IOI2015中国国家集训队第一次作业
试题泛做

宁波市镇海中学 邹逍遥

Contents

1 Codeforces

1.1 Codeforces 235 C

Description

给出一个长度为$n(n \leq 10^6)$的字符串s，同时有$q(q \leq 10^6)$个询问，询问的总长度不超过10^6。

每次我们询问对于给定的串q，s有多少个字串和它循环同构。

Analysis

首先我们建出s的后缀自动机，然后对于每个询问，我们找出q对应的节点x。然后利用后缀自动机在均摊$O(1)$ 的时间内在结尾添加字母并在开头删除字母重复$|q|$次即可统计出数量。

时间复杂度：$O(n + |\Sigma||q|)$

空间复杂度：$O(n|\Sigma|)$

1.2 Codeforces 235 D

Description

给出一个$n(n \leq 3000)$个点n条边的连通图，对它执行一个算法：

- 将这个图的大小加入总代价中。
- 随机选择一个点删去并递归处理各个连通子图。

求总代价的期望。

Analysis

定义$E_{i,j}$为当i被删去时j和i连通的概率，那么答案就等于$\sum_{i,j} E_{i,j}$。

当给出的图为一棵树时$E_{i,j}$就是i到j这条路径上i是第一个被选择的，可以看出$E_{i,j} = \frac{1}{dis_{i,j}+1}$。

当给出的图为环+外向树时就会有两条路径，使用容斥原理计算即可。

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

1.3 Codeforces 235 E

Description

求$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c}d(ijk)$
$a,b,c <= 2000$

Analysis

令

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

$$g(a,b,c)=\sum_{gcd(i,k)=gcd(i,j)=gcd(j,k)=1}[\frac{i}{a}][\frac{j}{b}][\frac{k}{c}] \tag{2}$$

首先我们来证明 $f(a,b,c)-f(a-1,b,c)-f(a,b-1,c)-f(a,b,c-1)+f(a-1,b-1,c)+f(a-1,b,c-1)+f(a,b-1,c-1)-f(a-1,b-1,c-1)=g(a,b,c)-g(a-1,b,c)-g(a,b-1,c)-g(a,b,c-1)+g(a-1,b-1,c)+g(a-1,b,c-1)+g(a,b-1,c-1)-g(a-1,b-1,c-1)$

将等式左右化简得左边$=d(abc)$。

右边$=\sum_{gcd(i,k)=gcd(i,j)=gcd(j,k)=1}([\frac{i}{a}]-[\frac{i}{a-1}])([\frac{j}{b}]-[\frac{j}{b-1}])([\frac{k}{c}]-[\frac{k}{c-1}])$

即$\sum_{i,j,k}[gcd(i,k)=gcd(i,j)=gcd(j,k)=1][a\%i=b\%j=c\%k=0]$

对于任意一个质数p，设x是最大的使p^x整除a的x，y和z的定义类似。那么：

等式左边p的出现次数为$x+y+z+1$(0到$x+y+z$的任意一个数均可)

等式右边p的出现次数也为$x+y+z+1$($(0,0,0)$或$(1\sim x,0,0)$或$(0,1\sim y,0)$或$(0,0,1\sim z)$)

证明了这个等式以后我们可以利用类似递推的方法来证明$f(a,b,c)=g(a,b,c)$这里不再赘述。

利用容斥原理，我们可以将$g(a,b,c)$化为

$$\sum_{i=1}^{a}\sum_{x=1}^{\min(b,c)}[\gcd(i,x)==1]\mu_x h(\frac{x}{b},i)h(\frac{x}{c},i)\frac{i}{a}$$

其中$h(i,j)$表示在$1..i$中与j互质的数的个数。

设$n=\max(a,b,c)$，假如每次暴力计算h那么时间复杂度为$O(n^2\log n)$空间复杂度为$O(n)$。假如预先计算好所有的h值（只有$O(n\sqrt{n})$种）那么时间复杂度为$O(n^2+n\sqrt{n}\log n)=O(n^2)$空间复杂度为$O(n\sqrt{n})$。当然这两种做法都能通过全部数据。

时间复杂度：$O(n^2)$或$O(n^2\log n)$

空间复杂度：$O(n\sqrt{n})$或$O(n)$

1.4 Codeforces 238 D

Description

给定一种编程语言：这种语言只由0123456789<>组成。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。重复以下操作直到指针指向串外：

- 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则删除这个数字，串的长度减一；
- 如果指针指的位置是<或>，那么指针的移动方向对应得改为向左或向右（与符号的尖角方向相同），接着指针沿着新的移动方向移动。如果新的位置也是< 或>，则删除原来的字符。

任何时刻如果指针指向了串外，程序就结束运行。 现在我们有一个长度为$n(n \leq 10^5)$的程序，你需要回答$q(q \leq 10^5)$个询问。每个询问会给你两个数l,r，如果把$s_l...s_r$看成一个单独的程序，问你每个数字会被输出多少次。

Analysis

首先我们将整个程序模拟一遍，并且每次程序结束后假如仍有字符剩下就从第一个字符开始再模拟一次，直到没有字符为止。记下每个时刻总共输出了多少数字，每个字符在哪些时刻被访问到，每个字符第一次被访问的时候它的左边是哪个字符。

这样我们处理询问的时候只需找到左端点第一次出现的时间t，然后在左端点左边的字符和右端点右边的字符中二分出结束时间。知道了开始时间和结束时间就可以通过前缀和相减来算出中间一段时间的输出情况。

时间复杂度：$O(n + m \log n)$

空间复杂度：$O(n)$

1.5 Codeforces 240 F

Description

给定一个长度为$n(n \leq 10^5)$小写字母组成的字符串，有$m(m \leq 10^5)$个操作，每个操作包含两个数l,r，假如$s_{l...r}$可以重排成回文串就将这个子串重排成字典序最小的回文串，否则不操作。问最后这个串长什么样。

Analysis

检测是否可重排只需要检测是不是至多一个字母的数量是奇数即可。重排后一定是如下形式：“$aaabbbcccddd \cdots zzzXzzz \cdots dddcccbbbaaa$”，只

需要修改$O(|\Sigma|)$个区间即可。可以开$|\Sigma|$个线段树来维护每个位置是否有这个字母，这样就可以在$O(|\Sigma|\log n)$的时间内求出每个字母的数量并在$O(\log n)$的时间内覆盖某个区间。总复杂度$O(|\Sigma|n\log n)$。

时间复杂度：$O(|\Sigma|n\log n)$

空间复杂度：$O(|\Sigma|n)$

1.6 Codeforces 241 B

Description

给出$n(n \leq 50000)$个数，每个数不超过10^9，要求选出$m(m \leq \frac{n(n-1)}{2})$对数使得数对之间两两不同且数对异或后的和最大。

Analysis

首先考虑暴力做法：求出所有数对的异或值，排序后把最大的m个加起来。这是$O(n^2\log n)$的。

假如我们二分要选出的最小的数，问题就转化为了：有多少对数异或后不小于这个数。可以维护一棵线段树，然后枚举每个数，就变成了求线段树上$O(\log n)$个节点的和。加上二分复杂度就是$O(n\log^2 n)$。当然也可以不二分，直接在n棵线段树上同时走，可以优化到$O(n\log n)$。

现在我们已经知道了要选取的最小的数，问题变成了：异或后不小于这个数的对数之和。那么我们在线段树上每个节点记下每一位的个数，统计的时候按位枚举加入答案即可。复杂度$O(n\log^2 n)$。

时间复杂度：$O(n\log^2 n)$

空间复杂度：$O(n\log n)$

1.7 Codeforces 241 E

Description

给定一个$n(n \leq 1000)$个点$m(m \leq 5000)$条边的无向图，并保证存在一条从1到n的路径。你需要给每条边确定一个1或2的权值，使得所有从1到n的路径长度都相等。或者输出无解。

Analysis

因为所有1到n的路径长度相等，所以有可能在1到n路径上的点也都满足这个性质。那么只需要确定1到这些点的路径长度即可，即给每个点确定一个类似“势能”的概念。

考虑枚举所有1到n路径可能经过的边，对于这两个点互相更新对方的势能。由于更新的时候只会使势能减小，那么有解的情况一定会在$O(n)$次更新后停止。或者可以列出m条不等式使用差分约束系统求解。

时间复杂度：$O(nm)$

空间复杂度：$O(m)$

1.8 Codeforces 241 F

Description

给出一个人在$m \times n(m, n \leq 100)$网格图中的行走轨迹，求时刻$k(k \leq 10^5)$他的位置。轨迹给出方式：首先给出$t(t \leq 1000)$个点，保证任意两个相邻点在同一水平线上或在同一竖直线上并且之间没有障碍物。这个人行走时从第一个点出发，每次以下一个点为目标笔直行走直到走到为止。

Analysis

对于每一步可以先枚举方向找到下一个点的位置然后模拟。由于题目的数据规模非常小这完全是可以承受的。

时间复杂度：$O(n|t|)$

空间复杂度：$O(mn)$

1.9 Codeforces 243 C

Description

一个人在一个非常大的网格图上行走。一共走了$n(n \leq 1000)$步，每步可以向四个方向中的一个连续行走任意步。走过的网格会被染色。现在你从边界进入，不能经过染色的格子。问有多少个格子是你不能达到的。

Analysis

将坐标离散，然后从边界开始floodfill一遍，把没有遍历到的格子加起来即可。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

1.10 Codeforces 243 D

Description

给你一个$n \times n(n \leq 1000)$的网格，每个格子上堆叠有一定数量的小立方体$(\leq 10^9)$。求从无穷远处以向量$v = (v_x, v_y, 0)$看过来能看到多少小立方

体。

Analysis

考虑先将小方块的投影求出，然后按和观望点的距离从近到远加入，那么新加入一个的时候需要知道投影的宽度内当前最低的方块的高度，然后更新这个区域的最大值。那么答案就是$\sum \max(v_x - \min(l_x, r_x), 0)$。使用线段树维护即可。

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n^2)$

1.11 Codeforces 248 E

Description

有$n(n \leq 10^5)$个架子，每个架子上有$a_i(a_i \leq 100)$个满的罐子。接下来有$m(m \leq 10^5)$次操作，每次操作可以从第x_i 个架子上随机拿下$k_i(k_i \leq 5)$个罐子并且在第y_i 个架子上放上k_i个空罐子。问每次操作完后没有满罐子的架子数量的期望。

Analysis

用$f(i, j)$表示第i个架子上有j个满罐子的期望。那么只要维护$\sum f(i, 0)$即可。

由于满罐子的数量是不会增加的，所以需要存储的量只有$O(\max(a_i) \times n)$个。每次更新的时候枚举j用$f(x, j)$ 来更新$f(x, j-1)$即可。

时间复杂度：$O(n + \max(a_i) \times \sum k_i)$

空间复杂度：$O(\max(a_i) \times n)$

1.12 Codeforces 249 D

Description

给定平面上的$n(n \leq 100000)$个点，你刚开始在原点，每次可以走到右上方的任意一个点使得当前点和目标点的连线的斜率处于给定的区间内。问最多能走几步。

Analysis

首先把坐标转换一下使得能走到的点就是x, y坐标都大于它的点。然后就转化成了找一些点使得每个点的两个坐标都比前面那个点大。按x坐标排序后就变成了经典的最长上升子序列问题，可以在$O(n \log n)$的时间内解决。

时间复杂度：$O(n \log n)$
空间复杂度：$O(n)$

1.13 Codeforces 249 E

Description

给定一个如下图所示的无穷大矩阵，有$t(t \leq 10^5)$组询问，每次询问一个矩形内的数的和。答案大于10^{10}则只需输出后10位。

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

Analysis

首先查询$(x1, y1, x2, y2)$等价于$(1, 1, x2, y2) - (1, 1, x1-1, y2) - (1, 1, x2, y1-1) + (1, 1, x1-1, y1-1)$，所以只需要支持查询前缀和即可。

首先左上角的正方形（即边长为$\min(x, y)$的正方形）答案是$\frac{(x^2+1)x^2}{2}$。假如$x \leq y$那么剩下部分的和就是$x \sum_{i=x}^{y-1} i^2 - \frac{(x-y)*x*(x+1)}{2}$否则剩下的和就是$y \sum_{i=y+1}^{x} - \frac{(x-y)*y*(y-1)}{2}$。$\sum i^2$的计算可以利用$\sum_{i=1}^{x} i^2 = \frac{x*(x+1)*(2x+1)}{6}$。可以使用高精度也可以选择模两个质数看是否相等来判断答案是否超过10^{10}。

时间复杂度：$O(t)$
空间复杂度：$O(1)$

1.14 Codeforces 253 E

Description

你有一个打印机要完成$n(n \leq 50000)$个任务，每个任务有起始时间，打印所需时间和优先级。每个时刻打印机总是会做优先级最高的任务。现在有一个任务的优先级未知，但是知道了它完成的时刻。你需要找出它的一个可行的优先级和这时其他任务的完成时刻。保证有解。

Analysis

假如已经知道了这个任务的优先级，那么我们可以按时间顺序加入任务并维护一个优先队列，这样就可以在$O(n \log n)$的时间内求出完成时间。

注意到优先级更高显然不会更晚完成，所以完成时间关于优先级单调不增。于是我们可以采用二分的方法求出可行的优先级。

时间复杂度：$O(n \log^2 n)$

空间复杂度：$O(n)$

1.15 Codeforces 254 D

Description

给你一个$n \times n (n \leq 1000)$的网格，每个格子里可能有墙或者老鼠。你可以在空地上放两颗炸弹，炸弹可以炸到不经过墙的四联通距离不超过$k (k \leq 8)$的格子。求一种方案炸死所有老鼠或者输出无解。

Analysis

首先在能炸到第一只老鼠的位置枚举第一颗炸弹的位置，如果这时候都被炸完了则第二颗随便放，否则找到一只没被炸到的，在能炸到它的范围内枚举第二颗炸弹的位置，最后检查方案。由于老鼠的数量超过$O(k^2)$可以直接输出无解。所以老鼠的数量是$O(k^2)$的，总复杂度就是$O(k^6)$完全可以承受。

时间复杂度：$O(n^2 + k^6)$

空间复杂度：$O(n^2 + k^4)$

1.16 Codeforces 256 D

Description

有$n (n \leq 256$且保证n是2的次幂$)$个人，每个人可以说有a_i个人说真话（a_i必须在1到n之间），问有多少种方案使得你可以断定恰好$k (k \leq n)$个人一定在说谎。

Analysis

首先解决给出方案的情况下如何求出断定说谎的人数。由于所有说同一个数字的人是等价的，所以只可能全部是或者全部不是。容易发现假如说i的人有j个，那么当且仅当$i = j$时可能满足条件，否则这些人一定在说谎。

那么用$f(i, j, k)$表示当前枚举到i个人的情况，前j个人中有k个肯定说谎的方案数，那么容易推出转移方程$f(i, j, k) = f(i - 1, j - i, k) + \sum f(i -$

$1, j-x, k-x)(i \neq x)$。其中i的这一维可以利用滚动数组优化掉。这个方法的时间复杂度是$O(n^4)$，空间复杂度是$O(n^2)$，还不足以通过本题。但由于题目保证n是2的次幂，所以可能的询问数量非常少，所以可以打表预处理。

时间复杂度：$O(1)$
空间复杂度：$O(1)$

1.17 Codeforces 257 E

Description

在一栋高为$m(m \leq 10^5)$的楼房上有一座电梯，有$n(n \leq 10^5)$个人，第i个人会在$t_i(t_i \leq 10^0)$时刻在x_i层出现，目的地是y_i层。每个时刻假如有人在电梯内或有人在等电梯，电梯会向人多的一边（里外需求之和）移动一层，假如一样多就向上。移动完后人可以进出电梯，时间忽略不计。现在让你模拟电梯的运作并输出每个人到达目的地的时间。

Analysis

首先我们可以记下上面人数和下面人数并实时更新，所以每一步的决策都可以$O(1)$确定。注意到假如过程中没有人上下电梯并且没有新的人出现那么电梯的决策不变。所以我们可以用两个set维护电梯里的人目标位置和电梯外的人所在位置。这样可以在$O(\log n)$的时间内求出下一个下电梯人的时间和下一个上电梯人的时间，然后和下一个新出现人的时间取最小值就可以算出下一步的位置。注意到每次都有至少一个人跨越了一个阶段，所以运作的次数不超过$3n$。

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

1.18 Codeforces 258 D

Description

给出一个长度为$n(n \leq 1000)$的序列，有$m(m \leq 1000)$次操作，每次操作有50%的概率交换给定的两个数(a_i, a_j)。 问最后逆序对的期望。

Analysis

我们可以动态维护每个操作后的任意两个数之间的大小关系。简单地说就是维护a_i比a_j大的概率。

一开始时显然这个概率就是$[a_i \geq a_j]$。每次操作后和a_i, a_j无关的数概率显然不受影响。那么我们只需要重新计算那些和a_i, a_j有关的数对的概率即可。

时间复杂度：$O(mn)$

空间复杂度：$O(n^2)$

1.19 Codeforces 260 E

Description

给定平面上的$n(n \leq 10^5)$个点，要求画四条直线（横着两条竖着两条）把这些点划成9份（直线不能经过点）。使得每份里的点的数量经过重排后与给出的9个数一致。

Analysis

首先9!枚举每一块内的点数，然后算出四条直线的坐标并检查是否每一块都符合要求。这就转化成了一个经典的二维数点问题。做法有很多：（在线）树套树$O(\log^2 n)$，（离线插入+在线查询）函数式线段树$O(\log n)$，（离线插入+在线查询）树状数组套vector+二分查找$O(\log n)$，（离线插入+离线查询）扫描线+树状数组$O(\log n)$。

时间复杂度：$O(9! \log n)$

空间复杂度：$O(n \log n)$

1.20 Codeforces 261 D

Description

给出一个长度为$n(n \leq 10^5)$序列，序列中的元素均小于$maxb(maxb \leq 10^5, n \times maxb \leq 2 \times 10^7)$，问这个序列重复$t(t \leq 10^9)$次后的最长上升子序列的长度。

Analysis

当$t \geq maxb$时，答案显然是$maxb$。所以只需要考虑$t \leq maxb$的情况。

然后我们对于每一位算出到这一位时，以某个数值i为结尾的最长上升子序列的长度f_i。我们维护f_i的前缀最大值g_i，显然g_i单调不降。所以需要更新的g_i是连续的一段。我们可以在每碰到一位时就暴力更新所有需要更新的g_i，即不停地向后找直到找到不能更新的为止。由于$g_i \leq n$，且$i \leq maxb$，所以更新的次数不会超过$n \times maxb$。

时间复杂度：$O(n \cdot maxb)$

空间复杂度：$O(n + maxb)$

1.21 Codeforces 261 E

Description

给你一个计算器，它有两个变量a,b。初始时$a = 1, b = 0$。每次操作可以$b = b + 1$或者$a = a * b$。问进行$p(p \leq 100)$次操作后能得到几个l和$r(r \leq 10^9)$之间的数。

Analysis

通过打表发现10^9内最大质因子不超过p的数只有不超过3000000个。我们可以先把这些可能满足条件的数全部求出来。然后从小到大枚举结束的时候b的大小，并动态维护每个数最少需要乘几次。即：枚举到i时，对于每个数x，用$f_{\frac{x}{i}} + 1$去更新它。利用two pointers维护可以做到$O(N)$。假如在某个b时，一个数的操作次数$+b \leq p$的话这个数就是可以生成的。

时间复杂度：$O(Np)$（N为10^9内最大质因子不超过p的数的个数）

空间复杂度：$O(N)$

1.22 Codeforces 263 E

Description

给定一个$n \times m(m, n \leq 1000)$的矩阵和正整数$k$，定义$f(x, y) = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j} \cdot \max(0, k-|i-x|-|j-y|)$，$mval =$满足$k \leq x \leq n-k+1, k \leq y \leq m-k+1$的最大的$f(x, y)$。要求输出任意一组$(x, y)$满足$k \leq x \leq n - k + 1, k \leq y \leq m - k + 1, f(x, y) = mval$。

Analysis

要求的值可以分成k个子矩形的和，所以通过预处理前缀和就可以在$O(k)$的时间内算出任意一个$f(x, y)$的值。这样可以做到$O(n^3)$，已经可以通过全部测试数据。

稍加观察可以发现，由于这些子矩形的位置关系非常特殊，在计算的时候可以分成四部分，每一部分都是前缀和中连续的一段。所以可以通过预处理前缀和的"/"方向和"\"方向的连续k个的和就可以做到$O(1)$计算$f(x, y)$。

时间复杂度：$O(mn)$

空间复杂度：$O(mn)$

1.23 Codeforces 264 E

Description

给定两个只含RGB长度不超过10^6的串，两个人一开始分别站在每个串的第一个字母上。每次你可以喊一个字母，然后两个人看自己脚下是不是这个字母，如果是就向前走一步。如果这么做会使某个人走出边界那么就不能喊这个字母。问有多少种状态是可达的。

Analysis

设第一个人在串a的第x格，第二个人在串b的第y格，设$A = a_{1\cdots x-1}, B = a_{1\cdots x}, C = b_{1\cdots y-1}, D = b_{1\cdots y}$。那么假如$D$是$A$的子串或者$B$是$C$的子串那么显然不可达。

那么假如D不是A的子串且B不是C的子串，那么我们来试着构造方案。

- 假如$a_1 = b_1$那么只要喊a_1那么达到的状态还是满足条件的状态。
- 假如$a_1 \neq b_1$且$a_{2\cdots x}$不是$b_{1\cdots y-1}$的子串那么只要喊a_1那么达到的状态还是满足条件的状态。
- 假如$a_1 \neq b_1$且$b_{2\cdots y}$不是$a_{1\cdots x-1}$的子串那么只要喊b_1那么达到的状态还是满足条件的状态。
- 假如$a_1 \neq b_1$且$b_{2\cdots y}$是$a_{1\cdots x-1}$的子串且$a_{2\cdots x}$是$b_{1\cdots y-1}$的子串那么这种状态就是不可行的。

假如满足第四个条件，那么两个串长度一定相等，且A串形如ABABAB，B串形如BABABA。即所有形如????AB和????BA的在算到最后两位的时候都会不可行。所以只需再排除掉这一种即可。

实现的时候可以枚举第一个串的位置，然后利用two pointers维护第二个串可行的最左点和最右点。计算答案的时候再减去区间内满足连续两个字母和枚举到的位置的连续两个字母反序的个数即可。这个可以先求一遍前缀和来快速求出。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

1.24 Codeforces 267 C

Description

给定一张$n(n \leq 100)$个点$m(m \leq 5000)$条无向边的图，求一个从1到n最大的网络流使得从1到n的每条路径的流量和都相等，并输出方案。

Analysis

由于从1到n的每条路径的流量和都相等，所以对于从1到n的每个可能经过的点p，从1到p的每条路径的流量和都相等。于是只需要给每个点确定一个势能，流量就是相邻两点的势能差。首先将1和n的势能设为常数，用高斯消元解出每个点的势能，之后根据边的流量进行缩放即可。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

1.25 Codeforces 268 D

Description

要修建一个高度为$n(n \leq 1000)$的东西，每个高度上都有且仅有一根杆子朝向四个方向中的一个问有多少种修建方案使得可以从四个方向中的某一个方向以$h(h \leq 30)$的步长（即每次只能走和当前杆子距离不超过h的）往上爬到顶。

Analysis

考虑动态规划。用$f_{i,u,i,j,k}(i \leq n, u \leq 1, i, j, k \leq h)$表示爬前$i$根杆子，能不能从某个确定的主方向到达，另外三个方向的上一根杆子的高度差按顺时针分别是i, j, k。转移的时候每个状态可以转移到$i+1$的四个状态（分别以每个方向为主方向。）使用滚动数组可以将空间优化到$O(h^3)$。

时间复杂度：$O(nh^3)$

空间复杂度：$O(h^3)$

1.26 Codeforces 269 D

Description

给定一个平面上的$n(n \leq 10^5)$条水平线段，保证线段之间没有公共点。线段i能到线段j当且仅当：

- $\max(l_i, l_j) < \min(r_i, r_j)$
- $h_j < h_i$
- 不存在$k(h_i \leq h_k \leq h_j)$使得i, k和k, j都满足以上两个条件。

求一条顶部到底部的路径使得相邻两块板之间的相交部分中的最小值最大。

Analysis

考虑从上向下扫描。维护当前从底部看到的线段。由于新加进一条线段最多使线段条数+2，所以线段数为$O(n)$。又因为需要扫描到的线段都会被删除，所以均摊复杂度也是$O(n)$。查询一次线段位置的复杂度是$O(\log n)$。维护的总复杂度就是$O(n \log n)$。可以用STL实现。

维护出了上面的线段的情况以后就可以更新答案了。考虑这条线段的区间内有k条线段从左到右为$a_1 \cdots a_k$，那么分析一下哪些情况是可以转移过来的。

- $k = 1$。这时候整条线段都被上面的线段覆盖且中间没有其他线段。
- a_1的右端点就是它对应的线段的右端点。容易看出中间不含任何线段。a_k同理。
- a_i对应的线段就是a_i。这时候整条线段都被这条线段看见。

显然这三种情况包含了所有的情况（上包含下，下包含上，下左上右，下右上左）。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

1.27 Codeforces 273 D

Description

问一个$m \times n (m, n \leq 150)$的棋盘上有多少种染色方案使得任意两个被染色的格子之间都连通且只通过染色格子的距离等于$|x1 - x2| + |y1 - y2|$。

Analysis

容易发现每一行和每一列的染色格都必须是连续的（除非没有一个格子被染色）。那么每一行的左边界和右边界就是一个单峰函数，即先上升后下降。

考虑动态规划：$f(i, j, k, left, right)$表示在$i * n$的棋盘上第一行和第i行都有染色格且第i行染色格为j至k，左边处于$left$状态（0或1，表示上升或下降），右边处于$right$状态。转移的时候利用预处理前缀和就可以$O(1)$查询区间和。

时间复杂度：$O(mn^2)$

空间复杂度：$O(n^2)$

1.28 Codeforces 273 E

Description

有一个游戏是这样的：首先现在纸上写下$n(n \leq 1000)$对数$l_i, r_i(1 \leq l_i \leq r_i \leq m)(m \leq 10^9)$，然后两位玩家轮流操作，每次可以选择一对$l_i, r_i$满足$r_i - l_i > 2$然后将这对数变成$l_i + \lfloor \frac{r_i - l_i}{3} \rfloor, l_i + 2\lfloor \frac{r_i - l_i}{3} \rfloor$或$l_i, r_i - \lfloor \frac{r_i - l_i}{3} \rfloor$。不能操作的人输。

问有多少种初始状态使得先手必胜。模$10^9 + 7$。

Analysis

首先可以发现SG值只和$r_i - l_i$有关，所以DP的时候按$r_i - l_i$DP即可。

暴力求SG值会超时。那么观察到SG值可以分成很多段相同的区间，并且容易证明区间的个数是$O(\log m)$的（因为左端点为i的一段SG值长度为$\frac{i}{3}$左右）。那么可以把每段区间的SG值求出来，这样就可以在$O(\log m)$的时间内求出SG值为$0, 1, 2$的分别有多少对l_i, r_i。

接下来考虑DP：$f(i, j)$表示写下i对数且SG值为j时的方案数，那么可以在$O(n)$的时间内求出答案。

时间复杂度：$O(n + \log m)$

空间复杂度：$O(n + \log m)$

1.29 Codeforces 274 C

Description

平面上有$n(n \leq 100)$个点，时间为t的时候每个点上会有一个半径为t的圆。求最后一个洞消失的时间或者输出没有洞出现过。

Analysis

对于任意的三个点，如果它们构成的三角形是锐角三角形，则外心有可能成为答案；如果是钝角三角形则不会形成洞；如果是直角三角形那么必须存在一个点使得这四个点构成矩形那么矩形的中心有可能成为答案。

对于每个可能成为答案的点假如在它成为洞之前就被其他的圆覆盖了那么就不会有洞。只需要枚举每一个点看最晚覆盖到它的点是不是枚举它的那三个点即可。

时间复杂度：$O(n^4)$

空间复杂度：$O(n)$

1.30 Codeforces 277 D

Description

给你$n(n \leq 1000)$道题和$m(m \leq 1560)$的时间，每道题有easy和hard，要做某道题的hard 要先做easy。给出每部分的所需时间和得分。easy题必定做对，hard题有p_i的概率做错。求最高期望得分和满足最高期望得分的情况下的最小期望罚时（罚时是最后一次正确提交的时间）

Analysis

第一问可以用背包解决：每道题可以做easy或做easy+hard。

假如我们已经求出了该做哪些题，那么做题顺序肯定是先做所有的easy再做hard。考虑交换相邻的两个hard造成的影响，化简后可以得到x在y前面当且仅当$(1-fail[x])\times fail[y]\times time[x]+(1-fail[y])\times(time[x]+time[y])<(1-fail[y])\times fail[x]\times time[y]+(1-fail[x])\times(time[x]+time[y])$。

所以只要先将题目排序，那么hard的顺序一定是在原排序中递增的顺序。所以只要排序后DP，最优解一定是将hard加在当前最优解的最后。这就非常容易计算了。

时间复杂度：$O(mn)$

空间复杂度：$O(m+n)$

1.31 Codeforces 283 E

Description

有$n(n \leq 100000)$头牛，每头牛有一个互不相同的战斗值，战斗值高的必定能战胜战斗值低的。现在有$m(m \leq 100000)$组操作，每次操作可以将所有双方战斗值都处于l,r之间的比赛结果翻转。问所有操作都做完后有多少个三元组(p,q,r)满足p胜q，q胜r，r胜p。

Analysis

这样的三元组个数=所有三元组个数-有一头牛胜了两头牛的个数。后者就是$\sum C(\text{这头牛能赢的牛的数量},2)$。那么每次操作相当于对一个正方形取反。这可以利用扫描线+线段树解决。在处理的时候不必记录该点的胜负情况，只需记录取反次数奇偶性即可。

时间复杂度：$O((n+m)\log n)$

空间复杂度：$O(n\log n)$

1.32 Codeforces 285 E

Description

问$1\cdots n(n\leq 1000)$的排列中有多少个恰好有$k(k\leq n)$个位置i满足$|a_i-i|=1$。模10^9+7。

Analysis

首先考虑求出不少于k个位置满足的方案数。令$f(i,j,S)$表示前i位至少有j个满足并且第i位和第$i+1$位的使用状态为S的方案数。这样可以在$O(n^2)$的时间内求出这些答案。那么$g(i)=\sum_S f(n,i,S)$就是不少于i个位置满足的方案数。

设$F(i)$为恰好i个位置满足的方案数，那么显然$F(n)=g(n)$。那么考虑已经知道$F(i+1)\cdots F(n)$和$g(i)$如何求出$f(i)$。i在j中出现的次数就是C_j^i次，所以只需要将$g(i)$减掉这些重复部分即可。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

1.33 Codeforces 286 D

Description

有一条道路，有$m(m\leq 10^5)$道墙会在t_i时刻突然出现，范围是l_i到r_i，墙可以以任意形式相交。有$n(n\leq 10^5)$对人，第i对人会在p_i时刻从原点出发，问有多少时间这两个人被墙挡住。

Analysis

首先将墙离散，这样每一段内的墙出现时间都相同。我们只需要找出每一段内最早出现的时间就可以了，后来出现的墙是没有影响的。首先按时间排序，就可以利用并查集在$O(m\alpha(m))$的时间内解决（或者用set在$O(m\log m)$的时间内解决）。

这样现在的墙都是独立的，可以分开考虑贡献。每一堵墙(l_i,r_i,t_i)对起点小于t_i-r_i的不造成影响，对起点位于t_i-r_i和t_i-l_i之间的位置x造成$x-(t_i-r_i)$的贡献，对起点大于t_i-l_i的造成r_i-l_i的影响。这样我们可以把每堵墙(l_i,r_i,t_i)转化成(t_i-r_i,t_i-l_i)这条线段，然后每个人的答案就是它左边的所有线段长度之和。这可以利用扫描线在$O(m\log m)$的时间内解决。

时间复杂度：$O(m\log m+n)$

空间复杂度：$O(m+n)$

1.34 Codeforces 286 E

Description

给$n(n \leq 10^6)$个包，第i个包能装质量正好为a_i的物品。你要选出尽量少的物品，使得满足以下两个条件：

- 每个包都能表示成某些物品的和（每种物品有无限个）
- 每种总质量不超过$m(m \leq 10^6)$的选取物品的方案都有一只对应的包能装下。

输出方案或输出无解。

Analysis

考虑如下的贪心策略：首先将a_i排序，那么小于a_1的物品显然不能拿（否则只选那个物品就不存在对应的包了），那么a_1一定要拿。以此类推上去，假如存在两个包的大小等于这个包，那么就不需要拿了（不管那两个包拿还是没拿），否则就要拿上。最后检查一遍是否拼出了除了这些大小以外的大小即可。这样的复杂度是$O(n^2)$的。

容易发现上面的贪心策略其实就是“取所有不能被其他两个包拼成的包”，无解当且仅当“有两个包容量加起来不超过m并且没有一个对应的包”。所以只需要一次卷积即可。使用FFT可以再$O(m \log m)$的时间内解决。

时间复杂度：$O(m \log m)$
空间复杂度：$O(m)$

1.35 Codeforces 293 B

Description

给定一个$n \times m(m, n \leq 1000)$的矩阵，每个格子需要涂上一种颜色，总共有$k(k \leq 10)$种颜色。有些格子只能涂一种颜色，有些格子可以涂任意的颜色。要求最后从左上角到右下角的所有只向下不向右的路径上颜色均不相同。求方案数 $\mod 1000000007$。

Analysis

首先发现假如$n + m - 1 \geq k$那么肯定无解。所以需要做的只有m, n很小的情况。

考虑搜索任意一种合法的涂色方案。直接做肯定会超时，所以我们可以只搜最小表示的情况，即每个格子的颜色不能比前面最大的颜色大1以

上。这样就可以不重不漏地搜出所有方案（搜出的方案代表哪些格子颜色相同）。通过达打表可以发现，这样的方案数非常少完全可以承受。

然后需要检验一下每种方案符不符合当前情况，即是否原矩阵同色的位置该矩阵内也同色，是否该矩阵同色的位置原矩阵内也同色。统计的时候每种方案再乘上一个排列数算出和它等价的方案数即可。

时间复杂度：指数级

空间复杂度：$O(k^2)$

1.36 Codeforces 293 E

Description

给出一个$n(n \leq 10^5)$带边权的树，询问有多少对点满足树上带权距离$\leq W(W \leq 10^9)$，不带权距离$\leq L(L \leq n)$。

Analysis

考虑点分治：每次统计经过当前选取的重心的路径条数。首先可以BFS出每个点到重心的距离，然后每个点需要统计两个距离分别小于某个值的点的个数。这是经典的静态二维数点问题可以用扫描线+树状数组在$O(n \log n)$的时间内解决。注意这样会重复计算同一个子树中的点对，需要对每个子树都计算一次并在答案中减去这一部分。

时间复杂度：$O(n \log^2 n)$

空间复杂度：$O(n)$

1.37 Codeforces 294 D

Description

给你一个$m \times n(m, n \leq 10^5)$的网格，一开始都是白色的。在$(x, y)$处有一个机器人，面朝四个方向之一（左上，左下，右上，右下）。机器人会沿着这个方向不停走下去，碰到边界时会遵循光的反射定律改变方向。机器人每走过一个格子就会损失一个染料并将这一格染黑（不管他现在是什么颜色）。求机器人至少要花掉多少染料才能将网格涂成黑白相间的或者输出不可行。

Analysis

首先可以猜想一个结论：当所有边上的该访问的点都访问过的时候就是整个棋盘都访问过的时候。假如这个结论是正确的，那么只需要模拟行动，每次从一条边走到另一条边。当某个边上的点被访问3次以上就无解。下面我们来证明这个结论。

所有边上的点（除了第一个访问的和最后一个访问的）延伸出去的两条边都是被访问过的。第一个和最后一个至少有一条边是被访问过的。因为能够访问到所有边上的点，所以路径肯定能够成一条回路（可能缺了一条边）或者是从角上出发在角上结束的路径。那么每个不在边上的点都可以从两个方向访问到。而最多只有一个方向可能没有访问。所以每个中间点都已经被访问过了。

时间复杂度：$O(n+m)$

空间复杂度：$O(n+m)$

1.38 Codeforces 295 D

Description

给定一个$m \times n(m,n \leq 2000)$的棋盘，你需要将一些格子涂成黑色，使得满足以下条件：存在三个数l,t,r使得只在l行到r行中有黑格，且这些行中的黑格连续，且对于两行x和y，假如$x \geq t$那么y行是x行的子集，否则x行是y行的子集。求方案数模10^9+7。

Analysis

首先先求出上半部分的方案数。用$f(i,j)$来表示在$i \times j$的棋盘上最后一行全部涂满的方案数$(j \geq 2)$。那么$f(i,j)=1+f(i-1,2)*(j-2+1)+f(i-1,3)*(j-3+1)+\cdots+f(i-1,j)$。即$f(i,j)=f(i,j-1)+f(i-1,2)+f(i-1,3)+\cdots+f(i-1,j)(j \geq 3)$。那么只需要在DP的时候再记录一个前缀和即可。

求出了上半部分的方案数以后我们可以枚举t和第t行的宽度x，那么这种情况下的方案数就是$(n-x+1) \times (f(t,x)-f(t-1,x)) \times f(m-t+1,x)$。把所有情况的方案数加起来即可。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

1.39 Codeforces 301 C

Description

你需要用一个奇怪的语言写一些算法。这个算法以一个字符串a作为输入。

这种语言由很多条命令组成，每条命令格式如下：

- $s >> w$，其中s和w是长度不超过7的字符串（每个字符是数字或问号），含义为：如果s在a中出现过，则将s在a中第一次出现的位置替换为w。

- $s <> w$，含义同上，但是执行完操作后中止算法。

执行命令的规则如下：每次遍历每条命令，找到第一条可以产生效果的命令，执行之，然后再从头开始找新的命令。如果没有可执行的指令，算法结束。结束时的串就作为输出。

现在给出$n(n \leq 100)$个正整数，每个数不超过10^{25}，让你用这个语言写一个算法，要求这个算法对于每个给出的数作为输入，都能输出它+1后的值。命令条数不超过50，对于每个输入算法的执行步数不超过200。

Analysis

这题看似需要对每个不同的输入设计不同的方案，但其实可以构造出一种通用的方案对所有数都有效。

比如可以固定输出这个串：

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

这个算法的步骤是：

- 首先串输入时由于没有问号所以执行最后一条语句在开头加入问号。
- 然后第12-21条语句把这个问号移到末尾

- 接下来第22条语句把它变成两个问号。
- 然后假如最后一位是0-8，那么算法结束。如果是9，就换成0并前移问号并重复这一步骤。
- 假如这对问号被移到了开头，那么就应该被替换成1。

时间复杂度：$O(1)$
空间复杂度：$O(1)$

1.40 Codeforces 303 D

Description

定义Rotatable Number为长度为n的可含前导零的数，满足这个数的1倍，2倍...n倍都与自己循环同构。求最大的$b(1 \le b \le x)$使得b进制下存在长度为$n(n \le 5 \times 10^6)$的Rotatable Number。$x \le 10^9$

Analysis

从维基百科可以知道，长度为n的Rotatable Number都是$\frac{1}{n+1}$的小数点后n位。所以

$$kx = \frac{k}{n+1} = b^p x - q$$

于是有

$$k \equiv b^p \mod n+1$$

因为对于$1..n$的每个k都存在一个对应的p，所以$n+1$一定是质数且b一定是$n+1$的原根。那么只需要判断一个数是不是另一个数的原根即可。

判断x是不是质数p的原根只需要判断是不是只有$x^{p-1} = p$即是不是所有$p-1$的约数i都不满足$x^i = p$即可。时间复杂度$O(\sqrt{n})$。由于质数有$\phi(\phi(p))$个原根，所以暴力枚举或者随机都有很好的效果。

时间复杂度：$O(\frac{n}{\phi(\phi(n))}\sqrt{x} + \sqrt{n})$
空间复杂度：$O(1)$

1.41 Codeforces 303 E

Description

有n个学生，每个学生的分数分布在l_i到r_i之间，且不会有并列出现。输出n^2个实数表示每个学生得每个名次的概率。

Analysis

首先将区间离散，之后对于每个学生枚举他的分数处于某一个离散后的区间内时各个名次的概率。这部分的复杂度是$O(n^2)$。

考虑用$f_{i,j}$表示有i个人比这个区间高，有j个人比这个区间低，剩下的人处于这个区间内的概率，那么$f_{i,j}$会对ans_{i+1}到ans_{n-j}造成$\frac{f_{i,j}}{n-i-j}$的贡献（因为在那个区间内的所有人期望名次都相同）。而$f_{i,j}$可以利用$O(n^3)$的DP求出。于是我们就得到了一个$O(n^5)$的算法。这个算法经测试可以通过所有测试数据。

经过观察发现$f_{i,j}$其实就是一个二元多项式，而每次循环其实就是给$f_{i,j}$乘上一个二元一次多项式。那么对于相同的一段区间不同的两个人乘上的多项式大部分都是相同的，而且每个人其实就是其他所有人的多项式的乘积。那么我们只要预处理出所有多项式的乘积，然后对于每个人除掉他自己的多项式即可。这个做法的复杂度是$O(n^4)$，不过由于本题精度要求非常高而且这种算法引入了除法运算导致失去了很多精度于是无法通过本题。

时间复杂度：$O(n^5)$

空间复杂度：$O(n^2)$

1.42 Codeforces 305 D

Description

给定一张$n(n \leq 10^6)$个点$m(m \leq 10^5)$条边的有向图，且保证原图中任意从u到v的有向边满足不等式：$u \leq v$。求有多少种加边的方案使得加边后的图满足以下条件：

- 从i出发可以到$i+1, i+2 \cdots n$。
- 任意从u到v的有向边满足不等式：$u \leq v$。
- 两点之间最多有一条边。
- 对于一对点i、$j(i \leq j)$，若$j-i \leq k$，那么从i到j的最短距离等于$j-i$条边。
- 对于一对点i、$j(i \leq j)$，若$j-i \geq k$，那么从i到j的最短距离等于$j-i$或$j-i-k$条边。

答案模10^9+7。

Analysis

由于i可以到$i+1, i+2 \cdots n$，所以i和$i+1$之间一定有边，即最后的图一定包含1到n的一条链。除了这种边以外只可能有i到$i+k+1$的边。假如输入中存在其他边那么答案肯定是0。

第二类的边还需要满足一个条件就是不能有两条边使得第一条边的终点不大于第二条边起点，即最大的起点-最小的起点$\leq k$。那么只需要枚举起点最小的那条边的起点是几号点，并算出可加可不加的边的数量p（这可以通过预处理前缀和解决），那么答案就应该加上2^p。最后要注意的是假如初始没有第二条边那么也可以不加第二条边，所以答案要+1。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

1.43 Codeforces 305 E

Description

两个人玩一个游戏。初始时只有有一个长度为$n(n \leq 5000)$的字符串s。

两个人轮流操作，每次操作可以选择一个字符串t，然后选择一个位置i（不能是第一个或最后一个）使得存在一个$k(0 \leq i-k, i+k \leq |t|)$满足$t_{i-1} = t_{i+1}, t_{i-2} = t_{i+2}, \cdots, t_{i-k} = t_{i+k}$，然后删除这个串并加入两个串：$t_1 t_2 \cdots t_{i-1}, t_{i+1} t_{i+2} \cdots t_{|t|}$。

问先手胜还是后手胜。若先手胜则需输出最小第一步可行方案。

Analysis

注意到假如$k = p$时满足条件，那么$k = p-1$时也满足条件。所以只要判断$k = 1$时是否满足条件即可，即判断t_{i-1}是否和t_{i+1}相等。

现在我们换一种形式：在操作时并不将串分成两半，而是将选中的字母标记为不可用。那么一个字母t_i能被选择的条件是：$t_{i-1} = t_{i+1}$并且t_{i-1}和t_{i+1}都没有被选择过。

那么我们先处理出所有满足$t_{i-1} = t_{i+1}$的i，那么每一段连续的可行点之间都是独立的，不会相互影响。所以我们可以计算出每一段的SG值然后异或起来就可以算出整个局面的SG 值。

注意到每一段的SG值只和这一段的长度有关。而长度总共只有5000种。于是我们可以预处理出每个长度的SG值，就可以做到$O(n)$计算某个局面的SG值。

那么我们可以从前到后枚举第一步操作然后$O(n)$扫一遍判断是否可行就可以找到方案。

时间复杂度：$O(n^2)$，已经可以通过本题所有数据

空间复杂度：$O(n)$

1.44 Codeforces 306 C

Description

有$n(n \leq 4000)$天，$w(w \leq 4000)$件好事，$b(b \leq 4000)$件坏事，每天可以发生大于0件好事或者大于0件坏事。并且已知坏事发生的日期是连续的（即假如第i天的前一天没有坏事那么之前的$i-1$天都没有坏事。后一天以此类推）。求一共有多少种方案。

Analysis

首先枚举坏事发生的区间，然后就变成了在几天内分几件事的问题。这个问题可以用隔板法转化为组合数问题。所以我们可以与处理出4000以内的组合数$O(1)$统计。

时间复杂度：$O(n^2 + n \times (w+b))$

空间复杂度：$O(n \times (w+b))$

1.45 Codeforces 306 D

Description

要求构造一个所有内角都相等的凸$n(3 \leq n \leq 100)$边形使得任意两条边的长度差不小于10^{-3}。边的长度不得超过1000。（边的长度不一定是整数）

Analysis

由于边的长度只需要相差10^{-3}即可，所以几乎不影响多边形的形状。所以可以这样：首先选取$n-2$个非常接近的数（相对误差非常小），比如$100, 100.002, 100.004 \cdots$，然后用这$n-2$个数去构造多边形的前$n-2$条边，最后两条边利用交点来求。在构造的时候假如按边长度升序排列那么最后两条边将会比100大出不少，但由于相对误差很小，最后两条边虽然会比之前选取的边大很多但是还是不会超过1000。容易看出这个解一定合法。

时间复杂度：$O(n)$

空间复杂度：$O(1)$

1.46 Codeforces 309 B

Description

给n个总长度不超过10^6的单词，要求选出尽量多的连续的单词，使得可以分成不超过c块，每块可以在宽为r的一行内写下（即长度和+单词数

量-1不超过r）。($r \times c \leq 10^6$)

Analysis

首先把所有单词长度+1，宽度也+1，那么就可以直接用区间和比较能否放在一行内。

用two pointers算出每个单词开始只用一行能到的最右边位置，然后倍增算出用2^k行能到的位置。这样就能在$O(\log n)$的时间内算出每个位置作为最左点的答案。总复杂度$O(n \log n)$

时间复杂度：$O(n \log n)$

空间复杂度：$O(n \log n)$

1.47 Codeforces 309 D

Description

给定一个正三角形，每条边被$n(n \leq 32000)$个洞等分成了$n+1$份。现在要在每条边上选一个洞构成一个钝角三角形，其中和正三角形的角距离不超过m段的洞不能被选择，求方案数。

Analysis

首先发现只需要枚举钝角所在边然后将答案乘3即可。

枚举了作为钝角的那个点以后在另一条边上按顺序枚举第二个点的同时维护第三条边上的可行范围并更新答案。

时间复杂度$O(n^2)$，需要加一些常数优化。比如：当第三条边上没有可行点时就break，第一条边上的点只需要枚举一半另一半可以通过对称得到。

时间复杂度：$O(n^2)$

空间复杂度：$O(1)$

1.48 Codeforces 311 E

Description

有$n(n \leq 10^4)$只狗，你知道他们的性别，并且可以花费a_i改变第i只狗的性别。现在有$m(m \leq 2000)$个要求，假如给出的$p_i(p_i \leq 10)$只狗的性别全部等于t_i的话可以获得v_i的奖金，否则假如这个人是你的朋友那么损失g 的前，假如不是那么什么事都不会发生。求最大收益或最小亏损。

Analysis

建立最小割模型：首先假设获得全部奖金，然后利用最小割求出最小亏损。

首先为源汇点各分配一个性别。为每只狗建立一个结点，并从源点向它连边，它向汇点连边。性别相同的那边代价为0，另一边代价为a_i。这样割哪一边就表示选了哪一边的性别。

为每个要求建立一个结点，假如要求是和源点相同的性别，那么从p_i只狗的这些结点向要求结点连边，要求结点再向汇点连边。否则就从源点向要求结点连边，要求结点再向p_i只狗的这些结点连边。最后跑一遍最小割即可。

最后建出来的图点数为$n+m+2$，边数为$2n+m+\sum p_i$。

时间复杂度：$O(V^2E)$即$O((n+\sum p_i)^3)$

空间复杂度：$O(n+\sum p_i)$

1.49 Codeforces 314 E

Description

输入一个长度为$n(n \leq 10^5)$的串，包含小写字母和问号。求把问号替换成（小写or大写）字母后，把字母相同的大小写字母当成一对左括号和右括号后是一个合法的括号序列的方案数。并且字母x不能出现。答案模2^{32}。

Analysis

首先我们只需统计括号序列的方案数然后最后乘上 $n/2$减去原串中的字母数 即可。

我们用f_i表示有i个右括号的方案数。转移的时候对于每个字母位什么都不用做；对于每个问号位需要对所有满足要求的i做$f_i += f_{i-1}$。由于模2^{32}可以直接用unsigned int存。由于常数非常小所以能在1s左右出解。

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

1.50 Codeforces 316 D

Description

有$n(n \leq 10^6)$个人，每个人手里拿着一个编号为$1 \cdots n$的球。每次可以让两个人交换手中的球。但是每个人有交换次数上限$a_i(1 \leq a_i \leq 2)$。求最后可能的持球方案数量摸10^9+7。

Analysis

容易发现人的顺序没有意义，所有1和所有2都是等价的。只需考虑1和2的数量即可。

首先考虑假设我们知道了目标状态，如何判断是否可行。

假如一个人只剩一次了那么他不能随便和别人交换了，所以交换方式固定。交换完后有两种情况：1. 和他交换的那个人原来是2，那么现在那个人变成了1，所以重复上述过程。2.和他交换的那个人原来是1，那么那个人拿到的球只能是正确的。所以在置换中那个人指向的是当前的这个人。所以这一路上的人形成一个环。3.本来就拿到正确的球。这在置换中是一个自环。

容易发现只第二种情况中会失败。进一步分析发现当且仅当一个置换中有超过两个1才会失败。那么此时2的存在不会影响可能性。所以可以先考虑1的分组再将2加进去。

1的分组可以用一个很简单的DP来实现：$f(i)$表示i个1有多少种方案。那么新加进一个1的时候可以成自环也可以和另外一个1组成一个环。所以$f(i) = f(i-1) + (i-1) * f(i-2)$，利用滚动数组做到$O(1)$空间。

在加2的时候第i个加入的2有 1的个数$+i$ 种加的位置。所以方案数再乘上$\frac{(a+b)!}{a!}$ 即可。

时间复杂度：$O(n)$

空间复杂度：$O(1)$

1.51 Codeforces 316 G

Description

给定一个字符串s_0和$n(n \leq 10)$个模板串$s_1 \cdots s_n$（串长均不超过50000），问s_0中有多少个子串满足对任意i在s_i中出现次数在l_i到r_i之间。

Analysis

首先将所有串连接在一起，中间用空白字符分割。然后建出这个合并串的后缀自动机，那么可以很方便地统计出每个结点所代表的那些串在每个模板串中的出现次数（对于每个模板串的结点都+1，然后算出逆序后缀树的子树和即可）。对于每个结点，假如它满足所有条件并且在s_0中出现过那么就将它所代表的那些子串（即$len_i - len_{f_i}$）加到答案中即可。

由于后缀自动机的内存需求较大，并且本题时间复杂度低于空间复杂度，若将每个结点的儿子改用map 存储可降低约一半的空间开销，但是时间会慢5倍左右。

时间复杂度：$O(n^2|s|)$

空间复杂度：$O(n|s|(\Sigma + n))$

1.52 Codeforces 317 C

Description

给定一个$n(n \leq 300)$个点$e(e \leq 50000)$条边的图，每个点有相同的容量上限v，初始容量a_i和目标容量b_i。

每次操作可以选择相邻的两个点并转移大于0的流量。但是转移完毕后容量不能超出容量上限也不能低于0。要求在$2n^2$步之内达到目标或者输出无解。

Analysis

首先只需要考虑其中一棵生成森林。然后枚举两个点i, j，假如$a_i \geq b_i \ and \ a_j \leq b_j$那么我们从$i$ 转移到$j \min(a_i - b_i, b_j - a_j)$的流量。因为转移一次后不平衡点就少一个，所以最多进行n次转移。所以我们只需要在$2n$次之内完成i到j的转移即可。

我们需要实现一个函数$move(i, j, k)$表示在$2dis_{i,j}$次之内从i到j转移k的流量。那么有以下几种情况：

- 首先假如$dis_{i,j} = 1$直接输出方案并退出。
- 找到j向i走一步到的点p
- 假如$a_p \geq k$那么先$move(p, j, k)$然后$move(i, p, k)$
- 否则就先$move(p, j, a_p)$然后$move(i, p, k)$然后$move(p, j, k - a_p)$

容易看出这个算法可以在$2dis_{i,j}$次之内成功转移。而且转移过程中不会超过容量上限。

由于这题数据规模很小，可以不求出生成森林而直接用floyd代替。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^2)$

1.53 Codeforces 319 D

Description

给定一个长度为$n(n \leq 50000)$的字符串，要求每次选择一个连续出现两次的串并将其中一份删除直到没有符合条件的串为止。若有多个优先选择最短的，长度相同的时候优先选择前面的。

Analysis

考虑暴力做法：从小到大枚举长度并$O(n)$检查是否有满足条件的串并删除。由于删掉一个重复串后不会有更短的重复串出现，所以可以直接从小到大枚举。这样复杂度就是$O(n^2)$的。

假如枚举出了重复串长度len以后，把当前的串每len个字母分成一份总共$\frac{|s|}{len}$份，对于每一份只需要检查这一份和前后的份的最长公共前后缀（最长公共前后缀可以用hash预处理在$O(\log n)$的时间内算出）之和，如果结果$\geq len$那么就找到了一个重复串。由于最多只有$\sqrt{n}$个len会导致串长减少，所以每次可以暴力更新串。

时间复杂度：$O(n\sqrt{n} + n\log^2 n)$

空间复杂度：$O(n)$

1.54 Codeforces 321 D

Description

有一个n行n列$(n \leq 33)$的表格，每个格子上有一个数字。n是一个奇数，不妨设$x = \frac{n+1}{2}$。每次可以选择一个x行x列的子矩阵，并将其中的所有元素乘-1。这个操作可以使用任意多次。最大化表格里的数字和。

Analysis

容易注意到同一个位置翻转两次是无意义的，而可以选择的位置只有$x*x$个，所以状态数为2^{x*x}。定义“翻转矩阵”f为每个格子被操作的次数 $\bmod 2$，则可以证明：

- 1.对于任意$1 <= i <= n, 1 <= j < x$满足$f[i][j] \oplus f[i][x] = f[i][j+x]$
- 2.对于任意$1 <= i < x, 1 <= j <= n$满足$f[i][j] \oplus f[x][j] = f[i+x][j]$

这是因为每次操作只能改变这3个格子中的偶数个，所以这三个格子的和始终为0或2。所以，当左上角的$x*x$个格子确定了以后，整个矩阵就确定了，所以满足以上两个条件的状态只有2^{x*x}种，即都可以达到。

假如我们枚举$f_{1,x}f_{2,x}...f_{x,x}$，那么每一列都是独立的，可以分别贪心。具体做法如下：

由于$f_{i,j}$和$f_{i,j+x}$只能同时变符号，所以这两个格子可以加在一起计算。

对于第j列，枚举$f_{x,j}$的值然后$f_{i,j}$和$f_{i+x,j}$的值也只能同时变符号所以相关的这四个格子的值就确定了，于是就可以确定出这一列的最优解。

时间复杂度：$O(2^x * x^3)$

空间复杂度：$O(n^2)$

1.55 Codeforces 323 B

Description

你要构造一个有$n(3 \leq n \leq 1000)$个结点的竞赛图，使得对任意两个结点u和v（u≠v），从u到v 的最短距离不超过2。竞赛图就是基图为无向完全图的有向图（每对结点之间有一条有向边相连，且无自环）。

Analysis

通过搜索发现$n = 4$时无解。

假如n是奇数，那么可以让所有点排成一个环，然后每个点向它后面的$\frac{n-1}{2}$个点连边。这样显然是可行的。

假如n是偶数，那么可以先处理出前$n-1$个点，然后最后一个点和奇数编号的连出边，剩下的连入边。由于$n = 4$ 时前三个点只有一条出边到下一个点，就无法覆盖所有的点。但当$n > 4$时是可以做到的。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

1.56 Codeforces 323 C

Description

给定两个$1..n(n \leq 1000000)$的排列和$m(m \leq 200000)$次询问，每次问有多少个数在第一个排列中的位置是$[l1, r1]$，在第二个排列中的位置是$[l2, r2]$，强制在线。

Analysis

把这两个排列想象成置换，然后对第一个排列用第二个排列的逆置换。这样原问题就等价为：在新的排列中某个区间内有多少个数的值在某个范围内。这可以用可持久化线段树解决。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n \log n)$

1.57 Codeforces 325 C

Description

有n个怪物和m个变换规则$(n, m \leq 10^5)$，每个规则是把某个怪物变成至少0个怪物和至少1个钻石。问从每个东西开始全部变成钻石最少和最多能得到的钻石数。假如一个怪物无法全部变成钻石输出-1 -1，假如一个怪物可以变出无穷多个钻石输出-2。

Analysis

对于最小的话可以用类似Dijkstra的方法来做：先用没有怪物算出能到没有怪物状态的距离并加入优先队列。然后每次取出优先队列中最小的并将这个怪物的距离确定，并用这个怪物去更新其他怪物并加入优先队列。最后没有访问的怪物就是无法全部变成钻石的。

最大的话可以用记忆化搜索：对于每个怪物搜索那些不会进入-1-1怪的方案。假如搜索到了当前正在栈中的怪物，那么可以肯定目前栈中这些怪物都是-2可以直接返回。所以这一步是$O(n+m)$的。

时间复杂度：$O(n \log n + m)$

空间复杂度：$O(n)$

1.58 Codeforces 325 D

Description

给出一个圆柱体，并将它的侧面划分成$m \times n (m, n \leq 3000)$个矩形。初始的时候都是海，有$q (q \leq 300000)$ 次操作，每次操作可以将一块海变成陆地，但是如果操作完以后圆柱体的两个底面不四连通了就撤销操作。问有几次操作成功了。

Analysis

底面不四连通等价于存在一块陆地能八连通地绕一圈连接到自己。所以我们将矩形复制一份变成$m \times 2n$，每次加入陆地同时修改左右两个点。假如加入了一个点后底面不连通了那么这个点一定可以和复制出来的那个点八联通。那么这就变成了一个简单的并查集问题。

需要注意的是复制完后右边界和左边界还是需要连通的。

时间复杂度：$O(q\alpha(q))$

空间复杂度：$O(mn)$

1.59 Codeforces 325 E

Description

给出一个$n (n \leq 100000)$个点的图，标号为$0..n-1$，每个点x向$x * 2 \mod n$和$x * 2 + 1 \mod n$连边。求一条哈密顿回路或者输出无解。

Analysis

首先注意到假如n为奇数则无解。证明：能到0号点的只有$\frac{n-1}{2}$，能到$n-1$号点的也只有$\frac{n-1}{2}$ 所以肯定无解。

当n为偶数时我们可以采用这样的构造方法：从0开始倒着搜，假如$\frac{x+n}{2}$没被访问过就走这个点否则假如$\frac{x}{2}$没被访问过就走这个点，假如都被访问过就结束。

以下是证明：

引理1：x不可能和$x+\frac{x}{2}$在同一个环中，除非是n个点的环。

证明：假如在同一个环中，因为能到$x*2 \mod n$和$x*2+1 \mod n$的只有x和$x+\frac{n}{2}$，所以$x*2 \mod n$ 和$x*2+1 \mod n$也和它们在同一个环中。以此类推能推出所有点都在一个环中。

引理2：除非已经走遍了所有的点否则两个点中至少有一个能走。

证明：假如两个点都不能走，那么有两种情况：1.假如$x \neq n-1$那么$\frac{x+n}{2}$和$\frac{x}{2}$都不和x 或$x+\frac{n}{2}$中的任何一个相同。所以都不能走只发生在x和$x+\frac{x}{2}$都已经访问过了的情况，这显然不可能。2.假如$x=n-1$，那么说明$\frac{n-1}{2}$ 不能走。因为搜索的时候是先走$\frac{x+n}{2}$的，所以$n-1$已经被访问过了。这显然也不可能出现。

因为不会过早地回到0，也不会无路可走，所以这个算法能够正确地执行。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

1.60 Codeforces 329 D

Description

构造一个$n \times n(n=100)$的地图，每个格子可以是空地或者一个石头，每个石头有一个方向。你需要激活一个石头，这个石头会沿着它的方向一直走直到撞墙或撞石头，如果是石头那么新石头会被激活，如果它移动了至少一格就会发出响声。构造一个地图，使得响声至少为100000。（在清澄上$n \leq 300$且为偶数，响声数至少为$\frac{n}{2}^3-\frac{n}{2}^2$）

Analysis

注意到形如>>>>>.>.>.>的东西会发出$x_1 \times x_2$的响声（x_1为左边石头个数，x_2为右边空隙个数）所以假如一排有一半的紧挨着的石头和一半的隔一格的石头就能发出最多的响声。并且碰撞次数是所有x_1中最小的那个，所以就应该保证每条路都尽可能的长。所以我们可以采取不断上下绕然后利用最下面一条边连起来构造$n+1$ 条非常长的路。这样就能达到碰撞次数要求。

时间复杂度：$O(n^2)$

空间复杂度：$O(n^2)$

1.61 Codeforces 331 C

Description

给定一个数$n(n \leq 10^{18})$，每次可以将这个数减去它某一位上的数，问最少要多少次才能变成0。

Analysis

一个很容易发现的结论是减的次数这个函数是单调不减的，因为不存在相邻两个数使得大的那个的最大数位比小的那个大1 以上。所以每次减最大的数肯定是最优的。那么考虑动态规划：$dp(i,j)$表示数字i在前几位最大数是j的情况下需要消的次数和消完后剩下的数（介于-9和0之间）。

由于大多数的状态是没用的，所以直接记忆化搜索即可。每次算出去掉首位的答案和减掉末尾的答案。容易发现除了开始的数衍生出的$O(\log n)$个数外只有形如x99999999y的数会被计算。所以总的复杂度是$O(\log n)$的。

时间复杂度：$O(\log n)$

空间复杂度：$O(\log n)$

1.62 Codeforces 332 D

Description

给定一个$n(n \leq 2000)$个点的带权无向图，满足任意选出$k(k \leq n)$个点都有且仅有一个点和这k个点都相邻。代价为这k条边的和。求代价的期望。

Analysis

当$k = 1$时，直接枚举每个点即可。$k = 2$时枚举会和的那个点然后统计处边即可。当$n = k + 1$且是完全图时，枚举没被选到的那个点即可。复杂度都是$O(n)$。假如我们能证明$k \geq 3$时一定是$n = k + 1$ 的完全图那么这题就解决了。

以下均默认$k \geq 3$。

引理1：任选两个点，和这两个点都相邻的点的个数是$k - 1$。

证明：假如多余$k - 1$那么从这些点中选出k个那么就有两个相邻点，与题意矛盾。假如少于$k - 1$那么选上所有的点和枚举的这两个点就找不到和这些点相邻的点了，与题意矛盾。

引理2：这个图中一定含有点数为$n + 1$的完全子图。

证明：任选k个点$a_1 \cdots a_k$，找到它们对应的相邻点a_{k+1}。因为这其中任取k个点都和剩下的点连通，容易看出这是一个完全图。

结合引理1,2容易发现这个完全子图和子图外的点没有边。又因为$k \geq 2$时必须是一个连通图，所以这个完全子图就是原图，即$n = k + 1$。

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

1.63 Codeforces 332 E

Description

有一个解密算法，它接收两个串p和q作为输入。其中q是01串。这个算法首先把q不停地复制并接在自己后面，直到p 的长度不大于q的长度为止，然后删掉q后面多余的部分，这时$p.length() = q.length()$。接下来删掉p串中q 串对应位置是0的地方。返回p串。

现在给出$p(|p| \leq 10^6)$串和返回的串$s(|s| \leq 200)$，要求一个字典序最小的满足要求的长度为$t(t \leq 2000)$的q串。

Analysis

设p按t长度为一段分成的段数为len即$len = \lfloor \frac{|p|}{t} \rfloor$。首先我们枚举$q$ 串中的1的个数x，那么将答案串按x为一段分割成的段数一定是len。那么最多只有$O(\frac{|s|t}{|p|})$个可行的x。

知道1的个数以后那么p和q的每一段都分别对应。容易发现假如两个位置都可以是1且对应的q中的位置相同，那么右边的位置不仅字典序小左边可选择的1数量也更多所以肯定优于左边。所以只需要从右向左扫的时候贪心取1即可。判断某个位置可不可以是1只需要检查p和q的每一段在这一位上是否相等即可。这一步的复杂度是$O(len * t)$即$O(|p|)$。

时间复杂度：$O(|p| + |s|t)$

空间复杂度：$O(|p|)$

1.64 Codeforces 333 C

Description

要求给出$m(m \leq 300000)$个不同的长度为8的串使得可以在这些串上添加括号，加号，减号，乘号后结果等于给定的值$m(m \leq 10000)$。

Analysis

首先我们把这个8位串分成前4位和后4位。我们枚举前4位并算出可能的组合，然后后四位用加减法补齐。由于前四位能算出来的数都比较小（不超过10000），这样大多数的数算出来以后都能产生一种方案。通过打表发现即使只从左向右计算的方案数也足够达到300000。所以实现的时候

非常方便：只需要搜索即可。在搜索的时候记录当前的数字是几，当前算了前几位，算出来的结果是多少这三个参数就可以了。

时间复杂度：$O(m)$

空间复杂度：$O(1)$

1.65 Codeforces 335 D

Description

给定平面上$n(n \leq 10^5)$个不相交的矩形，坐标都不超过$X(X \leq 3000)$问是否存在某几个矩形拼成了一个正方形。如果有输出方案。

Analysis

首先可能成为最终正方形的左上角的点一定只可能是这些矩形左上角的点中的一个。那么我们可以枚举正方形左上角的点和边长，然后只需要检查这个正方形区域内被覆盖的面积和四条边是不是完整即可。查询面积可以维护二维前缀和来做到。因为正方形的任意一处边界必定是某个矩形的边，所以只需要对每个矩形记下哪些位置被它的边覆盖即可。那么只需要查询四条边是不是都被矩形边覆盖，维护一个前缀和即可。

直接做的复杂度是$O(nX)$，可以加上两个小优化：假如某个边长x由于面积不完整而不可行，那么这个点的其他边长也不可行；假如某个边长x使向右或者向下的边不完整那么这个点的其他边长也不可行。由于这两个剪枝的存在加上找到解就可以退出在实际情况中效果很好，几乎无法构造出数据使这个算法超时。

时间复杂度：$O(nX + X^2)$

空间复杂度：$O(n + X^2)$

1.66 Codeforces 338 E

Description

给定一个长度为n的序列a和长度为m的序列$b(m, n \leq 150000)$和h。问b的长度为n的连续子序列中有几个满足能和a中的数两两配对使得每一对的和都不小于h。

Analysis

假设a排序后的序列为p，由于每一对和不小于h即这些数排序后分别大于$h - p_i$，那么至少要有1个数不小于p_1，2个数不小于$p_2 \cdots$。所以我们只需要维护大于某个数的数的数量即可。那么每加入一个数或者删除一个数只需要前缀+1或者-1即可。这可以用线段树方便地维护。

当所有结点的最小值小于0时就说明至少有一个数不能配对。所以只要在修改的时候同时维护一个最小值即可。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

1.67 Codeforces 339 E

Description

给出一个长度为$n(n \leq 1000)$的全排列，每次操作可以选择一段区间并将其翻转。要求给出一个不超过3次的操作序列使得操作完后变成$1, 2, ..., n$的全排列。

Analysis

不妨考虑搜索。直接搜索复杂度是$O(n^6)$肯定无法承受。

首先我们可以设计一个贪心的算法将序列分成尽可能少的段使得每段内两两之间的差均为1或均为-1。这个算法很容易实现，只要每次尽量取长的一段即可。容易看出每次交换最多使段数-2。所以还剩k次操作时假如段数$\geq 1 + 2 \times k$则肯定无解。

然后再加入一个剪枝：交换完后区间头尾两个数至少有一个和旁边的数差1或者贴着边界。这样就可以防止无谓的交换。

加入了第二个剪枝后容易看出对于一个有k段的序列可行的方案最多只有$O(k^2)$种，由于第一个剪枝k被限制得很小，所以几乎可以忽略不计。

时间复杂度：$O(n^2)$

空间复杂度：$O(n)$

1.68 Codeforces 341 E

Description

给$n(n \leq 2000)$个箱子，每个箱子里有一些糖果。糖果总和不超过$m(m \leq 10^6)$。每次可以选定两个箱子i, j 满足$a_i \geq a_j$然后将a_j个糖果从i中移到j中。求一种方案使得最后有且仅有两个箱子有糖果。

Analysis

假如对于任意三个箱子我们都能使其中一个变成空那么只要不断重复这个过程即可剩下两个。

假设三个箱子的糖果数量分别为$A, B, C(A \leq B \leq C)$那么假如我们总能通过一系列操作产生一个小于A的箱子，那么通过不断重复这个过程

就可以完成。设$k = BdivA, t = BmodA$那么只要能够让B减k次A就可以使B变成t从而完成。

每次从B移到A都会使A加倍并且B减少相应数值。假如从C移到A会使A加倍但是B不变。这让我们联想到了快速幂。即t可以表示成2的次幂，对于每一位1都用B去接，对于每一位0都用C去接。由于$B \leq C$这总是可以做到的。使B变成t后再次排序并重复即可完成。

时间复杂度：$O(n \log m)$

空间复杂度：$O(n)$

1.69 Codeforces 342 D

Description

给定一个$3 \times n(n \leq 10000)$的网格，有些格子是障碍，障碍中有且仅有一个特殊障碍。要求用1×2的骨牌覆盖且至少要有一个骨牌能滑动到特殊障碍处。求方案数模$10^9 + 7$。

Analysis

假如不考虑特殊障碍的影响的话可以用一个$O(n)$的DP来解决：$f(i, j)$表示第i行状态为j时的方案数。由于只有3列可以很容易的做到$O(1)$转移。

假如需要考虑特殊障碍的话我们枚举四个方向的骨牌情况再用容斥原理合并即可。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

1.70 Codeforces 346 E

Description

给定$a, p, n, h(a, p, n, h \leq 10^9)$，将$a \times i\%p(1 \leq i \leq n)$，排序后问最远的相邻两个数之间的距离是否不超过$h$。保证$a$和$p$互质。$10^4$组数据。

Analysis

首先先将数字写出来。当$a = 5, p = 23$时：

```
0 5 10 15 20
2 7 12 17 22
4 9 14 19
1 6 11 16 21
3 8 13 18
```

容易发现在$20 \cdots 23$的范围内的距离是不会超过$15 \cdots 19$的。因为即使最后

没有覆盖到20 以上，$15\cdots19$ 中也会有一个和$20\cdots23$中距离相等的对应的gap。所以只需考虑$15\cdots19$即可。

假如由于最后一行不完整导致$0\cdots4$和$15\cdots19$gap不相同那么把最后一行删掉也不会影响答案。所以删完以后$0\cdots4$和$15\cdots19$答案相同只需要看$0\cdots4$即可，即上面列出的数字中的第一列。只要求出第一列的gap大小就是答案。容易发现这又成了一个和原问题类似的问题。于是递归解决即可。

由于a和p互质，所以时间复杂度为$O(\log n)$。证明类似辗转相除法。

时间复杂度：$O(\log n)$

空间复杂度：$O(1)$

1.71 Codeforces 348 E

Description

给定一棵$n(n \leq 10^5)$个点的带边权的树，其中有一些点是黑点。现在要求你移除一个白点，使得尽量多的点能到达的最远黑点距离发生变化。求出最大值和达到最大值的方案。

Analysis

考虑枚举每个黑点x然后求出所有能使这个点变化的白点。容易发现这些点必须处于x到每个最远黑点的路径的交集上。

考虑如何快速地求出交集。首先找到距离最远的两个黑点a,b（如果有多个则任取），并将它们之间的路径称为直径。那么对于每个点x，求出x在直径上的投影y。假如y离a更近，那么最远点只有b；假如y离b更近，那么最远点只有a；假如y离a和b一样近，那么最远点至少含a,b并且交集一定是xa和xb的交集。

求出了交集之后问题转化为了多条线段权值+1和查询点权。这个问题可以用离线lca+dfs或LCT在$O(n \log n)$的时间内解决。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

1.72 Codeforces 351 D

Description

定义一个序列$a_1a_2...a_n$的分解方式：

- 选取$v,t,k(1 \leq v,k \leq n, 0 \leq t, v+tk \leq n)$满足$a_v = a_{v+k}, a_{v+k} = a_{v+2k}...a_{v+tk} = a_{v+(t-1)k}$。

- 将选出的这几个位置的字母删除并将剩下的字母按任意顺序排成一个长度为$n-(t+1)$的串
- 重复第一步直至这个串变成空串

现在给定一个长度为$n(n \leq 100000)$的序列s，和$m(m \leq 100000)$组询问，每次询问给出s的一个子串并询问需要几次才能消除完毕。

Analysis

对于一个串，进行了一次操作以后我们可以将它升序重排，这样相同的字母都在一起，每个字母都只需要消一次。设共有d种不同字母，那么假如第一次操作能够消完某一个字母那么需要消d次，否则需要$d+1$次。

使用莫队算法可以在$O(n\sqrt{n})$的时间内解决。具体做法：维护d很容易做到，只需要开一个数组记录每个字母的出现次数即可。然后我们再维护一个数组代表某个字母的序列中有几处“间隔不等”，即相邻三项相同字母之间的间隔不等。这也可以很容易地维护出来。

虽然这样已经能在CF上通过，但是还有一种更优的做法。考虑枚举右端点，每次处理所有右端点和当前位置相等的询问。那么我们在枚举右端点的时候需要维护每个位置的左端点的答案。具体来说可以维护两棵树状数组，树状数组A 记录每个左端点有多少个不同的字母，树状数组B记录每个左端点有多少个字母不满足等差的条件。仔细观察可以发现A和B在右端点移动了一位以后只会在某一段区间加上1。于是可以轻松用树状数组维护。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

1.73 Codeforces 360 D

Description

给定$n(n \leq 10^4)$个数$a_1, a_2, \cdots, a_n$和$m(m \leq 10^5)$个数$b_1, b_2, \cdots, b_m$ 和一个质数$p(p \leq 10^9)$。 现在有n个集合，每个集合刚开始都只有一个元素1。每次可以从集合i中选出一个元素x，如果$x \times a_i^{b_j} mod p$（j为1到m之间的任意数）不在集合i中则把他加进集合i。直到不能操作为止。求最后所有集合的并集大小。

Analysis

因为p是质数，所以肯定能找到p的一个原根，设它为g。那么$a_i = g^{r_i}$。因为第i个集合里包含的数（模p意义下）是$a_i^{\sum b_j c_j}$ 即$g^{r_i \sum b_j c_j}$即$g^{r_i \sum b_j c_j mod (p-1)}$。

设$t = gcd(b_1, b_2, \cdots, b_m, p-1)$，那么第$i$个集合里的数就是$g^{r_i kt}$，其中$k$为任意自然数。因为$t$是固定的，那么使$g = g^t$，那么第$i$个集合里的数就是$g^{r_i k}$。设$q_i = gcd(r_i, q-1)$，那么第$i$个集合里的数就是$g^{q_i k}$。即假如我们把$g^0, g^1, \cdots, g^{p-2}$排成一排，那么第$i$个集合里的数就是每隔$q_i$个取一个。其中$q_i|(p-1)$。

接下来我们需要找出q_i。容易看出第i个集合的大小是$\frac{q-1}{q_i}$，也等于最小的d_i 使得$a_i^{d_i t} = 1$。 所以之需要找到d_i那么$q_i = \frac{q-1}{d_i}$。

最后我们使用容斥原理算出并集大小即可。注意在计算的时候要算出每个约数的答案而不能只计算d_i的答案。

时间复杂度：$O(n\sqrt{p} + n^2)$

空间复杂度：$O(n)$

1.74 Codeforces 360 E

Description

给出一张n个点m条边的有向图$(n, m \leq 10000)$，每条边的边权有一个上界和一个下界，你可以确定每条边的边权。你刚开始在$s1$，你的对手在$s2$，你们的目的地是$s3$。问能否通过确定边权使你比对手早到。假如不行那么能否能和对手同时到。

Analysis

对于那些到达时间小于对手的点，它的所有出边肯定是最小权（因为对手肯定不会到这个点），反之则是最大权（因为你肯定不会到）。实现的时候类似堆优化Dijkstra，每次取出最小的点，假如这时另一边的人还没到那么边权必然不小于他。所以只需要优先取对手的点即可。判断平局时只需要将到达时间相同的点也改成最小权，即做最短路的时候优先取自己的点即可。

时间复杂度：$O(m \log m)$

空间复杂度：$O(n + m)$

2 Google Code Jam

2.1 GCJ 2008 E

Description

给定一个$m \times n (m, n \leq 50)$的网格，其中每个格子有黑白两种颜色，有些格子的颜色是不确定的。每个黑格能带来4的喜悦值，每两个相邻的黑格会使喜悦值-1。求喜悦值的最大值。

Analysis

首先发现喜悦值就是相邻的不同颜色的格子的个数（包括黑格和边界）。那么在边界上加上一圈白格不会影响答案。

直接求最大值不好求，但是注意到这是网格所以可以二分图染色，即把所有横纵坐标奇偶相等的点黑白翻转，那么答案就变成了总边数-相邻不同色的格子个数。于是只需要求出相邻不同色格子个数的最小值即可。这个问题可以用最小割解决。（连S表示染白色，连T表示染黑色，不确定的格子同时连S和T，然后对于两个相邻的格子之间连代价为1的边，那么假如这两个点归属不同就需要割掉这条边。）

时间复杂度：$O(n^6)$

空间复杂度：$O(n^2)$

2.2 GCJ 2009 A

Description

在一年里，有$T(T \leq 50)$场活动。这些活动的开始日期还没有确定，但是日程安排已经公布（即活动开始后的每一天是否有比赛）。在某些情况下，一天可能有多场比赛。对于每一个拥有S个比赛的日子，你会获得S^2点愉悦值。初始愉悦值为0。 这一年有$N(N \leq 10^9)$天，每次活动的开始日期等概率地分布在每一天上（即：在每一天开始的概率都是$\frac{1}{N}$）。现在你想知道愉悦值的期望。以最简带分数形式输出。假如一次活动的某次比赛的日期在第二年，那么不应该被算入总愉悦值内。每次活动只有不超过$m(m \leq 50)$场比赛，且最晚的比赛不会超过活动开始后10000天。活动列表d满足$1 \leq d_2 \leq d_3 \leq \cdots \leq d_m \leq 10000$。

Analysis

设x_i为第i天的比赛数，$D_{i,j}$表示第j个活动开始时间在i之前的比赛有几个，$a_{i,j}$为第i次活动的第j次比赛在哪一天举行，$Y_{i,j}$表示第j次活动有几场比赛在第i天举行（显然$0 \leq Y_{i,j} \leq 1$），显然$x_i = \sum_j Y_{i,j}$，则题目所求为：

$$
\begin{aligned}
E(\sum_{i=1}^{N} x_i^2) &= \sum_{i=1}^{N} E(x_i^2) \\
&= \sum_{i=1}^{N} (\sum_{j=1}^{T} E(Y_{i,j}^2) + 2 \times \sum_{j=1}^{T} \sum_{k=j+1}^{T} E(Y_{i,j} Y_{i,k})) \\
&= \sum_{i=1}^{N} (\sum_{j=1}^{T} E(Y_{i,j}) + 2 \times \sum_{j=1}^{T} \sum_{k=j+1}^{T} E(Y_{i,j} Y_{i,k})) \\
&= \sum_{i=1}^{N} (\sum_{j=1}^{T} \frac{D_{i,j}}{N} + \frac{\sum_{j=1}^{T} \sum_{k=1}^{T} D_{i,j} D_{i,k} [j \neq k]}{N^2}) \\
&= \frac{\sum_{j=1}^{T} \sum_{u=1}^{m_j} (N - a_{j,u} + 1)}{N} + \sum_{i=1}^{N} \frac{\sum_{j=1}^{T} \sum_{k=1}^{T} \sum_{u=1}^{m_j} \sum_{v=1}^{m_k} [a_{j,u} \leq i] [a_{k,v} \leq i] [j \neq k]}{N^2} \\
&= \frac{\sum_{j=1}^{T} \sum_{u=1}^{m_j} (N - a_{j,u} + 1)}{N} + \frac{\sum_{j=1}^{T} \sum_{k=1}^{T} \sum_{u=1}^{m_j} \sum_{v=1}^{m_k} (N - \max(a_{j,u}, a_{k,v}) + 1) [j \neq k]}{N^2}
\end{aligned}
$$

这里有一个小技巧可以避免使用高精度整数：将式子通分以后分母是N^2，于是我们可以只记整数部分和分子最后约分就可以避免高精度整数。

时间复杂度：$O(m^2T^2 + \log N)$

空间复杂度：$O(mT)$

2.3 GCJ 2009 B

Description

给定平面上的$n(n \leq 10^5)$个点，求一个周长最小的三角形。

Analysis

类似平面最近点对的方法，首先将点按x坐标均等地分成两份并递归计算，然后计算出公用两边点的答案。设递归计算的答案为p，那么离均分线距离超过$p/2$ 的点都可以忽略。将选出的点投影到均分线上并按y坐标扫过来，对于每个点只枚举y坐标和它相差不超过$p/2$的。可以证明满足条件的点不会很多，是常数级别的。于是可以$O(n^2)$枚举和哪两个点组成三角形而不会影响复杂度。这样做的时间复杂度是$O(n \log^2 n)$，空间是$O(n)$的。但是假如每次传入已经按y排序好的数组就可以把时间复杂度降到$O(n \log n)$但是空间需要$O(n \log n)$。

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

2.4 GCJ 2009 D

Description

给定$n(n \leq 500)$个平面上的塔，每个塔可以选择升级或不升级。升级需要v_i的费用，但是一个塔升级后和它距离不超过r_i的塔也必须升级。求最大收益。

Analysis

经典的最大权闭合子图问题，使用最小割解决。（每个点和S和T都连并在收益少的那边加上代价，然后对于两个能影响到的点连无穷大的边。）
时间复杂度：$O(n^4)$
空间复杂度：$O(n^2)$

2.5 GCJ 2010 A

Description

有一个栈，每次可以压进一个字符，弹出一个字符或者打印栈顶字符。现在有一个长度为$n(n \leq 2000)$的只含ABC的串，求用这个栈打印这个串至少需要几步。栈在工作结束之后必须是空的。

Analysis

首先注意到若栈顶就是这个字符那么这一步执行打印肯定是最优解之一。若栈顶下面一个就是这个字符那么弹出栈顶也不会差于压入这个字符。所以栈中不会有两个相同的字符之间隔着小于2个字符。所以栈中的字符一定是$XYZXYZXYZ\cdots$形式的。于是栈的状态只有$O(n)$种，使用$O(n^2)$的DP即可。
时间复杂度：$O(n^2)$
空间复杂度：$O(n)$

2.6 GCJ 2010 C

Description

有$k(k \leq 1000)$个顾客要买若干个（不超过$C(C \leq 10^{12})$个）糖果，问你至少需要打包多少箱糖果，使得无论如何都可以满足每个顾客的要求。

Analysis

先给出正确的做法：每次打包一份floor(当前已打包糖果数/k)+1直到总糖果数不小于kC。下面来证明为什么是正确的。

必要性：假如有一种方案比这种方案的答案小，那么一定存在一个数t使得floor(包大小小于t的包糖果和/k)+1$\leq t$。那么k个t的顾客就无法满足要求。

充分性：只需证明：将包从小到大排序后，前k包可以满足任意一种要求总和不大于包内总和的方案。这可以用数学归纳法证明。$k=1$时显然成立。假如$k=p-1$时成立，那么当$k=p$时，设前$k-1$ 个包的和为n，则第k个包为$\lfloor \frac{n}{k} \rfloor + 1$，这样前$k$个包总和就是$n + \lfloor \frac{n}{k} \rfloor + 1$，根据抽屉原理，最大的顾客大小至少是$\lfloor \frac{n}{k} \rfloor + 1$。所以我们为最大的顾客分配第$k$个包，于是问题转化为了$k-1$的情况，所以$k=p$的时候也成立。

时间复杂度：$O(\log(kC))$

空间复杂度：$O(1)$

2.7 GCJ 2011 A

Description

给定一个字符串$s(|s| \leq 450000)$，将每一段极大的字母全部相同的子串称为一个“Run”。问将有多少种s重排列以后的方案使得重排后“Run”的数量不变。原串中的“Run”的数量不超过$m(m \leq 100)$。答案模1000003。

Analysis

考虑动态规划:$f_{i,j}$表示前i种字母有j个“Run”的方案数。转移的时候枚举这个字母分成的段数和这些段中嵌在原来的“Run”里面的个数，乘上组合数转移即可。组合数使用预处理阶乘即可$O(1)$计算。

时间复杂度：$O(|\Sigma|m^3 + |s|)$

空间复杂度：$O(m)$

2.8 GCJ 2013 E

Description

给定一个长度为$n(n \leq 2000)$的序列，每个元素都不超过10000。你可以删掉序列中的一个元素直到整个序列单调不增。求方案数。答案模10007。

Analysis

考虑对于每个单调不增的子序列s计算以这个子序列结尾的方案数。设这个子序列长度为t，那么方案数就是$(n-t)!$减去那些在之前就已经单调不增的方案（设为p_s）。但是直接算比较麻烦，考虑用每个序列减去比它短1的序列的p_s值。那么每个子序列的贡献就是$(1-t)(n-t)!$。那么只需要求出每个长度的子序列数量即可。

考虑动态规划：$f_{i,j}$表示长度为i最后一个大小为j的子序列个数，转移的时候利用树状数组做到$O(\log n)$转移即可。

时间复杂度：$O(n^2 \log n)$

空间复杂度：$O(n)$

2.9 GCJ 2014 C

Description

给你一个$n(n \leq 10000)$个节点顶点染色的树，问能不能将它放在一个平面上满足边不相交且沿一条直线轴对称。

Analysis

对称有两种情况：

- 对称轴不和任意一条边重合。这种情况下对称轴平分了一条边并且这条边两边的子树全等。只需要枚举每一条边判断即可。
- 对称轴和树上的一条链重合。这时链上的每一条边连接的两个子树都满足“可对称”性质。一个子树可对称当且仅当它的子树可以两两配对并且没有多出来的或者只多出来一个可对称的子树。只需要枚举每一条边判断是否两边都满足条件即可。

由于子树的数量是$O(n)$的，计算每个子树的hash值或是否可对称是均摊$O(\log n)$的，总复杂度为$O(n \log n)$。

hash的方法有很多，我选用的是$hash_x = \prod_{y \in son_x} hash_y + v_x$。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

3 USA Computing Olympiad

3.1 USACO 06 DEC

Description

给定一个长度为$n(n \leq 100000)$的匹配串s和一个长度为$m(m \leq 25000)$的

模板串S，问匹配串中有哪些长度为m 的子串满足和模板串的大小关系相同（即离散后结果相同）。

Analysis

假如已经知道某个串$a_1 \cdots a_p$和模板串前p位同构，那么假如a_{p+1}满足以下条件，$a_1 \cdots a_{p+1}$就和模板串前$p+1$位同构。

- a_{p+1}和a中下标为 $S_1 \cdots S_p$中第一个和S_{p+1}相等的位置 的数字相等。
- a_{p+1}大于a中下标为 $S_1 \cdots S_p$中比S_{p+1}小的最大数的位置 的数字。
- a_{p+1}小于a中下标为 $S_1 \cdots S_p$中比S_{p+1}大的最小数的位置 的数字。

所以只需要预处理出每一位的对应的三个下标，就可以在$O(1)$的时间内进行一位的匹配。

这样我们就可以利用KMP算法优化这个匹配：先求出模板串自身匹配后的失配位置就可以在$O(n+m)$的时间内解决这个问题。

时间复杂度：$O(n+m)$

空间复杂度：$O(n+m)$

3.2 USACO 07 DEC

Description

给定一个长度为$n(n \leq 30000)$的字符串，每次可以从头里拿出一个字符或者从尾拿出一个字符并加到答案串的末尾。求将所有字符拿完后字典序最小的答案串。

Analysis

考虑暴力做法：每次操作比较最左边和最右边的字符，如果大小不一样那么就拿出小的那个。否则接着比较里面一位的字符，直到比较出大小为止。容易发现拿较小的那头的不会更差。

所以每次实际上是将正着的串和反着的串进行大小比较。这可以用后缀数组在$O(n \log n)$的时间内轻松完成。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

3.3 USACO 07 OPEN

Description

给定一个$2 * n(n \leq 15000)$的网格，相邻两点之间可以连边。有$m(m \leq 50000)$次操作，每次操作可以加一条边，删一条边或查询两个点是否能够不走到两点所在的列之外相互到达。强制在线。

Analysis

考虑使用线段树维护区间内左端点和右端点的连通情况（上上，上下，下上，下下）。那么每次修改只需要修改$O(\log n)$个节点。查询的时候只需要将区间分成$O(\log n)$个部分然后合并在一起即可。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

3.4 USACO 08 MAR

Description

有$n(n \leq 50000)$块土地，每块土地有长L_i和宽W_i。每次可以购买任意块土地，代价为$\max L_i * \max W_i$。求购买所有土地的最小代价。

Analysis

首先那些可以被另外一块地完美覆盖（长宽都不大于另一块地）的地可以忽略。然后按照L_i升序排序后，W_i为降序。

考虑朴素$O(n^2)$DP:$f_i = \max(f_j + W_{j+1} * L_i)(j \leq i)$。注意到这个方程是满足斜率优化的，我们把$(f_j, W_{j+1})$ 看成一个点并维护一个下凸壳，那么每次在凸壳上二分即可。注意到在这一题中L_i不减，所以上一次忽略的点这一次也会忽略，所以直接维护单调队列可以做到$O(n)$，不过由于排序的原因不影响复杂度。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

3.5 USACO 08 NOV

Description

有$n(n \leq 100000)$天，第i天需要a_i个玩具。每次可以从商店里买新玩具，或者将用过的玩具送去清洗得到可用的玩具。有两种清洗方式，分别需要$N1$天和$N2$天，单价分别为$C1$和$C2$。 所有价格都不超过60。问最少要花多少钱才能满足要求。

Analysis

容易发现买新玩具必定是在开始的时候一次性买完。假设确定了刚开始要买的玩具数量，那么对于每一天可以贪心选取：首先先使用新玩具，没有新玩具了就使用较便宜的清洗方式中时间最近的那些玩具，还是不够就是用较贵的清洗方式中的时间最近的玩具，还是不够就无解。这可以利用两个队列存下不够快洗的玩具和够快洗不够慢洗的玩具，再用一个变量存下够慢洗的玩具数量来做到线性。

设$f(x)$为开始x个玩具情况下的清洗代价，容易发现$f(x+1)-f(x) <= f(x)-f(x-1)$。因为新加一个玩具造成的代价减少上一个加的玩具也可以做到。假如不满足的话我们可以将第x个玩具用第$x-1$ 个玩具的方式来减少代价。所以$f(x)$是一个下凸函数。因为买玩具代价的函数$g(x)=x*C$显然是一个下凸函数。所以两个函数相加仍然是一个下凸函数。所以我们可以使用三分找出最小值。

时间复杂度：$O(n\log(n\times maxC))$

空间复杂度：$O(n)$

3.6 USACO 08 OPEN

Description

给定平面上$n(n\leq 10^5)$个点，并在曼哈顿距离小于m的点之间连边，求连通块个数和最大的连通块大小。

Analysis

由最小生成树的环切性质可得：求出原图的最小生成森林后连通性和原图一致。于是只需要求出曼哈顿距离最小生成树即可。

曼哈顿距离最小生成树有一个性质：每个点只有可能向八个方向（每45°为分割）中离自己最近的点连边。假如最小生成树是连另外一个点的话可以很容易证明换成最近的那个点向较远点连不会更差。找最近的点可以用离线+扫描线+离散+树状数组在$O(n\log n)$的时间内轻松完成。

那么边数就减小到了$O(n)$级别，使用Kruskal即可在$O(n\alpha(n))$的时间内完成。

时间复杂度：$O(n\log n)$

空间复杂度：$O(n)$

3.7 USACO 09 MAR

Description

给定$n(n\leq 40000)$个数（每个数不超过$m(n\leq 40000)$），要求将这些数分成一些块，代价为每块内不同数字数的平方和。求最小代价。

Analysis

考虑暴力DP：$f(i)$表示前i个数的最小代价，转移的时候枚举前一段断点$O(n^2)$转移（需要预处理出两点之间不同数字数）。但是这样会有很多无用状态。注意到$f(i)$不降，那么假如$f(i)$可以从$f(j)$转移过来并且从$f(j-1)$转移过来这一段的不同数字数还是相等那么从$f(j-1)$转移肯定不会更差。所以可以预处理出满足上一段不同数字的个数为t时的最左点进行转移。但是这样状态数仍然是$O(n^2)$的需要进一步优化。由于每个数独自分一段答案就是n，所以不可能有一段的数字个数超过$\sqrt{n}$。这样要预处理的数量就降到了$O(n\sqrt{n})$，足以通过本题。

时间复杂度：$O(n\sqrt{n})$

空间复杂度：$O(n\sqrt{n}+m)$

3.8 USACO 09 OPEN

Description

给定一个长度为$n(n \leq 100000)$的序列，要求将序列分割成尽可能多的块，满足从左到右块内和单调不增。

Analysis

这道题直接贪心是错误的。

先将序列翻转，那么要求就变成了单调不降。考虑DP：$f(i)$表示前i个最多分割成多少块。容易发现$f(i)$单调不降。用$g(i)$表示分割的时候最优情况下第一块分到的位置。因为$f(i)$不降，所以我们需要尽量最大化$g(i)$。那么就得到了一个显然的$O(n^2)$DP：$O(n)$枚举从哪个位置转移。

这个DP是可以利用单调队列优化的：用$s[i]$表示前i个数的和，$p(i)$表示$s[g(i)]-s[i-1]$，那么$g(i)$要满足的条件是：$s[g(i)-1]-s[i-1]>=p(g(i))$即$s[g(i)-1]-p(g(i))>=s[i-1]$。所以所有满足$i>j$且$s[g(i)-1]-p(g(i))>=s[g(j)-1]-p(g(j))$的$j$可以忽略不计；所有满足$i>j$且$i$和$j$都可以成为答案的$j$都可以忽略不计。

时间复杂度：$O(n)$

空间复杂度：$O(n)$

3.9 USACO 10 DEC

Description

给定一个模板串$s(|s| \leq 50000)$，再给定一个样本串$t(|t| \leq 50000)$，要求将t分成尽量少的段使得每一段都是s的子串。保证有解。

Analysis

设f_i为最大的j使得$t_{i...j}$是s的子串。显然f_i单调不降。

那么我们就可以设计一个贪心算法，每次取尽量长的一段，由于f_i不降，容易看出取得长不会比取得短更差。

用后缀自动机可以轻松的实现寻找匹配串的工作。

时间复杂度：$O(m+n)$

空间复杂度：$O(n|\Sigma|)$

3.10 USACO 10 OPEN

Description

给定平面上$n(n \leq 10^5)$个点，求有多少个三角形包含原点。保证给出的点中不含原点并且没有两个点连线经过原点。

Analysis

只要求出有多少个三角形不包含原点然后用总数去减即可。对于每个不经过原点的三角形我们按顺时针方向找到的第一个点满足剩下两个点都在这个点顺时针方向$180°$以内。所以我们可以先按极角序排序，然后枚举每个点作为顺时针第一个点，用two pointer维护可行的区间并计算答案。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

3.11 USACO 12 JAN

Description

给定$n(n \leq 14)$组牌，每组牌包含8张。两个人轮流行动，进行n轮游戏，第i轮用第i组牌。一开始有一个变量$x = 0$，每轮第一个人选择前四张或后四张，第二个人选择四张中的前二张或后二张，设为a,b，那么x变为$(x \times a + b) mod m$。假如n轮结束以后$x \geq t$且$x \geq m - t$则第一个人获胜。现在给出第二个人每轮的策略，问第一个人在保证必胜的情况下字典序最小的方案。保证第一个人存在必胜策略。

Analysis

对于每一步判断选择后四张能不能必胜，可以就选后四张否则选前四张。那么只需要快速的判断一个局面是否必胜即可。

由于双方轮流行动，可以使用博弈搜索：一个局面必胜当且仅当有一个后继状态必败。直接做会被构造数据卡成$O(4^n)$，所以每次随机先判断哪个决策就不能被卡了。

时间复杂度：指数级
空间复杂度：$O(n)$

3.12 USACO 12 MAR

Description

有$n(n \leq 18)$个物品，要求分成尽可能少的份数使得每一份的质量和都不超过m。

Analysis

状压DP：用二进制状态来表示拿了这些物品以后最少分成几份且最后一份中剩下的大小。转移的时候枚举上一个状态$O(n)$ 转移。
时间复杂度：$O(n2^n)$
空间复杂度：$O(2^n)$

3.13 USACO 12 DEC Gangs of Instanbull

Description

有$n(n \leq 1000000)$头奶牛和$m(m \leq n)$个帮派，每个奶牛都属于其中一个帮派。

奶牛们正在争夺一个大牧场的控制权：他们会按一定的顺序进入牧场，每进来一头牛，假如牧场里没有奶牛或只有他们帮派的奶牛，那么他会占领这个牧场。否则他会和其中一头奶牛同归于尽。求最后能否使1号帮派占领牧场。如果可以输出最后剩余的最多奶牛数目并且输出字典序最小的方案。

Analysis

首先每次可以让其他两个帮派个损失一头牛，所以最后剩下的牛个数等于max(其他帮派牛总数%2，其他帮派中牛最多的牛数*2-其他帮派中总牛数)。这是至少需要损失的牛数。这样可以解决前两问。

输出方案的时候首先需要特判掉$n = 2$的情况：先出动所有1再出动所有2。

否则则先派出那些肯定要损失的牛，然后从小到大派出牛直到有一个帮派牛的个数已经占到除了那些最后占领牧场的牛之外的牛中的一半。那么这时候必须让其他帮派的每一头牛都和最多的哪个帮派同归于尽。所以先使用最多的帮派将牧场上剩余的牛清空，然后对于编号比他小的帮派先派出整只然后用最多的那个帮派消灭。剩下的按字典序派出即可。最后再派出帮派1 的剩余牛。容易证明这样的字典序最小。

时间复杂度：$O(n)$
空间复杂度：$O(n)$

3.14 USACO 12 DEC First!

Description

给定$n(n \leq 30000)$个字符串，总长$m \leq 300000$，要求任意改变字母大小顺序后所有有可能成为字典序最小的串。

Analysis

一个串要成为最小串的条件是：它的前缀都不是串且其他的串的第一个和它不同的字母都大于它的。

那么我们把这些串加到一个Trie里，然后遍历这棵Trie并同时维护字母间应该满足的大小关系。每次找到一个串后就进行一次拓扑排序找环并返回（因为它的子树都不可能最小）。

时间复杂度：$O(m|\Sigma|)$
空间复杂度：$O(m|\Sigma|)$

3.15 USACO 13 MAR

Description

给定$n(n \leq 10^5)$条线段，保证第一条线段的左端点是$(0,0)$且没有线段相交。现在有个人从第一条线段左端点开始走向右端点，走到右端点后自由下落到下面一条线段并继续行走。若下面没有线段则结束。问能走过几条线段。

Analysis

假如我们处理出了每条线段下落到的线段那么就可以$O(n)$模拟一遍行走过程。

考虑扫描线：用平衡树维护当前线段的高低位置关系，每次删除一条线的时候在平衡树中找到下面的那个线段并将这条线段指向那个线段。原题中有许多边界问题需要考虑，如：左边界可站右边界不可站。这就需要x坐标相同的先插入后删除并且删除的时候先统一删除再更新答案。

时间复杂度：$O(n \log n)$
空间复杂度：$O(n)$

3.16 USACO 13 OPEN Photo

Description

给定$m(m \leq 2 \times 10^5)$个点和$n(n \leq 10^5)$个区间，问最多能将多少个点染色使得每个区间内染色点有且仅有一个。无解输出-1。

Analysis

考虑暴力$O(n^2)$DP:f_i表示第i个点染色且$1 \cdots i$范围内满足要求的最多数量。那么$f_j = 1 + \max(DP[i] : \max(b_j : a_j \leq x) \leq i \leq min(b_j : x \leq a_j))$。

假如考虑每个f_i能转移到的范围就可以发现其实是一个区间。可以$O(n)$求出每个f_i能转移到的区间，然后更新f_i后使用线段树维护转移即可。

时间复杂度：$O(n \log n)$

空间复杂度：$O(n)$

3.17 USACO 13 OPEN Figure Eight

Description

给定一个$n \times n(n \leq 300)$的01矩阵，你需要在里面画出两个矩形a,b使得满足以下条件：

- a的底边被b的顶边包含
- 矩形的边长至少是2
- 矩形边界上不能有1

最大化$(l_a - 2)^2 \times (l_b - 2)^2$（即两个矩形除了四条边外面积的乘积）。假如无解输出-1。

Analysis

首先$O(n^2)$处理出每一行的前缀和这样就可以$O(1)$判断一条水平线段是否包含1。

然后$O(n^3)$处理出以每一条水平线段为底边的最大面积$ftop$。同时处理出以每一条水平线段为顶边的最大面积$fbot$。

最后$O(n^3)$枚举b的顶边并维护$ftop$的区间最大值和$fbot$的对应位置相乘更新答案。

时间复杂度：$O(n^3)$

空间复杂度：$O(n^3)$