说出的颜色指令到达。

$1 \leq |a|, |b| \leq 10^6$

101.2 算法讨论

对于第一个人站在石头上的每一个位置，我们可以求出第二个人最靠前和最靠后能够到达的位置l_i, r_i。很显然这之间的位置第二个人不一定能够到达，我们可以考虑具体哪些位置不能到达。有一个性质：如果第一个人在i，第二个人在j，并且$l_i \leq j \leq r_i$，这个状态不能到达当且仅当$a_i = b_{j-1}, a_{i-1} = b_j, a_i \neq b_j$，那么我们可以用一个前缀和数组记录其中不合法的个数，之后利用区间减法快速求出即可。

至于这个性质，可以将所有状态看做二维平面上的点，这个点不可达当前仅当这个条件满足，还是比较直观的。

101.3 时空复杂度

时间复杂度：$O(|a|)$
空间复杂度：$O(|a|)$

101.4 备注

这个性质很难发现，我并没有严格证明，官方题解里面是有证明的。

102 280D k-Maximum Subsequence Sum

102.1 试题大意

给你一个长度为n的序列a，你需要处理以下两种操作。

- 给定i, val，令$a_i = val$。
- 询问$a_l, a_{l+1}, ..., a_r$这段子序列中，至多k个子段的和最大是多少。具体的，你可以选出不超过k对$(x_1, y_1), (x_2, y_2), ..., (x_t, y_t)(l \leq x_1 \leq y_1 < x_2 \leq y_2 < ... < x_t \leq y_t \leq r, t \leq k)$使得$\sum_{i=1}^{t} \sum_{j=x_i}^{y_i} a_j$最大。如果$t = 0$，那么他们的和为0。

操作的个数不超过m。

$1 \leq n, m \leq 10^5, 1 \leq k \leq 20, |a_i| \leq 500$，保证询问（操作二）不超过10000个。

102.2 算法讨论

我们先不管修改和回答询问，考虑给出一段序列，如何选出k个子段使得他们的和最大。从特殊情况入手，如果$k = 1$，不妨令长度为i的前缀和为s_i，那么肯定就是找$\max(s_i - s_{j-1})(j \le i)$，这个记录一下最小值就能快速算出。再考虑选出的$k$段中一定有某一段起点为$j$，某一段终点为$i$，因为如果这两个任意一个不出现都可以通过替换将他们换进去。由于这两个点一定在最终答案中，不妨考虑如何添加新的段，不难发现只有三种选择：在j左边或者i右边重新添加一段，或者把$j..i$分成两段。我们以这些选择的代价把选择全都丢进一个大根堆里，每次找一个收益最大的，然后再新加入这个选择完成之后新的可能的三个选择。找出三个新的选择可以用线段树快速的查询，此题便迎刃而解了。

当然还有一种思考方式，这种方式更加直白。考虑我们用费用流来回答询问，由于模型十分简单而且k非常小我们直接模拟增广路的过程，也可以得到上面同样的解答过程。但这种方法的证明就十分显然了。（当然你得会证明费用流的正确性，而不仅仅是会使用）

102.3 时空复杂度

时间复杂度：$O(nk \log n)$
空间复杂度：$O(nk \log n)$

102.4 备注

比较有趣的中国人题吧，如果外国人出题一定不会把线段树硬生生的套上去的。并且这题在清澄上出题人竟然少给了一个条件，需要卡一点常数。

103 293D Ksusha and Square

103.1 试题大意

给出一个凸多边形，随机取在多边形内的两个不同的整点（横纵左边均为整数的点），用这两个点作为对角线的端点画一个正方形，求这个正方形面积的期望。

$1 \le n \le 10^5, |x_i|, |y_i| \le 10^6$

103.2 算法讨论

稍微脑补一下，要求的东西大概就是：

$$\frac{\sum_a \sum_b \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}^2}{total\_pairs\_of\_points}$$

很容易发现我们可以把横纵两维拆开考虑，另凸多边形中在$x = a$上的点有c_a个，那么我们要求

$$\sum_i i^2 \sum_j c_j \cdot c_{j+i}$$

这样看这似乎是一个卷积，当然不能这么算。

$$\sum_j c_j \sum_i (i-j)^2 \cdot c_i$$

考虑j从小到大，如何维护后面那一部分。虽然他是一个二次函数，看上去不好维护，不过差分一下就变成一次的了。看上去仍然需要用线段树维护实际上不然，我们直接对差分继续进行差分，这样可以变成常数，然后就可以直接做了。

103.3 时空复杂度

时间复杂度：$O(\max|x_i|)$
空间复杂度：$O(\max|x_i|)$

103.4 备注

还以为要用FFT呢，还有要注意是不同的点，计算期望的时候要注意。

104 306C White, Black and White Again

104.1 试题大意

你的生活总是满足“一些好事，一些坏事，再一些好事”的规律。在接下来的n天你的生活里会发生w件不同的好事和b件不同的坏事，每天至少

发生一件事，并且一天内要么全是好事要么全是坏事。所以未来n天会先有若干天发生好事，然后若干天发生坏事，再有若干天发生好事（若干> 0）。要求统计这n天事件发生的方案数（一天内的事件发生的顺序不同也算作不同）。输出答案对$10^9 + 9$取模。

$1 \leq n, w, b \leq 4000$

104.2 算法讨论

一个简单DP题。不妨先枚举坏事发生的天数i，可以发现我们并不需要分开考虑左右两端的情况，只需计算将w件好事分$n - i$天发生和将b件坏事分i天发生的方案数乘起来再乘上将他们合并的方案数即可。至于将a件事分b天发生我们直接DP，令$f_{a,b}$为方案数，转移的话枚举最后一维怎么放就行了。

104.3 时空复杂度

时间复杂度：$O(n^2)$
空间复杂度：$O(n^2)$

105 317C Balance

105.1 试题大意

给出一张n个点无向图，每个点有存有一些水，你可以通过m条边来调整水量，具体的，如果一条边连接x, y，在一次操作中你可以将若干水量从x调到y或者从y调到x。但任何时候必须保证每个点的水量不超过v，初始时每个点的水量为a_i，目标状态时该点的水量为b_i。求一个操作次数不超过$2 \cdot n^2$的方案，从初始状态到目标状态，或者判断出无解。

$1 \leq n \leq 300$

105.2 算法讨论

考虑每个点的水量限制都是相同的v，那么这题就非常好做了。求出原图的一个生成树，然后对其进行DFS，在回溯时对这个点进行暴力调整，因为水量上限相同因此肯定存在一种调整方案，并且每个点可以用到n次操

作是不会超过上限的。无解的情况当然就是对于某个连通块初始水量之和和目标水量之和不等就行了。

105.3 时空复杂度

时间复杂度：$O(n^2)$
空间复杂度：$O(n^2)$

105.4 备注

条件比较宽松的构造题吧，主要是利用生成树保证操作次数以及考虑到水量限制相同所以可以直接暴力。

106 294D Shaass and Painter Robot

106.1 试题大意

给出一个$n \cdot m$的网格，初始时每个格子都是白色的。上面有一个机器人，位于(x, y)，并且有一个初始方向（左上，左下，右上，右下其中之一），并且机器人会一直朝着这个方向走，到达边界是会依据光线的反射定律改变方向继续行走。经过的格子都会被染黑，当网格被染成黑白相间之时机器人会立马停止行动，问他一共走了多少步，或者判断他会一直走下去。

$1 \leq n, m \leq 10^5$

106.2 算法讨论

考虑对于基本上每一斜行或者一斜列机器人都只会经过一次，我们仍然可以暴力模拟他经过的路径。比较麻烦的是如何判断当前网格已经被染成黑白相间，我们脑补一下整个染色的过程，其实大概是：从起点开始→到达四个角的一个→反向之后再回到起点→到四个角的另外一个→结束。因此我们用一个set把经过的斜行斜列的端点都记录下来，如果所有端点都被经过了那么肯定就已经被染成黑白相间的了。剩下的问题只剩暴力模拟应该是很简单的了。

106.3 时空复杂度

时间复杂度：$O(n \log n)$
空间复杂度：$O(n \log n)$

106.4 备注

Div2-D差不多就只有这个难度。

107 319D Have You Ever Heard About the Word?

107.1 试题大意

定义循环串为一个字符串a，并且a可以写成$bb...b$至少两个b拼在一起，并且称b是a的循环节。给出一个字符串s，每次在s的子串中找一个最短的循环串，对于长度相同的选取起点最靠前的一个，然后将这个子串替换成他的循环节，直到不能操作位置。求最终的字符串。

$1 \leq |s| \leq 50000$

107.2 算法讨论

嘛，一个很暴力的题，敢想敢做。

考虑我们从小到大枚举循环节的长度i，然后删除所有长度为$2i$的循环串之后再将字符串暴力重建，这么暴力的方法其实只会枚举到$\sqrt{n}$个不同长度的循环节。对于判断是否存在一个长度为$2i$的循环串我们可以利用求k重复子串的经典方法，将字符串每i个分为一段，就会分成$\frac{n}{i}$段，每一段用hash判断相邻两段前缀和后缀能够匹配的最大范围（其实算前缀就足够了），如果能够匹配则暴力删除。这样判断的时间复杂度是$O(n \log n)$的。

107.3 时空复杂度

时间复杂度：$O(n\sqrt{n})$
空间复杂度：$O(n)$

107.4 备注

两种看起来会TLE的算法其实均摊下来都没有问题。

108 314E Sereja and Square

108.1 试题大意

对于一个字符串，定义匹配为从前往后每一个小写字母往后匹配一个未被匹配的它对应的大写字母。定义该字符串合法为所有字母都有匹配并且所有匹配都不相交。给出一个长度为n的字符串，一些位置上是小写字母，其他的位置上没有填字母。你需要给没有填字母的位置填入字母使得字符串合法，问方案数（不允许使用字母x）。输出答案对2^{32}取模。

$1 \leq n \leq 10^5$

108.2 算法讨论

哈哈哈，没想到吧，此题是n^2的。

这其实是一个括号匹配，在新建左括号的时候有25种方案而已。而这题的正解是直接$O(n^2)$暴力DP，十分简单。

108.3 时空复杂度

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

108.4 备注

怨念满满啊，为什么$n = 10^5$也可以n^2做呀。。。不过需要注意的一点是需要用unsigned int自然溢出，写取模操作肯定是会TLE的了。

109 321D Ceil and Flipboard

109.1 试题大意

给出一个n行n列的表格（保证n是一个奇数），每个格子里初始时有一

个数。你每次可以选择一个大小为$x \cdot x(x = \frac{n+1}{2})$的部分，将部分中所有的数都取反。你可以操作任意次，最大化所有元素之和，并输出这个结果。

$1 \leq n \leq 33$

109.2 算法讨论

非常有趣的一道题。

首先我们需要一个巨大的“观察”，这个操作有一个非常重要的性质。令$P_{i,j}$表示第i行j列的格子是否被取反，那么下面两个等式恒成立。

$$P_{i,j} \otimes P_{i,x} \otimes P_{i,x+j} = 0, \quad \forall\ 1 \leq i \leq n, 1 \leq j < n$$
$$P_{i,j} \otimes P_{x,j} \otimes P_{x+i,j} = 0, \quad \forall\ 1 \leq i < x, 1 \leq j \leq n$$

正确性比较显然，因为任意一个$x \cdot x$的正方形要么同时覆盖2个，要么没有覆盖。

这样我们只需知道左上角大小为$x \cdot x$部分的$P_{i,j}$，就能推出所有的P了。而事实上我们可以通过操作构造出左上角$x \cdot x$部分的任意一种方案。也就是说对于任意一种左上角大小为$x \cdot x$方格内的取值都能得到，而我们又可以通过上面两个等式更加方便的算出其余部分的取值。

我们不妨固定所有的$P_{i,x}, P_{x,i}(1 \leq i \leq x)$，这样剩下的$(x-1) \cdot (x-1)$的格子就可以直接贪心取值了。但需要固定的数仍是$2x$级别，无法暴力枚举，更仔细一点的观察之后不难发现，我们只需固定$P_{i,x}$，然后列和列之间就已经没有影响了，我们逐列贪心即可。

到此问题便圆满解决。

109.3 时空复杂度

时间复杂度：$O(2^x)$

空间复杂度：$O(2^x)$

109.4 备注

这题虽然之前做过一遍，但再来做一次的时候竟然还是不会做，想了很久才想出来，非常有趣。

第四部分　USACO试题范做

110　US Open 10　Triangle Counting

110.1　试题大意

给出n个笛卡尔坐标系上的整点，统计有多少三角形包含原点(0,0)。

不会有两个点连线所在的直线经过原点。

$1 \leq n \leq 10^5$，坐标范围是$[-10^5, 10^5]$

110.2　算法讨论

最暴力的方法，用$O(n^3)$的时间复杂度暴力枚举所有的三角形，再对于每个三角形判断是否包含原点。

接下来需要考虑的是如何优化这个算法。考虑已经枚举了两个点x, y，第三个点的选取范围实际上是两个半平面的交，因为半平面的边界都经过原点，不妨将半平面转化成极角，也就是统计一个极角区间里有多少点。但有n^2个极角区间需要统计，仍然不能通过本题。不妨换一种枚举顺序，将区间拆开考虑，统计每个点作为了多少个区间的左端点以及多少个区间的右端点，最后统计答案。这样的复杂度是$O(nlogn)$，但代码实现比较复杂。

为了让实现过程变得更方便，不妨采用补集转化的思想，即统计有多少不包含原点的三角形，再用总三角形数减去它即得到答案。不包含原点的三角形个数非常容易统计，考虑再枚举了一个点之后，剩下的两个点的选择区域都是在这个点与原点的连线形成的半平面的一侧，并且两个点的选取并不互相影响，可以直接利用组合数计算。而总三角形数直接就是$\binom{n}{3}$。虽然在时间上没有优势，但这个基于补集转化思想的算法主程序几乎只由排序和two points构成，非常方便。

至此问题便圆满解决。

110.3　时空复杂度

时间复杂度：$O(n \log n)$

空间复杂度：$O(n \log n)$

110.4 备注

来自自己写的题解。