IOI2015 中国国家集训队第一次作业

Codeforces/USACO Monthly Contest/Google Code Jam Finals

试题泛做题解

姓名：束欣凯（SHUXK97）[1]

0 前言

忙活了很久终于做完了呢。（长舒一口气）

这次的泛做题解一共有 102 道题，算是刚刚好完成任务吧。虽然数量比不上那些真正的神犇（去年花神 150/150 实在是太可怕了，今年一定也有），但是也倾注了很多心血呢：

首先是做这 102 道题（在这里感谢所有 53 名集训队员贡献了这些题目）。每一道题都是认认真真地去做的。几个月里每天都在电脑前花费几个小时，看过一页又一页的题目，用完一张又一张的草稿纸，敲出一行又一行的代码，最后转化成清橙作业页面的一个个“正确”。值了！

内容的撰写也花了不少的功夫：每道题的题面尽量做到简洁明晰，题解尽量做到简明易懂（尽管如此，纯文字量也超过 100K 了呢）；题目中需要用到的结论，如果直观上不是比较显然，都在后面赋了证明（有几个结论的证明头疼了我几天）；公式是一个一个在 Word 2010 中手敲出来的；排版没有用 Latex，而是选择了我比较喜欢的格式（不知道大家喜欢吗）。

然后评价一下今年选的题目，比去年质量要高。去年选的 ACM/ICPC World Finals 试题签到题多，搜索题多，几何题多，与目前的 OI 贴合得不是很紧密（但是也有很多好题）。今年选择的 Codeforces/USACO Monthly Contest/Google Code Jam Finals 试题更加适合现在的 OI，既有传统的 DP、贪心、图论、数据结构、字符串题，又有新颖的非传统题，而且对思维的要求更高，时常有一些巧妙的思路让人拍案叫绝。

做这些题目对我来说是极大的锻炼和提高，我非常喜欢。

最后是一些感言。

六年 OI 路，且行且珍惜。作为集训队中的弱菜，冬令营以后，我的 OI 生涯也要结束了。蓦然回首，从普及组时的懵懂，到初得金牌时的喜悦，再到现在的淡然。六年间，OI 改变了我许多。通过 OI，我学会了严谨思考，学会了淡然处世，收获了许多快乐，也交到了许多朋友。这段 OI 生涯将成为我心中一段美好的回忆。真的猛士，将更奋然而前行！

同时也希望这篇泛做题解能给后生提供一些帮助。

[1] ID：SHUXK。联系方式：shuxinkai@126.com。

目录

1 Codeforces 试题泛做题解

1.1 CF235C Cyclical Quest

题目大意

WJMZBMR 有一个字符串s和N个字符串x_i。他要对每个x_i询问，在s中有多少个子串与x_i“旋转同构”。“旋转同构”指将一个字符串开头的若干字符移至结尾处可以得到另一个字符串。（当然直接同构也是算的）

$|s| \le 10^6$，$N \le 10^5$，$Sum = \sum_{i=1}^{N}|x_i| \le 10^6$，时间限制 3s。

简要题解

这道题一看就要用后缀数据结构来做。这里选用性价比最高的后缀自动机。

我们先建立起s的后缀自动机，然后将每个x_i在后缀自动机上面走。整个字符串走完了以后，我们查询这个状态在s中出现的次数。（预处理每个状态在s中出现的次数是O(N)的）

由于“旋转同构”的存在，我们还要维护从头删去一个字符和在结尾加上一个字符之后在后缀自动机上的结点，用后缀自动机实现这两个操作比较简便（正因为如此才用后缀自动机啊）。

然后要考虑x_i自身是周期性串的问题。我的解决方法是：如果x_i的两个“旋转串”在后缀自动机上走到了相同的结点，就说明它们是一样的串。此时就应该停止（因为后面也一定是以前出现过的串了）。

时间复杂度$O(|s| + Sum)$，空间复杂度$O(52|s| + Sum)$。

1.2 CF235E Number Challenge

题目大意

定义d(n)为n的约数个数。现给定a,b,c，求$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} d(ijk)$。

$a,b,c \le 2000$，时间限制 3s。

简要题解

这道题是 WJMZBMR 出的，在比赛中 rng_58 给出了一个神奇的式子：

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{c} d(ijk) = \sum_{(i,j)=(j,k)=(k,i)=1} \left\lfloor\frac{a}{i}\right\rfloor\left\lfloor\frac{b}{j}\right\rfloor\left\lfloor\frac{c}{k}\right\rfloor$$

证明：我们只需证$d(abc) = \sum_{i|a,j|b,k|c,(i,j)=(j,k)=(k,i)=1} 1$。

左右两边都是积性函数，所以只需要看一个质因子p。

设$v_p(a) = x, v_p(b) = y, v_p(c) = z$（$v_p(a)$表示 a 的质因子分解中p的次数），

则左式 $= x + y + z + 1$，右式 $= x + y + z + 1$（枚举i,j,k的值即可得出结论）。

等式成立。

现在利用这个等式来做这道题。直接枚举i,j,k肯定不行，然后就要用莫比乌斯反演降低时间复杂度：

$$\sum_{(k,ij)=1}\left\lfloor\frac{c}{k}\right\rfloor=\sum_{k=1}^{c}\left\lfloor\frac{c}{k}\right\rfloor\sum_{d|(ij,k)}\mu(d)=\sum_{d=1,d|ij}^{c}\mu(d)\sum_{k=1}^{\left\lfloor\frac{c}{d}\right\rfloor}\left\lfloor\frac{c}{kd}\right\rfloor$$

我们要先预处理$f(d)=\sum_{k=1}^{\left\lfloor\frac{c}{d}\right\rfloor}\left\lfloor\frac{c}{kd}\right\rfloor$，再预处理出$G(K)=\sum_{d=1,d|K}^{c}\mu(d)f(d)$，然后就能够枚举满足$(i,j)=1$的$i,j$来做了。

时间复杂度$O(ablogc)$，空间复杂度$O(ab)$。

1.3 CF238D Tape Programming

题目大意

一个程序是由数字和“<”“>”构成的非空串。程序运行时有一个指针。最开始指针的指向最左字符，移动方向为向右。我们重复以下操作直到指针指向串外时停止：

1. 如果指针指向一个数字，输出它再将其减一，指针沿原方向移动一格。如果原数字为 0 则删除这个数字；
2. 如果指针指向“<”或“>”，那么指针的移动方向改为尖角方向，接着指针沿新方向移动一格。如果新的位置也是“<”或“>”，则删除原来的字符。

现在有一个长度为N的程序以及q个询问，每个询问给出$[l_i,r_i]$，问如果运行$[l_i,r_i]$之间的程序每个数字会输出多少次。

$N,q\le 10^5$，时间限制 1s。

简要题解

这道题的思路还是很神的：如果我们运行$1\sim N$的程序，那么第一次进入任何一个区间都一定是从这个区间的左端点进入的，而且方向向右（这个结论想想还是显然的）。在进入这个区间之后第一次离开这个区间时，我们就将这个区间的程序运行完了。

于是问题就变成了运行$1\sim N$的程序，查询第一次进入$[l_i,r_i]$时输出$0\sim 9$的次数，然后再查询之后第一次离开$[l_i,r_i]$时输出$0\sim 9$的次数就可以了。

这个可以用前缀和维护输出$0\sim 9$的次数，用线段树维护第一次离开某一个区间的时间。

真的结束了吗？并没有，因为运行一遍$1\sim N$的程序有可能有些格子到不了。这时就需要再运行若干次，直到所有的地方都已经到过了为止。

具体实现时，可以令$A[0]='>',A[N+1]='<'$，这样就不会出现死循环的情况。

时间复杂度$O(10NlogN)$，空间复杂度$O(100N)$。

1.4 CF238E Meeting Her

题目大意

在一个N个点M条边（每条边长度为 1）的有向图中有K路公交车。每路公交车从S_i出发到T_i，并且沿着最短路径走（如果有多种最短路径，则任选一种）。

Urpal 要从a走到b，求最坏情况下他最少需要做几次公交车才能到达。

$N\le 100$，$K\le 100$，时间限制 100ms。

简要题解

我们首先要计算的是每对S_i和T_i之间的最短距离，然后进行迭代。

设Ans_i表示从i出发到终点b最少需要做公交车的次数，初始时$Ans_b = 0$，其余为无限大。每一次迭代时对每一条公交线路这样考虑：

如果一个结点有一条公交线路一定会经过，并且经过它以后的所有路径都能到达Ans值不是无穷大的结点，则更新这个结点的Ans值。具体的操作方法只要对每一条公交线路从尾到头考虑即可。

如何判断一条公交线路一定会经过某个点（设它与起点距离为d）？由于边权均为 1，那么就可以看最短路径树上是否有其它与起点距离也为d的点。

时间复杂度$O(N^3 + N^2K + M)$，空间复杂度$O(N^2 + NK)$。

1.5 CF240E Road Repairs

题目大意

有一个N个结点M条边的有向图，现在有一些边损坏了。现在要修复一些边使得从 1 出发可以到达所有的结点，问最少要修复几条边，并求出方案。

$N, M \leq 10^5$，时间限制 1s。

简要题解

这是裸的最小树形图了，我们直接用朱刘算法就可以了。

于是我现场学习了一下朱刘算法。由于是初学，时间和空间复杂度都惨不忍睹，就不要嘲笑我了。

我也发现了一些很奇怪的事情：在 Codeforces 上有很多莫名其妙（我太弱了，完全不知道是怎么过的）的程序 AC 了，然后看到清橙上的出题人加强了数据，卡掉了奇怪的方法。庆祝一下！然后我新学的朱刘在 AC 了清橙数据之后在 CF 上狂 WA 不止，后来发现我写错了若干个地方（怎么过的）。最后改了两个小时才过掉 CF 上的数据。囧！

时间复杂度$O(NM)$，空间复杂度$O(NM)$。

1.6 CF240F Torcoder

题目大意

给定一个长度为N的字符串，要求进行M次操作。每一次操作将$[L_i, R_i]$的子串进行字符重排，排成能得到的字典序最小的回文串。每次操作结束后不撤销。

求M次操作过后的字符串。

$N, M \leq 10^5$，时间限制 3s。

简要题解

首先确定一个串如何重排成回文串：

如果有多于一个字母有奇数个，显然不行；如果有一个字母有奇数个，将其中的一个放在最中间。剩下来的按照字母顺序从外向内排。

我们用线段树来维护一段区间内每个字母出现的次数，判断能否重排就相当于线段树查询，重排成回文串就相当于进行至多 26 次区间覆盖。这样就可以了。

时间复杂度$O(N + 26M\log N)$，空间复杂度$O(26N)$。

1.7 CF241B Friends

题目大意

N个非负整数A_i两两异或共有$\frac{N(N-1)}{2}$个值，求前M大异或值的总和。

$N \leq 50000$，$M \leq \frac{N(N-1)}{2}$，$0 \leq A_i \leq 10^9$，时间限制 200ms。

简要题解

核心思想是从高到低考虑第M大异或和S的每一位，与此同时统计求和。

用[??*]xor[##*]表示某两个集合中的数的互相 xor（如果??与##相等则表示某一个集合中的数的互相 xor），那么考虑每一位时都只要考虑若干个[??*]xor[##*]（比如说考虑最高位时只需要考虑 0 开头的数 xor 1 开头的数，即[0*]xor[1*]）。

在从高到低的考虑到第i位时，利用之前已经分出来的所有[??*]xor[##*]求出这一位为 1 的异或值个数Tot与总和Sum（此时只考虑第 i 位~第 0 位的总和）。

如果$Tot \geq M$, 则S的第i位为 1，此时 $Ans += M \times 2^i$，在考虑以后的位时这样分划：

$$[??*]xor[##*] \rightarrow [??0*]xor[##1*]和[??1*]xor[##0*]$$

如果$Tot < M$，则S的第i位为 0，此时$Ans += Sum$，$M -= Tot$，在考虑以后的位时这样分划：

$$[??*]xor[##*] \rightarrow [??0*]xor[##0*]和[??1*]xor[##1*]$$

这样，从最高位一直考虑到最低位就可以求出Ans了。

接下来就是如何快速计算每一个[??*]xor[##*]对Tot和Sum的贡献，这个可以通过预处理在O(30)的时间内做到：将A_i排序，那么每一个[??*]和[##*]就代表一段连续的A_i，只需要预处理A_i每一位数字的前缀和即可。

时间复杂度$O(30^2N)$，空间复杂度O(30N)。

这种方法实际效果非常快，足以通过 200ms 的时间限制。

1.8 CF241D Numbers

题目大意

有一个1~N的排列，要从中选出一个非空子序列满足：

1. 选出的元素异或和为 0；
2. 如果将选出的元素从左到右无间隔地写在一行组成一个十进制数，那么这个数是一个质数p的倍数。

求一个方案。

$N, p \leq 50000$，时间限制 1s。

简要题解

这道题做法比较猎奇：只需要考虑1~31之间的整数就可以了。为什么？

我们大致想象一下：选出来的子序列一共有2^{31}个，满足异或和为 0 的约占$\frac{1}{32}$，即2^{26}个。

如果$p \neq 2$且$p \neq 5$，那么这2^{26}个数大致呈随机分布，有解的概率非常接近于 1。

如果$p = 2$或$p = 5$，由于p太小导致几乎不可能构造出能卡掉的情况。

时间复杂度$O(31^2p)$，空间复杂度$O(31p)$。

1.9 CF241E Flight

题目大意

给定一个N个点M条边的有向无环图，每条边的长度均为 1。你可以将某些边的长度改成 2，使得从 1 到N的所有路径长度都相等。求一个方案。

$N \leq 1000$，$M \leq 5000$，时间限制 1s。

简要题解

首先正反两遍 DFS 找出从 1 到N的路径上可能走到的所有边，以后就只要考虑这些边。

设$d[i]$表示点 1 到点i的距离，那么对于一条要考虑的边(x,y)，就可以得到不等式$1 \leq d[y] - d[x] \leq 2$。

这是一个类似于差分约束的东西，我们就按照求最短路的方法进行迭代。如果有解，最多只要对每条边迭代N次就能收敛；否则N次迭代之后仍然会有矛盾。

时间复杂度$O(NM)$，空间复杂度$O(N+M)$。

1.10 CF241F Race

题目大意

在$N \times M$的城市里，道路都是水平或竖直的且宽度为 1，且交叉口不超过 26 个，从每个格子走到相邻的格子都需要一定的时间。

一个人从某一个格子出发，要连续地经过给定的交叉路口序列，最终到达某一个格子停下。问第K分钟时他在哪里。

$N,M \leq 100$，$K \leq 100000$，时间限制 1s。

简要题解

这道题很没意思，不知道为什么放到国家集训队作业里了，只要直接模拟就行了。

由于交叉路口是“连续地经过”，意思就是在序列中相邻的两个路口在地图上是在同一条水平线或竖直线上的，并且可以直接到达（中间不用经过其他路口）。这样非常容易就能编出来了。

时间复杂度$O(NM+K)$，空间复杂度$O(NM)$。

1.11 CF243C Colorado Potato Beetle

题目大意

在一个无限大的方格矩阵中画出首尾相连的N条线，每条线方向平行于坐标轴。将折线经过的小方格涂色。求这些线段围住的总面积（包括围在内部的未涂色的部分）。

$N \leq 1000$，时间限制 1s。

简要题解

只需要先离散化，离散化以后最多就只有$4N^2$个矩形。然后把这N条线所在的矩形全部标记为不可访问，再从这些矩形以外的地方开始 floodfill，最后看有多少面积没有覆盖到

即可。

时间复杂度$O(N^2)$，空间复杂度$O(N)$。

1.12 CF249C Piglet's Birthday

题目大意

Winnie 家有N个架子，每个架子上最初有a_i个装满蜜的蜜罐。他进行了q次操作，每一次从第u_i个架子上拿k_i个蜜罐吃完，然后把它们都放到第v_i个架子上。

求每一次操作后期望有多少个架子上的蜜罐都是空的（没有罐子也算）。

$N,q \le 10^5$，$a_i \le 100$，$k_i \le 5$，时间限制 1s。

简要题解

我们维护f[i][j]，表示第i个架子上有j个没吃完的蜜罐的概率。

每一次操作后真正对期望有贡献的只有第u_i个架子，于是就用组合数计算概率，维护一下：$f'[u_i][j-k'] += f[u_i][j]C_j^{k'}C_{a_i-j}^{k_i-k'}$（j表示当前第$u_i$架子上有多少个蜜罐，k'表示拿了多少个装满蜜的蜜罐）。然后更新第u_i和v_i个罐子的a值。

计算组合数时由于上标不超过 5，我们可以预处理出来。

时间复杂度$O(Sum \times k_{max})$，空间复杂度$O(Sum \times k_{max})$（$Sum = \sum_{i=1}^{N} a_i$）。

1.13 CF249D Donkey and Stars

题目大意

天上有N颗星星，每颗星星的坐标给定。

Donkey 从$(0,0)$开始望，他望向(a,b)和(u,v)方向，然后选择一颗在这两个方向之间（不包括边界）的星星，然后走到这颗星星；再望向(a,b)和(u,v)方向，然后选择一颗星星走到它；再望向(a,b)和(u,v)方向……直到无法进行为止。

问他最多能走到多少颗星星。

$N \le 10^5$，时间限制 1s。

简要题解

如果你把坐标变换成水平和竖直的，那么就变成了最长上升子序列的问题。然后用线段树搞一搞就可以出来了。

时间复杂度$O(NlogN)$，空间复杂度$O(N)$。

1.14 CF249E Endless Matrix

题目大意

有一个这样的矩阵：

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 …

… … … … … … …

求左上角为$(x1,y1)$，右下角为$(x2,y2)$的子矩阵中所有数的和。

如果不足 10 位则直接输出，否则只输出后 10 位（包括前导 0）。

$x_1,y_1,x_2,y_2 \le 10^9$，数据组数$T \le 10^4$，时间限制 1s。

简要题解

显然可以转化成求左上角为$(1,1)$，右下角为(x,y)的子矩阵中所有数的和。

令$p = \min\{x,y\}$，那么左上角$p \times p$的正方形中的数就是$1,2,\dots,p^2$。然后分情况讨论：

如果$x > y$，考虑第一列左下角的数是$(y+1)^2,\dots,x^2$，可以用平方和公式求出和，后面就是等差数列了；

如果$x < y$，考虑第一行右上角的数是$x^2+1,\dots,(y-1)^2+1$，可以用平方和公式求出和，后面就是等差数列了。

关于输出的讨厌问题，我的做法是直接用高精度求出真实值（反正不会超过10^{36}），然后再讨论一下。写起来有点麻烦。

时间复杂度$O(T)$，空间复杂度$O(1)$。

1.15 CF251D Two Sets

题目大意

将N个非法整数A_i分到两个集合中，第一个集合中的数异或和为X_1，第二个集合中的数异或和为X_2。要求$X_1 + X_2$最大且在满足前一条件的前提下让X_1最小。求一个方案。

$N \le 100000$，$A_i \le 10^{18}$，时间限制 1s。

简要题解

令所有数的异或和为X。如果X的某个二进制位为 1，那么X_1和X_2的对应二进制位一定是一个 0 一个 1；如果X的某个二进制位为 0，可以证明：如果可能，X_1和X_2的对应二进制位一定都是 1（否则无法使$X_1 + X_2$最大）。

然后从高到低考虑那些X为 0 的二进制位，每一位的条件都是一个异或方程。将它与前面的方程联立，如果没有出现矛盾则加入。这样就可以$X_1 + X_2$最大。

在$X_1 + X_2$最大的条件下如何让X_1尽量小呢？

这时就要再从高到低考虑那些X为 1 的二进制位，尽量使X_1的那一位为 0。每一位的条件也是一个异或方程。将它与前面的方程（包括使$X_1 + X_2$最大的方程）联立，如果没有出现矛盾则加入。这样就可以了。

时间复杂度$O(64^2N)$，空间复杂度$O(64N)$。

1.16 CF253E Printer

题目大意

有一台打印机和N个任务。每个任务都有到达的时间、要打印的数量和优先级。每个任务自到达以后就放在等待队列里，直至任务完成为止。

每一秒钟的开始，如果等待队列里有任务，打印机从等待队列里选出优先级最高的任务，并打印一页。所有任务优先级互不相同。

当前有一个任务优先级不明，但要求它恰好在T秒后完成。求它的一个可能优先级，并且模拟出每个任务完成的时间。

$N \leq 50000$，时间限制 4s。

简要题解

这道题有浓浓的二分答案性质。因为你将一个任务的优先级提高，它不会完成更晚。于是我们二分该任务的优先级Ans并且进行模拟。

模拟一遍需要的时间是O(NlogN)（因为要使用堆），这样总时间复杂度就是$O(N\log^2 N)$。但我们有更好的方法。

首先，在这个任务到达之前，无论二分的Ans是多少都是一样的。我们就先模拟到这个任务到达之前的情况。这个任务到达之后情况发生了变化，但是有统一的规律：所有在时刻T之前到达的任务，只要优先级不小于Ans，就一定会打印完；如果优先级小于Ans，就一张也不会打印出来。然后再看是不是正好在T秒打印完。这样，这个过程就降到了O(N)，总时间复杂度也就降到了O(NlogN)。

最后，要注意保证所有任务优先级互不相同。还有，二分答案时要小心不要溢出（因为我设$L = 1, R = 1234567890$，结果在Ans很大时溢出了，最后改成了$R = 1034567890$）。

时间复杂度O(NlogN)，空间复杂度O(N)。

1.17 CF256D Liars and Serge

题目大意

有N个人，有的人只说真话，有的人只说假话。

你问他们每一个人“有多少人说真话”。说真话的人会告诉你正确的数字，说假话的人会告诉你1~N任意一个不等于正确答案的数字。

然后你判断出了恰好有K个人一定说谎了。问他们的答案序列有多少种。

$N \leq 256$且是 2 的幂，时间限制 1s。

简要题解

如果有x个人说x，那么你就无法判断他们是否说谎。然后就有 DP 方程：

f[i][j][k]表示当前讨论完说1~i的人，一共有j个人说了1~i，其中恰有k个人一定说谎了。转移时枚举有多少个人说i，然后用组合数算一下就行了。

这样时间复杂度是$O(N^4)$的，大数据可能通不过。但是由于题目中对N的限制比较奇怪，你可以直接打表呀。

时间复杂度O(1)或$O(N^4)$，空间复杂度O(N)或$O(N^3)$。

1.18 CF258D Little Elephant and Broken Sorting

题目大意

小象有一个1~N的排列，它对这个排列操作M次，每次交换两个位置的数。

但是出了一些问题，它的每个操作只有一半的概率被执行。问最后完成操作的排列逆序对的期望个数。

$N,M \leq 1000$，时间限制 1s。

简要题解

设f[i][j]表示第i个数大于第j个数的概率，那么最后的答案就是所有的f[i][j]之和。

f[i][j]的初始值由初始排列决定，然后每次操作后只有$O(N)$个值会改变：

假设交换第x个数和第y个数，那么$f'[x][y] = f'[y][x] = 0.5$，

$f'[i][x] = f'[i][y] = \frac{f[i][x]+f[i][y]}{2}$，$f'[x][i] = f'[y][i] = \frac{f[x][i]+f[y][i]}{2}$。

时间复杂度$O(N^2 + NM)$，空间复杂度$O(N^2)$。

1.19 CF260E Dividing Kingdom

题目大意

平面上有N个点。问能否用两条水平线和两条竖直线将平面分成 9 份，使得每一份分别有$a_1,a_2,\ldots,a_9$个点（顺序不定）。不能有点在某条水平线或竖直线上。

$N \leq 10^5$，时间限制 1s。

简要题解

我们枚举$a_1,a_2,\ldots,a_9$分别对应的是哪一块，就很容易确定两条水平线和竖直线的位置（因为可以算出一行内有多少个点和一列内有多少个点）。

然后就是要具体地求出一块内的点数是不是规定值。这等价于求一个矩形内的点数。你可以将询问排序后再做，这里我偷懒直接写主席树了。

时间复杂度$O(9! \times 9\log N)$，空间复杂度$O(N\log N)$。

1.20 CF261D Maxim and Increasing Subsequence

题目大意

将一个长度为N的序列复制t遍，求这个长度为tN的序列的最长上升子序列的长度。

Maxim 有k个长度为N的正整数序列，它们的元素最大值均不超过maxb。对每一个序列求出将其复制t遍后的最长上升子序列的长度。

$N,maxb \leq 10^5$，$t \leq 10^9$，$N \times maxb \leq 2 \times 10^7$，$k \leq 10$，时间限制 3s。

简要题解

一个事实是$t \leq \min\{N,maxb\}$时才有意义，往后再复制就没有用了。

令f[i]为当前以不超过i的元素结尾的最长上升子序列的长度。然后对长度为tN的序列A_i中每一个元素进行一次更新：$f[A_i] = \max\{f[A_i], f[A_i - 1] + 1\}$。

每一次更新之后用$f[A_i]$更新$f[A_i + 1], f[A_i + 2], \ldots, f[maxb]$，一旦发现不能更新就退出。

一旦发现不能更新就退出！（因为很重要所以说两遍）

这个看上去很暴力的方法实际上效果相当好，因为对每一个i都有$f[i] \leq \min\{N,maxb\}$，总的更新次数不会超过$N \times maxb$。这样才能保证时间复杂度。

时间复杂度$O(kN \times maxb)$，空间复杂度$O(N + maxb)$。

1.21 CF261E Maxim and Calculator

题目大意

Maxim 的计算器显示两个数a和b。初始时$a=1$，$b=0$。有两种操作：

1. $b=b+1$；
2. $a=ab$。

问在[l,r]之间有多少个整数x满足，可以由初始状态操作不超过p次使得$a=x$。

$r \le 10^9$，$p \le 100$，时间限制 4s。

简要题解

实际上，在$1\sim10^9$中，素因子都不超过 100 的数不超过 4000000 个，于是我们先把这些数枚举出来并且从小到大排序。

接下来用一个 DP，f[i][j]表示用不超过j的数乘出i的最少次数，然后$f[i][j]+j$就是所需要的操作次数（$1 \le j \le p$）。由于已经排过序，所以 DP 的时间复杂度为$O(4000000p)$。

用滚动数组来节约空间。

时间复杂度$O(4000000\log 4000000+4000000p)$，空间复杂度$O(4000000)$。

1.22 CF263E Rhombus

题目大意

有一个$N\times M$的数表$A_{i,j}$，求出$f(x,y)=\sum_{i=1}^{N}\sum_{j=1}^{M}\max(0,k-|x-i|-|y-j|)$的最大值（$k \le x \le N-k+1$，$k \le y \le M-k+1$）。

$N,M \le 1000$，时间限制 3s。

简要题解

很容易发现求和的部分形状像是一个菱形（这就是这道题目的题目的来源吧）。

我们将这个菱形向右移动一格时，求和时就会有两个变化量：加上一个“▶”的和，减去一个“◀”的和。但是菱形还要上下移动，所以就向上下左右箭头的和都要求出来。那么如何求一个右箭头的和呢（其余情况对称）？

将右箭头上下移动时，求和的变化量是：加上一条长度为k的“/”的和，减去一条长度为k 的“\”的和；将右箭头左右移动时，在考虑斜线的同时还有考虑长度为$2k-1$的竖直的线的和。

总结一下，先要求出$N\times M$的数表中所有长度为k的“/”“\”的和以及长度为$2k-1$的横线和竖线的和，然后再求出上下左右箭头形的和，最后再求出整个菱形的和就行了。

以上是我写的方法，相当麻烦，极大地考验人的耐心（实际上很显然是我自己找罪受）。下面还有一些其他方法：

将菱形横竖剖分成四个三角形。我们向右移动三角形时会有两个变化量：斜着的k个数的和，以及竖着的k个数的加权和。求这个比我的方法简单多了。

或者将坐标轴旋转 45°，这样就将菱形变成了正方形。然后考虑矩阵前缀和的前缀和应该也能做。

时间复杂度$O(NM)$，空间复杂度$O(NM)$。

1.23 CF264E Roadside Trees

题目大意

有1~N共N个位置可以种树，每个月初树长高 1 米。有M个月，每个月初你要进行一次操作。操作有两种：

1. 在某个位置种下一棵初始高度为h的树；
2. 砍去从左到右第x棵树，此后这个位置不会再种树。

每次操作之后要求出当前从左到右树高的最长上升子序列的长度。保证任意时刻没有相同高度的树。

$N \leq 10^5$，$M \leq 2 \times 10^5$，$x,h \leq 10$，时间限制 3s。

简要题解

将每棵树种下去的初始高度减去它种下去的月份，这样就不要再考虑长高 1 米的问题。

这道题最关键的条件就是$x,h \leq 10$！假设每棵树的坐标为(Pos,H)，我们种下去的树一定是 y 坐标前 10 小的树，我们砍掉的树一定是 x 坐标前 10 小的树！设f[i]表示以i为最小元素的最长上升子序列的长度，那么一次操作改变的f[i]的值不超过 10 个！

对于改变 y 坐标前 10 小的树，为了找出这 10 棵树，要关于 y 坐标建立一个小根堆；为了计算出f[i]的值，要关于 x 坐标建立一棵线段树。然后种树时，先将 y 坐标前 10 小的树从小根堆和线段树中删去，然后加上新种的树，再倒序加入小根堆和线段树，同时求出f[i]。

对于改变 x 坐标前 10 小的树，为了找出这 10 棵树，要关于 x 坐标建立一个小根堆；为了计算出f[i]的值，要关于 y 坐标建立一棵线段树。然后种树时，先将 x 坐标前 10 小的树从小根堆和线段树中删去，然后去除砍掉的树，再倒序加入小根堆和线段树，同时求出f[i]。

要维护的东西有点多，写起来相当麻烦。

时间复杂度O(SumlogN)，空间复杂度O(N)（$Sum = \sum x + \sum h$）。

1.24 CF266D BerDonalds

题目大意

给定一个N个点的无向图，求距离最远顶点最近的点的位置以及它与最远顶点的距离（这个点可以不是顶点）。

$N \leq 200$，时间限制 500ms。

简要题解

首先 Floyd 不用多说，求出两两顶点之间的最短距离。然后顶点的情况就直接做掉。

然后考虑所求的点在某条边上，假设为(u,v)。那么一定有一部分结点是通过u到达的，另一部分的结点是通过v到达的。那么距离显然应该是$\frac{d[x,u]+l+d[y,v]}{2}$（x是第一部分结点中距离u最远的，y是第二部分结点中距离v最远的）。

如何快速计算这玩意儿呢？我们先将所有顶点按照距离u从近到远排序，这样第一部分的结点一定是一个前缀序列，然后对于后缀序列从尾到头扫一遍最大值就可以维护了。

你可能会担心这样计算出的位置其实并不在(u,v)上。实际上并不用担心。因为对于这个点真正在的那条边，计算的结果一定是在这条边上的；对于其它的边，计算的距离不可能比实际值还小。

如果对于每一条边都排一次序是$O(N^3 \log N)$的。你可以将排序的过程预处理，这样就是$O(N^2 \log N)$的了。

时间复杂度$O(N^3)$，空间复杂度$O(N^2)$。

1.25 CF266E More Queries to Array…

题目大意

有一个长度为N的序列A_i，有两种操作共M个：

1. 将$A_l, \dots, A_r$赋值为x；2. 询问$\sum_{i=l}^{r} A_i (i-l+1)^k$的值（$0 \le k \le 5$）。

$N, M \le 100000$，时间限制 5s。

简要题解

这是一个简单的线段树维护，用线段树维护一段区间$[l, r]$的$\sum_{i=l}^{r} A_i i^k$。

修改只需要先预处理出$\sum i^k$的前缀和；询问只需要先求出$\sum_{i=l}^{r} A_i i^k$，再用二项式定理就能在$O(k^2 + k \log N)$的时间内完成。

时间复杂度$O\left(N k_{max} + M {k_{max}}^2 + M k_{max} \log N\right)$，空间复杂度$O(N)$。

1.26 CF267C Berland Traffic

题目大意

在N个点M条边的图中，每条边有容量限制，求 1 到N的最大流，满足对于任意的x和y，从x到y无论选择什么路径，路径上所有边的流量之和都相等。

$N \le 100$，$M \le 5000$，时间限制 1s。

简要题解

看到要求满足的性质，妥妥的最短路既视感，那么就设$d[i]$表示最短路的距离标号，那么从x到y的流量就是$d[y] - d[x]$。

但是这又是网络流，于是要满足流量平衡性质，这样就有$N-2$个方程了。还有两个方程呢？$d[1] = 0, d[N] = 1$。哈哈！接下来就是高斯消元解方程了。

然后根据每条边实际的流量限制将$d[i]$扩大若干倍使之成为最大流，这里的倍数是指对所有边(x, y)，$\frac{g[x,y]}{|d[y]-d[x]|}$的最小值。最后注意诸如$d[y] = d[x]$且$g[x, y] \ne 0$的特殊情况。

时间复杂度$O(N^3)$，空间复杂度$O(N^2 + M)$。

1.27 CF268D Wall Bars

题目大意

Manao 要建造一个奇怪的高度为N的建筑物。在$1 \sim N$的每个高度修建一个横杆，横杆可以向四个方向建造。

有小孩子要来爬，他从地面可以爬到高度为$1, 2, \dots, h$的横杆。如果两根横杆方向相同，并且高度差不超过h，他可以从低的横杆爬到高的上。

Manao 的要求是这个小孩可以爬到高度为$N-h+1, N-h+2, \dots, N$的某一根横杆上，求

方案数。

$N \leq 1000$，$h \leq 30$，时间限制 1s。

简要题解

这道题显然是 DP，DP 方程不难想，但是状态巨麻烦：

设f[i][j][k][l][0/1]表示当前修建到第i层，其它三个方向的最高横杆离i的距离，能否从地面爬到第i根横杆的方案数。

然后转移就是每一次枚举下一根横杆修建的方向。写起来很麻烦呀。

常数优化：令$j \leq k \leq l$。

空间优化：用滚动数组。

注意事项 1：$j(k,l) = h \Leftrightarrow$从地面无法爬到$j(k,l)$的方向；

注意事项 2：最后满足要求的条件比较奇怪，所以从$i = N - h + 1$起，将f[i][j][k][l][1]计算出来以后不能再向后转移，$Ans+= f[i][j][k][l][1] \times 4^{N-i}$，然后将f[i][j][k][l][1]清零。

时间复杂度$O(Nh^3)$，空间复杂度$O(h^3)$。

1.28 CF273D Dima and Figure

题目大意

Dima 要在$N \times M$的白色纸片将若干个格子涂黑，使得涂黑的格子满足以下条件：

1. 至少要有一个黑格，且黑格相互四连通；
2. 从任意一个黑格(x_1,y_1)到(x_2,y_2)的距离必须是$|x_1 - x_2| + |y_1 - y_2|$（从一个黑格一步只能移动到与它四连通的另一个黑格）。

求总方案数。

$N,M \leq 150$，时间限制 1s。

简要题解

条件 2 是一个相当强大的性质！可以从中得到许多有用的结论。

首先就是不会存在空洞（否则与它在同一个方向上相邻的两格距离就不是 2 了），那么每一行的状态实际上就是$[L_i,R_i]$，这是我们 DP 的基础。

光有以上的条件还不够。比如说“工”字形，从左上角到左下角的距离就不满足条件。

仔细想一想可以得出结论：从上到下来看，L_i和R_i都必须是单峰的。L_i是先减后增（一旦开始增大就不能再减小）；R_i是先增后减（一旦开始减小就不能再增大）。

这样是不是就可以了呢？显然可以了，你可以分情况讨论一下。

那么就出现了一个五维的 DP 状态：f[i][j][k][0/1][0/1]表示当前在第i行，第i行的黑格为[j,k]，左边界能否继续减小，右边界能否继续增大。

转移略麻烦，因为有四个不同的状态相互转移，写起来需要一些时间。但是仍然可以做到每行$O(N^2)$。

时间复杂度$O(N^3)$，空间复杂度$O(N^3)$。

1.29 CF273E Dima and Game

题目大意

Dima 和 Anya 玩一个游戏：

游戏开始前 Dima 在纸上写下N对整数(L_i,R_i)（$1 \le L_i < R_i \le P$）。然后 Anya 先手，两人轮流操作。一次操作如下：

1. 选择一对(L_i,R_i)，满足$R_i - L_i > 2$；
2. 将它们替换为$\left(L_i + \left\lfloor \frac{R_i-L_i}{3} \right\rfloor, L_i + 2\left\lfloor \frac{R_i-L_i}{3} \right\rfloor\right)$或者$\left(L_i, R_i - \left\lfloor \frac{R_i-L_i}{3} \right\rfloor\right)$。

不能操作者输。

Dima 希望先手的 Anya 赢，问他有多少种写(L_i,R_i)的方法使得当两人都按照最优策略操作时 Anya 赢（顺序不同也算不同）。

$N \le 1000$，$P \le 10^9$，时间限制 1s。

简要题解

这道题也是一道正常的博弈论问题。

显然$R-L$相同的(L,R)实际上是一样的，故可设$SG[d]$表示当$R-L=d$时的 SG 函数。

$SG[d] = \text{mex}\{SG[\lfloor \frac{d}{3} \rfloor], SG[d - \lfloor \frac{d}{3} \rfloor]\}$，故$0 \le SG[d] \le 2$。

接下来的任务就是求出$SG[1], SG[2], \ldots, SG[P-1]$。

经过小规模数据的打表可以知道这个 SG 函数是分段的，在一段内的所有数 SG 函数值都相同。我写了一个迭代求出 SG 函数值的算法，发现在最大数据时也只要分 102 段。

然后就是求出 SG 值为 0,1,2 的(L_i,R_i)对的个数，最后再用一个 DP 求出结果即可。

时间复杂度$O(N+102)$，空间复杂度$O(N+102)$。

1.30 CF277D Google Code Jam

题目大意

Vasya 做 Google Code Jam 比赛。一场比赛有N道题，每道题分 Easy 和 Hard 部分。

现在知道每道题 Easy 和 Hard 的分数，Vasya 做 Easy 的时间，以及在 Easy 的基础上做 Hard 的时间（对于一道题必须先做 Easy 再做 Hard）。他做 Easy 可以 AC，但是做 Hard 会有p_i的几率挂掉。

Google Code Jam 的罚时就是最后一次 AC 的时间。

一场比赛总时间为T。Vasya 希望知道他的期望得分最大可能是多少，以及在期望得分最大的情况下期望罚时最小是多少。

$N \le 1000$，$T \le 1560$，时间限制 1s。

简要题解

一旦选好了做哪些 Easy 和哪些 Hard，期望得分就确定了，那就考虑期望罚时最小。

设第i题做的时间为T_i，挂掉的概率为p_i。对于两道题i和j：

先做i后做j的期望罚时为$(T_i+T_j)(1-p_j)+T_i(1-p_i)p_j$，

先做j后做i的期望罚时为$(T_i+T_j)(1-p_i)+T_j(1-p_j)p_i$。

若要使先做 i 更优，则有$(T_i+T_j)(1-p_j)+T_i(1-p_i)p_j < (T_i+T_j)(1-p_i)+T_j(1-p_j)p_i$，即$T_ip_i(1-p_j) < T_jp_j(1-p_i)$。

这样做 Hard 的顺序就确定了，应当将所有题目按此排序。

同时也可以知道一定是先做完所有选定的 Easy 再按照上述的顺序做 Hard。

于是就可以 DP 了，令$f[i][j], g[i][j]$表示花j分钟做前i题的最大期望得分与最小期望罚时。

转移时分别考虑不做这道题，只做 Easy，做 Easy 和 Hard（如果做 Hard，那么在当前

状态下，肯定把这个 Hard 放在最后做）。

时间复杂度$O(NT)$，空间复杂度$O(NT)$。

1.31 CF280D k-Maximum Subsequence Sum

题目大意

给定一个长度为N的序列，有M个操作。操作分两种：

1. 更新$A[x]$为y；
2. 求在$[l,r]$中取不超过k个连续子段的最大子段和。

$N,M \leq 10^5$，$k \leq 20$，时间限制 5s。

简要题解

首先，这道题显然是线段树维护。

一个容易想到的做法是，维护每一段区间里的$1\sim k$子段和，以及包括左边界和包括右边界的$1\sim k$子段和。这样确实可以做（虽然已经相当麻烦了），但是单次维护的复杂度是$O(k^2 \log N)$，很遗憾无法通过。于是我们需要另辟蹊径。

范浩强神犇提供了另一种静态求k子段和的方法：我们每一次求出最大子段和，然后将这一段取反，再求一次最大子段和……这样求k次就可以了。

正确性？我们联想一下，如果用费用流做这道题，那么实际的过程就是这样的，结果也是能正确地求出来的。

这样我们就从k子段和中解放出来了，只需要维护单子段和，时间复杂度自然降了一阶。但是还要维护其位置以及取反（最小子段和也要维护了），于是我花了 16 个变量才终于完成了维护。写起来真爽！

常数上的小优化：每一次取到的最大子段和是递减的。如果这次取到了负数，那么以后就不会再取到正数了。此时应当立即停止。

时间复杂度$O(N + MK\log N)$，空间复杂度$O(N)$。

1.32 CF283E Cow Tennis Tournament

题目大意

N头奶牛打比赛，每两头奶牛交手一次。每头奶牛有互不相同的能力值，总是能力值高的打败低的。

Farmer John 要改变比赛结果，他一共改变K次。每一次选择一个$[A_i,B_i]$，改变能力值在$[A_i,B_i]$之间的奶牛相互比赛的结果。

为了检验效果，他想知道K次改变之后有多少个三元组(p,q,r)，满足p打败q，q打败r，r打败p。

$N,K \leq 100000$，时间限制 1s。

简要题解

首先，这种三元组不好统计，应当补集转化。那么问题就变成：对于每头奶牛，有多少奶牛击败了它。

两头奶牛 a 和 b 的比赛结果，取决于它们的能力值以及有多少个区间$[A_i,B_i]$同时覆盖了它们（准确的说，只需要知道奇偶性）。

对于一头奶牛，我们把所有覆盖它的区间找出来。那么查询有多少头能力值小于它的奶牛被覆盖了奇数次，有多少头能力值大于它的奶牛被覆盖了偶数次。这个显然是可以用线段树来完成的。对所有的奶牛都进行这样的询问也很容易，因为如果你按能力值从小到大枚举奶牛，总共也只要$O(N)$次修改。

时间复杂度$O(NlogN + KlogN)$，空间复杂度$O(N + K)$。

1.33 CF285E Positions in Permutations

题目大意

P是$1\sim N$的一个排列。我们称排列P的位置i是“完美的”，当且仅当$|P_i - i| = 1$。

求$1\sim N$的所有排列中“完美”位置恰好有k个的排列的个数。

$N \leq 1000$，时间限制 1s。

简要题解

对于这种求“恰好k个”很难办的问题，我们一般先转化成“至少k个”或“至多k个”，求完之后再用容斥原理找到“恰好k个”的个数。这道题是转化为“至少k个”（因为至少k个意味着你只需要找到k个就可以了）。

对于“不一定完美”的位置，我们先不要去填，只去找已经确定下来的“完美”位置（这样就不会纠结某个数是否在前面很远的地方被填过了）。

DP 状态：$f[i][j][0/1][0/1][0/1]$表示在前i个位置中已经安排了j个“完美”的位置，$i-1, i, i+1$是否被使用过的方案数。

转移时就只考虑是否将当前的位置一定弄成“完美”的。最后求和$g[i]$表示安排了j个“完美”位置的排列数。但是$g[i]$并不是真正的答案，因为还有位置没有填。

真正的$Ans_i = g_i(N-i)! - \sum_{j=i+1}^{N} Ans_j$，从后向前倒推即可。

时间复杂度$O(N^2)$，空间复杂度$O(N^2)$。

1.34 CF293B Distinct Path

题目大意

在$N \times M$的方格纸上填数，每个格子可以填$1\sim K$之间的整数，有一些格子已经填过数。

填好的方格纸必须满足条件：左上角到右下角的每条路径都没有两个一样的整数。路径只能向右或向下走。

$N, M \leq 1000$，$K \leq 10$，时间限制 1s。

简要题解

首先$N, M \leq 1000$是吓人的，如果$N + M - 1 > K$显然无解。

然后我们先用最小表示法搜索出所有可能的路径，然后再对应到原来方格中，计算出有多少种对应的方案。这样搜索量最小。

最后注意答案可能会爆 int。

时间复杂度$O(?)$（我真不知道），空间复杂度$O(NM)$。

1.35 CF293D Ksusha and Square

题目大意

在平面上有一个N个顶点的凸多边形。如果随机选择两个不同的格点(格点可以在凸多边形内部或者边界上面)，并以它们之间的连线为对角线做一个正方形，求正方形的面积的期望。

$N \leq 10^5$，坐标绝对值范围不超过10^6，时间限制 1s。

简要题解

正方形面积就是两个点之间距离的平方的一半，于是可以将 x 坐标和 y 坐标分开看。

对于每一个 x 坐标，我们可以直接求出它对应的 y 坐标的范围。由于给的是一个凸多边形，所以N条边都跑完也只会对整个坐标来回各走一遍。

然后我们得到了诸如$s_1,\ldots,s_k$（s_i表示在凸多边形内部或边界，x 坐标为i的点的个数）的东西。x 坐标上的答案$Ans_x = \sum_{r=1}^{k} s_r \sum_{l=1}^{r-1} s_l (r-l)^2$。

平方是不好维护的，但是我们有大招：$n^2 = C_n^1 + 2C_n^2$。于是就只要维护$\sum_{l=1}^{r-1} s_l C_{r-l}^2$、$\sum_{l=1}^{r-1} s_l C_{r-l}^1$和$\sum_{l=1}^{r-1} s_l C_{r-l}^0$。然后你会发现当$r++$时更新这三个量的时间是$O(1)$的。

对 y 坐标同样这样做。如果你不想复制粘贴，可以将N个顶点的横纵坐标交换，然后直接调用上面的程序。

最后写的时候要注意在对负数取整的时候是向 0 取整的，这点比较坑。

时间复杂度$O(N + 2000000)$，空间复杂度$O(N + 2000000)$。

1.36 CF293E Close Vertices

题目大意

有一棵N个点的带边权树，询问有多少对点满足，两点之间的路径边数不超过L，路径的边权和不超过W。

$N \leq 100000$，时间限制 1s。

简要题解

来了一道树分治题。对于每一棵分治出来的树，我们 DFS 找到这棵树的中心（以它为根的最大子树最小），然后以中心为根遍历整棵树，记录下每个点的(d_1,d_2)（d_1表示该点到根路径的边数，d_2表示该点到根路径的边权和）。

按照题目条件，那么对于一个点，我们需要统计的是一个矩形区域的点数。那就按照某一维排序,将另一维存进树状数组里。对于在同一棵子树中的情况,我们分治到子树中去除。

常数有点紧，我还弄了两个小常数优化才过。

时间复杂度$O(N\log^2 N)$，空间复杂度$O(N)$。

1.37 CF294D Shaass and Painter Robot

题目大意

在$N \times M$的方格纸的某一格里有一个机器人，它初始面对某个对角线的方向。然后它开始移动，将所经之处都涂成黑色（不管之前是否是黑色）。当它到达边界时会遵循光的反射

定律改变方向。当它意识到所有能涂黑的格子都已经涂黑时就停止。

问此时它涂了多少次。或者宣布它根本停不下来。

$N,M \le 10^5$。时间限制 1s。

简要题解

可以证明，如果在四条边界上所有能到达的点都已经到达，那么此时就应该停止（想一想好像还挺显然的）。

于是我们每一次模拟它不改变方向的一段，记录下哪些边界上的格子已经到达，同时累加它涂黑的格子数。如果某个边界上的格子访问超过 2 次，但是仍然存在尚未到达的边界的格子，那就说明陷入了死循环。

还有一个思路：可以证明，如果能够停下来，那么最后一个格子一定在角落里，并且是第二次到达某一个角落。

如果运行了$2(N+M)$次仍然没有到达角落，那么必然陷入了死循环；否则我们统计一下从第一次到达角落到第二次到达角落的过程中总共改变了多少次方向。如果这个值等于边界上能到达的格子数，那就说明确实遍历了所有能到的方格，否则也说明进入了死循环。

这个思路是我随便想的，还没有去实现。如果是对的，那么空间复杂度就降到$O(1)$了。

时间复杂度$O(N+M)$，空间复杂度$O(N+M)$。

1.38 CF295D Greg and Caves

题目大意

Greg 要在$N \times M$的方格纸上画一个“洞”。洞的定义如下：

1. 存在一个区间$[L,R]$，使得在$[L,R]$之间的每一行有两个黑格，其余所有格子为白格；
2. 定义第i行的两个黑点位置为X_i和Y_i，存在一行$t \in [L,R]$，满足以下条件：
   当$L \le i < t$时，$X_{i+1} \le X_i < Y_i \le Y_{i+1}$；当$t \le i < R$时，$X_i \le X_{i+1} < Y_{i+1} \le Y_i$。

求他能画出的不同的洞的总数。

$N,M \le 2000$，时间限制 1s。

简要题解

我们先只考虑“洞”的上半部分。令$f[i][j]$表示在前i行，最低一行宽度为j的方案数。

那么$f[i][j] = 1 + \sum_{k=2}^{j}(j-k+1)f[i-1][k]$。

看上去似乎是$O(NM^2)$的 DP，但是式子右边其实是前缀和的形式，于是可以降次。

“洞”的下半部分和上半部分一模一样。接下来就只要考虑如何组合。

假设上半部分占了1~i行，下半部分占了第$i+1$~N行，

则$Ans+=\sum_{j=2}^{M} f[i][j] \sum_{k=2}^{j-1}(j-k+1)f[n-i][k]$。同理用前缀和优化。

最后还有考虑一种特殊情况：只有上半部分（因为$t=R$是可行的）。

那么为什么不考虑$t=L$的情况呢？仔细想一想就知道它可以包含在前面的情况中。

时间复杂度$O(NM)$，空间复杂度$O(NM)$。

1.39 CF297E Mystic Carvings

题目大意

圆周上有2N个点，有N条直线将N对点连接。

我们要选择三条线，假设分别是 A 与 a 相连，B 与 b 相连，C 与 c 相连。

如果这六个点在圆上顺次分布为 ABCabc 或者 AaBbCc，那么就符合要求。求满足要求的方案数（只要三条线相同就算同一种方案）。

$N \leq 100000$。

简要题解

其实我感觉这道题与 CTSC2008 图腾很相近，只是改到了圆上。

这两种其实是最不好求的。于是补集转化，考虑好求的几种：

1. ABbaCc。我们只要考虑中间的一条线，那么就需要知道它左右两边各有多少条线与之不相交；
2. ABaCca 和 ABCbca，他们的共同特点是：对于第一条线来说，其余两条线有一条与之相交，另一条与之不相交。那么我们只要知道总共有多少条线与它相交。

（注：对以上字母记号不甚理解的请自己画图参考。谢谢!）

现在，问题就变成了对于每一条线，求出它的左右边各有多少条线与它不相交。这个问题我是这样做的：

如果开环为链，对于每一条线我们就要求有多少条线完全包含于其中。

每遇到一个终点则将它对应的起点的位置加入集合S中，计算时求出在这条线上有多少个点在S中。可以用树状数组维护S。

圈上的情况也不难，只要在考虑线$[x,y]$时将$[y,2N+x]$和$[2N+x,2N+y]$也加入即可。

时间复杂度$O(N\log N)$，空间复杂度$O(N)$。

1.40 CF301C Yaroslav and Algorithm

题目大意

Yaroslav 有一种语言，这种语言的接受一个字符串A作为输入，输出也为一个字符串。

这种语言的程序由一些命令组成，第i个命令的形式为s[i] >> w[i]或s[i] <> w[i]，其中s[i]和w[i]是长度不超过 7 的字符串（可以为空），由数字或字符?组成。

运行流程为：

1. 寻找编号最小的命令k，满足s[k]是A的子串。若没有找到则结束；
2. 如果找到k，则在字符串A中，s[k]第一次出现的位置会被w[k]替换。如果这个命令形如s[k] >> w[k]，那么这个程序继续执行（回到 1）；如果形如s[i] <> w[i]，程序终止。

你需要写一个程序，使得对于给定的 N 个正整数，以每一个正整数a_i为输入时输出都是a_i+1（均作为字符串输入输出）。

$N \leq 100$，$1 \leq a_i \leq 10^{25}$，命令条数不超过 50，程序对每个输入的运行步数不超过 200。

简要题解

限于题目条件，我们就不要去想投机取巧的方法，还是老老实实地去实现。

如何将一个正整数加 1？做法就是从最后一位开始扫描，如果是0~8就直接加 1 结束，如果是 9 就改成 0 并继续向左扫描。

在这个问题中只有一个特殊字符?，我们只能用它当指针从最后一位开始扫描。

首先，如何将?加入一个数字末尾？首先>>?（在字符串首加入一个?），然后用这些命令使问号逐步右移：?0 >> 0?，?1 >> 1?，…，?9 >> 9?。

问号到了最右边，就要开始扫描了。为了不破坏之前的命令，这时用?>>??将指针改成??，然后就可以0??<>1，1??<>2，…，8??<>9，9??>>??0。

为了正确处理 999 的情况，需要加一句??<>1。最后要注意以上命令的排列顺序。

时间复杂度$O(1)$，空间复杂度$O(1)$。

1.41 CF301E Yaroslav and Arrangements

题目大意

Yaroslav 称一个数列$a_1,a_2,\ldots,a_r$是“良好的”，当且仅当以下两个条件均成立：

1. $|a_1-a_2|=1$，$|a_2-a_3|=1$，…，$|a_{r-1}-a_r|=1$，$|a_r-a_1|=1$；
2. $\min_{1\le i\le r} a_r = a_1$。

Yaroslav 称一个数列$b_1,b_2,\ldots,b_r$是“优秀的”，当前仅当以下三个条件均成立：

1. $b_1\le b_2\le\cdots\le b_r$；
2. $1\le r\le n$，$1\le b_i\le m$；
3. 通过重排$\{b_i\}$可以得到至少一个至多k个“良好”数列。

给定n,m,k，求不同的“优秀”数列的个数。

两个数列$\{a_i\}$和$\{b_i\}$不同当前仅当两个数列长度不同，或者存在一个位置i满足$x_i\neq y_i$。

$n,m,k\le 100$，时间限制 1s。

简要题解

看到“良好”数列的定义我就想到了山峰的形状。只要令$a_{r+1}=a_1$，那就是一个完美的山峰了。为了简化条件，我们只考虑$a_1=1$的情况（最后记录出现的最大数即可）。

现在问题是怎么对山峰进行 DP。

为了满足“优秀”数列的条件，我们只能先把所有的 1 填进去，再把所有的 2 填进去……依此类推。那么把数字p填进去能填进什么位置呢？

如果$p=1$，初始情况，那就随便填几个 1 都行（只要不少于 2 个）；

如果$p>1$，那么p只能并且必须填进两个相邻的$p-1$之间（我们将之称为“空”），不过在一个“空”里可以填进任意多个p（只要不少于 1 个）；然后两个相邻的p组成新的“空”供$p+1$填。

于是得到了 DP 状态：f[i][j][k][l]表示当前填了1~i的数，一共填了j个数，还有k个“空”需要填数，当前能排出l种不同数列的方案数。转移时枚举要填多少个i，计算出新的“空”的数量，用组合数算出能排出多少个不同的排列。

有常数优化：$i\le\frac{n}{2}$，$k\le\frac{n}{2}$（这里的k指的是 DP 状态而不是题目条件中的k）。

时间复杂度$O(n^3mk)$，空间复杂度$O(n^2mk)$。

1.42 CF303D Rotatable Number

题目大意

像 142857 这样乘上1~6的结果相当于将其旋转的数称为“可旋转数”。

他想要找到最大的b（$1<b<x$），满足在b进制下存在一个长度为n的正“可旋转数”（允许有前导零）。

$n\le 5\times 10^6$，$x\le 10^9$，时间限制 1s。

简要题解

这道题是维基百科结论题，结论是$n+1$是质数且b是模$n+1$的原根。

如何判断原根？直接暴力枚举n的所有约数x，看$b^x \bmod (n+1)$是否是 1 即可。

最后要注意对$n=1$的情况进行特判。

时间复杂度O(枚举 b 的次数$\times\sqrt{n}\log n$)，空间复杂度$O(1)$。

1.43 CF303E Random Ranking

题目大意

N个考生的分数分别在$[L_i,R_i]$之间随机分布(可以取实数)。得分越高的考生排名越靠后。问每个考生取得每个名次的概率。

由于随机性，所以同分的情况可以忽略。

$N \le 80$，时间限制 2s。

简要题解

如果我们知道一个人i的得分$Score_i$，那么他取得每个名次的概率就可以用$O(N^2)$的 DP 求出来：f[i][j]表示前i个人中有j个人排名比他高的概率。

但是$Score_i$的取值有无穷多种！不可能直接算！

对于这种无穷多种取值的事情，一定要取一个区间来考虑：如果一个人的得分在$[l,r]$中，如何计算他排名的概率呢？这时他的排名取决于两个因素：

1. 有多少人的分数在$(-\infty,l)$中（这样一定在他前面）；
2. 有多少人的分数和他一样在$[l,r]$中。

但是第 2 点难倒了我们。因为不同的$[L_i,R_i]$与$[l,r]$的相交部分各不相同，这样我们就无从下手。为了克服这个问题，就要进行离散化！

离散化以后再看第 2 点，如果有K个人的得分在$[l,r]$中，那么每个人获得1~K名的概率就都是$\frac{1}{K}$！然后 DP 呼之欲出：

f[i][j][k]表示前i个人有j个人排名在$[l,r]$，有k个人排名在$(-\infty,l)$的概率（这里要去掉我们要求概率的那个人）。

这个 DP 的复杂度为$O(N^3)$，但是一共有N个人，离散化以后的分数区间有$O(N)$个，时间复杂度达到了$O(N^5)$。不能接受！

有没有优化的余地呢？我是这样想的：

对于同一个$[l,r]$，我们对于每一个人都要计算一次，并且每一次去掉的人都不一样。这样显然很慢。

我的做法是，只计算一次（这一次不去掉任何一个人），然后对于每一个人用 DP 方程逆推出结果（DP 相当于$f[i-1][j][k] \to f[i][j'][k']$，逆推相当于$f[i][j][k] \to f[i-1][j'][k']$）。不需要利用高斯消元，稍微搞一下可以在$O(N^2)$的时间内完成一次逆推。于是我们成功完成了降次！

由于逆推需要除法，要注意尽量减小浮点误差。

时间复杂度$O(N^4)$，空间复杂度$O(N^3)$。

1.44 CF305D Olya and Graph

题目大意

有一条N个点M条边边权均为 1 的有向图，Olya 要添加若干条边权为 1 的边使此图满足以下条件：

1. 从点i出发可以到达点$i+1,i+2,\dots,N$；
2. 任意一条边(u,v)必须满足$u<v$；
3. 对于点i和点j（$i<j$），若$j-i\le k$，则i到j的距离为$j-i$；
4. 对于点i和点j（$i<j$），若$j-i>k$，则i到j的距离为$j-i$或$j-i-k$。

求方案数。

$N\le 10^6$，$M\le 10^5$，时间限制 1s。

简要题解

这个图的限制相当严格，首先每个点i都要有连边$i\to i+1$。

我们考虑此外的边(u,v)，如果$v-u\neq k+1$则显然不满足条件。

是不是足够了呢？对于两条边$(u,u+k+1)$和$(v,v+k+1)$，如果$v\ge u+k+1$，那也不满足条件。至此条件足够了。

可以轻松判断出原题已经不满足条件的情况。除掉之后记录下所有形如$(u,u+k+1)$的u的最大值max和最小值min。

如果没有这样的边，就1~N枚举第一条这样的边的位置，总方案数很容易算出来。

如果有这样的边，就$max-k$~min枚举第一条这样的边的位置，总方案数很容易算出来。

最后稍微注意一下边界就行了。

时间复杂度$O(N+M)$，空间复杂度$O(N)$。

1.45 CF305E Playing with String

题目大意

两个人在玩一个关于字符串集合的游戏。

最初集合中有一个给定的字符串S，每个人轮流操作，一次操作分三步：

1. 选取集合中任意一个字符串T；
2. 选择一个i（$2\le i\le |T|-1$），使得 $T[i-1]=T[i+1]$；
3. 将T从集合中删除，将$T[1\dots i-1]$和$T[i+1\dots |T|]$加入集合.

最后不能操作的人输。如果两人都用最优策略，问先手是否必胜及第一步的策略。

$|S|\le 5000$，时间限制 1s。

简要题解

可以发现，这个问题的实质是一个正常的博弈论问题：

两个人对一个 01 序列轮流操作。每次选择一个 1，将它和它两侧相邻的数字都变成 0。问先手是否有必胜策略。

我们只需要考虑连续的全 1 序列，对于长度为L的全 1 序列求出SG[L]：

$SG[L]=\text{mex}\{SG[L-2],SG[i-2]\text{xor }SG[L-i-1]\}$（$2\le i\le L-1$）

这个 SG 函数可以在$O(N^2)$的时间内求出来，后面就很容易完成了。

时间复杂度$O(|S|^2)$，空间复杂度$O(|S|)$。

1.46 CF306C White, Black and White Again

题目大意

Polycarpus 的N天生活满足“先发生好事，再发生坏事，然后又发生好事”的规律。他知道这N天里会发生w件两两不同的好事和b件两两不同的坏事。每天至少发生一件事，每天要么全部发生好事要么全部发生坏事。求这N天发生事件的方案数（每天发生的事的顺序不一样也算不一样）。

$N,w,b \leq 4000$，时间限制 1s。

简要题解

这是一个简单的组合数学问题。

先不考虑事件两两不同，在x天里发生y件事的方案数为C_{y-1}^{x-1}。

然后可知$Ans = w!\,b!\sum_{i=1}^{N} C_{w-1}^{i-1} C_{b-1}^{N-i-1}$。

时间复杂度$O(N+w+b)$，空间复杂度$O(N+w+b)$。

1.47 CF306D Polygon

题目大意

给定N，求一个N个内角均相同但N边长度互不相等的多边形。

$N \leq 100$，时间限制 1s。

简要题解

显而易见的事实是 N≤4 时无解，否则有解。

这道题做法很多，我的做法是先把其中$N-2$条边的边长定下来。然后再随机这些边长的排列顺序。由于此多边形内角已知，我们就能够排列出$N-1$个点的位置。然后利用角度算出最后一个点的位置。

如果确实组成了一个凸多边形就结束，否则重新随机再计算，直到形成凸多边形为止。

时间复杂度O(kN)（k是随机次数），空间复杂度O(N)。

1.48 CF309B Context Advertising

题目大意

有N个单词，你要从中选出连续的一段单词组成一个r行c列的矩阵。

每一行必须是完整的若干个单词，每两个单词之间用一个空格隔开，长度不能超过c。

问最多可以选择多少个单词，输出方案。

$N \leq 10^6$，$rc \leq 10^6$，总字符数不超过5×10^6，时间限制 2s。

简要题解

首先用单调队列求出从每一个单词出发不超过c列可以走到哪一个单词。

然后倍增，f[i][j]表示从第i个单词出发走2^j行可以走到哪一个单词。

处理询问时枚举每个单词开始，一次是O(logN)的。输出方案也不难。

时间复杂度O(NlogN)，空间复杂度O(NlogN)。

1.49 CF309D Tennis Rackets

题目大意

有一种正三角形的球拍，每条边被N个孔分成等长的$N+1$段，离每个顶点最近的M个孔不能穿线（一共6M个）。现在在三边各选择一个孔穿线，使之组成一个钝角三角形，求方案数（不考虑旋转、翻转同构）。

$N \leq 32000$，时间限制 3s。

简要题解

先把钝角顶点固定在一条边上，然后可以发现，当第二条边上的点单向移动时，第三条边能用的点的范围也是单向移动的。这样枚举前两条边上的点，第三条边上能用的点的个数是可以均摊$O(1)$算出来的。

判断能否组成钝角，这个用余弦定理即可。

本题时间限制太紧，需要一个常数上的优化：由于对称性，只需要枚举一半的情况。

时间复杂度$O(N^2)$，空间复杂度$O(1)$。

1.50 CF311C Fetch the Treasure

题目大意

Rainbow 有h间密室排成一排，编号为1~h。其中有n间有宝藏。

Freda 从第一间密室出发探险，她一次可以向前走k间密室或者回到第一间密室。现在有三种操作共m个：

1. Freda 新加了技能x，这样她一步除了可以按原有的方法走，还可以一次向前走x间密室；
2. 让某一间宝藏密室的价值减少；
3. 询问 Freda 能到达的密室中宝藏价值最大的一个。之后 Freda 会将其全部取走。

$n,m \leq 10^5$，$k \leq 10^4$，操作 1 的个数$s_1 \leq 20$，时间限制 1s。

简要题解

首先来看如何知道她能到达的所有密室。

因为如果她能到达密室X，就一定能到达密室$X+k$，我们维护F[i]表示在模k余i的所有密室中她能到达的编号最小的密室。当新加技能x的时候，我们就在F[i]之间相互更新。

然后就是维护宝藏信息。这个我们可以将她能到达的密室建立一个大根堆。由于新加技能x只会使能到达的密室增多而不会减少，我们就不要考虑删除了。修改价值就在大根堆里修改。如果一个密室尚未加入堆，就要在数组中修改了，以方便以后加入堆。

时间复杂度$O(m\log n + ks_1)$。

1.51 CF311E Biologist

题目大意

SmallR 要将N只狗改变性别，给每一只狗改变性别的费用为v_i，且只能改变一次。

有M个人打赌。每个人指定了k_i只狗，说如果 SmallR 将这k_i只狗全部变成指定的性别（同

一个人指定的性别相同），则他给 SmallR W_i元。

问 SmallR 怎样做才能最赚钱。

$N \leq 10^4$，$M \leq 2000$，$k_i \leq 10$。

简要题解

这道题一看就是网络流了，而且是最小割类型的网络流。

把所有的W_i预先加入Ans，然后就要求改变性别的钱和不能满足要求的W_i之和最小。

我们把雌性和雄性理解成 S 割和 T 割。如果从雌性转为雄性，则连边(S, i, v_i)；否则连边(i, T, v_i)。

然后是打赌的问题。假设要求改变成雌性，如果不能完成就是至少有一只为雄性。我们连边(第 i 个人,他要求的狗,$+\infty$)和$(S$,第 i 个人,$W_i)$。反之亦然。

然后求最小割就好啦。

时间复杂度$O(N^3 + N^2Sum)$，空间复杂度$O(N + Sum)$（$Sum = \sum_{i=1}^{M} k_i$）。

1.52 CF314E Sereja and Squares

题目大意

有 25 种括号，左括号用 a…z（不包括 x）表示，右括号对应地用 A…Z（不包括 X）表示。有一个N个字符的括号序列，但是一部分左括号和全部的右括号都被擦掉了。求出这个序列可以被还原成多少种括号序列。

$N \leq 100000$，时间限制 4s。

简要题解

很容易想出一个 DP 方程：f[i][j]表示前i个字符中有j个右括号（暂时先不考虑括号之间的差别，最后再去乘）。

$f[i][j] = f[i-1][j]$，若 S[i] ≠?

$f[i][j] = f[i-1][j-1]$（如果可以加右括号）$+ f[i-1][j]$，若 S[i] =?

然后用滚动数组加上常数优化就能通过。（为什么f[i][j]要定义成这样？因为这样用滚动数组运算量最小）

时间复杂度$O(N^2)$，空间复杂度$O(N)$。

1.53 CF316D PE Lesson

题目大意

有N个人，每个人手上有一个互不相同的球。每一次可以指定两个人让他们交换手中的球。但是有的人最多只能进行 1 次交换，剩下的人最多进行 2 次交换。问最后他们一共能得到多少种不同的持球方式（有人在两种持球方式中拿的球不同，则这两种持球方式不同）。

$N \leq 10^6$，时间限制 1s。

简要题解

先不考虑能进行 2 次交换的人，设$f_1[i]$表示有i个只能进行 1 次交换的人共有多少种持球方式。根据第i个人是自己玩还是找人交换有 DP 方程：$f_1[i] = f_1[i-1] + (i-1)f_1[i-2]$。

然后在考虑能进行 2 次交换的人。他可以自己玩，也可以插到任意一个置换的任意一个位置。（如果这个置换只有 1 个人，他就进行 1 次交换；否则他就进行两次交换）。

这样如果当前有m个人，他就有$m+1$种方式插进去，总方案数就要乘$m+1$。

时间复杂度$O(N)$，空间复杂度$O(N)$。

1.54 CF316E Summer Homework

题目大意

对一个长度为N的序列A_i进行M次操作，操作有三种：

1. 将某个A_x的值改为v；
2. 求$\sum_{i=l}^{r} A_i F_{i-l}$（这里$F_0 = F_1 = 1$，且$F_{i+2} = F_i + F_{i+1}$）；
3. 将$A_l, A_{l+1}, \ldots, A_r$全部加上$d$。

$N, M \leq 200000$，时间限制 3s。

简要题解

又是一道线段树维护区间的题目，这一回对于$[l,r]$，维护的是$\sum_{i=l}^{r} A_i F_i$。

修改非常好办，预处理出F_i的值和前缀和就行了。

重点是查询。有一个公式是$F_{i-n} = F_{-(n+1)} F_{i-1} + F_{-n} F_i$（注意这里的$F_i$和经典的定义不一样，公式也不一样）。于是我们发现对于$[l,r]$又要维护$\sum_{i=l}^{r} A_i F_{i-1}$。然后就可以了。

时间复杂度$O(N + M \log N)$，空间复杂度$O(N)$。

1.55 CF316G Good Substrings

题目大意

有n条规则，每条规则用(p,l,r)表示，其中p是字符串，l,r是非负整数。一个字符串s符合该规则当且仅当s在p中的出现次数在l~r之间。

再给一个字符串S，问S中有多少个不同的子串符合所有规则（如果一个串在S中出现了多次，统计时只计入一次）。

$n \leq 10$，$|S| \leq 50000$，$|p_i| \leq 50000$，时间限制 1s。

简要题解

又是一道后缀数据结构题，我又用了后缀自动机。

我们将所有的串拼成一个大串T（把S放在第一个），每两个串之间用一个分隔符分开。然后建立T的后缀自动机，并且预处理出每个状态在每个字符串中出现的次数。

那么为什么把S放在第一个呢？因为这样可以确保在S中出现的子串是不含分隔符的。

然后呢，就是逐一枚举每个状态，检查这个状态的字符串是否满足出现次数的条件。如果满足，就将这个状态的长度区间计入Ans。这样就可以保证相同的串只计一次了。

出题人好心地将内存卡到了 128M，而我的程序很不幸用了 177.2M。然后在长时间的搏斗之后我想到了一个绝招：在后缀自动机中对每个状态要记录 c[27]表示它沿某个字符会走到哪个结点。但是 c[27]中每个元素不会超过 1100000，却占了 4 个字节，十分浪费。于是我就用两个 int 记录三个数，然后 c[27]变成了 c[18]，内存就降到了 127.3M（好险!!!）。

时间复杂度$O(Sum)$，空间复杂度$O(36Sum)$（$Sum = |S| + \sum_{i=1}^{n} |p_i|$）。

1.56 CF321D Ciel and Flipboard

题目大意

Ciel 有一个$N \times N$的板子（N为奇数），每个格子有一个数。

设$X = \frac{N+1}{2}$，他可以一次将$X \times X$的正方形中的所有数改变正负号。他可以使用该操作任意多次。要求最大化板子上所有数之和。

$N \leq 33$，时间限制 1s。

简要题解

我们考虑每个数最后是否翻转（这里指改变正负号），那么如果左上角的$X \times X$正方形中每个数是否翻转确定下来，整个$N \times N$的情况就确定了（因为一共只有X^2种翻转方法）。

那么实际上是如何联系的呢？有以下等式（$a_{i,j}$表示(i,j)格是否翻转）：

$$a_{i,j} \text{xor } a_{i,X} \text{xor } a_{i,j+X} = 0，\ a_{i,j} \text{xor } a_{X,j} \text{xor } a_{X+i,j} = 0（1 \leq i,j \leq X）$$

这两个等式为什么成立，稍微想一下就出来了。那么接下来怎么做呢？

我们发现关键在于第X行和第X列，如果$a_{X,1}, a_{X,2}, \ldots, a_{X,X}$以及$a_{1,X}, a_{2,X}, \ldots, a_{X,X}$的值确定了，我们可以在$O(X^2)$的时间内枚举其它格子取 0/1 来得到最大值。但这样时间复杂度是$O(X^2 2^N)$，还不够。

我们可以只枚举$a_{X,1}, a_{X,2}, \ldots, a_{X,X}$的取值，对于$a_{i,X}$，它取 0/1 哪个更优只取决于第i行的数总和哪个更大。于是这样判定$a_{i,X}$的取值就在$O(X)$的时间内能完成，复杂度就降到了$O(X^2 2^X)$。

为了加快运行速度，可以将必要的量预处理出来。

时间复杂度$O(N^2 + X^2 2^X)$，空间复杂度$O(N^2 + X2^X)$。

1.57 CF323B Tournament-graph

题目大意

构造一个N个结点的竞赛图，使得对任意两个结点u和v（$u \neq v$），从u到v的最短距离不超过 2。

$3 \leq N \leq 1000$，时间限制 1s。

简要题解

容易证明若$N = K$时有解，则$N = K + 2$时也有解。（先构造出$N = K$的情况，然后$1 \sim K \to K+1$，$K+1 \to K+2$，$K+2 \to 1 \sim K$）

然后通过手算或随机化可以构造出$N = 3$和$N = 6$的情况，就可以对所有情况构造出解了。同时$N = 4$时无解易证（手画一下，或者穷举就能知道了）。

时间复杂度$O(N^2)$，空间复杂度$O(N)$。

1.58 CF323C Two Permutations

题目大意

有两个1~N的排列p和q。有M个询问，每次询问有多少个元素在p中的位置在$[l_1,r_1]$，在q中的位置在$[l_2,r_2]$。

$N \leq 1000000$，$M \leq 200000$，时间限制 10s。

简要题解

首先用置换将排列p变成$1,2,\ldots,N$，用同样的置换操作排列q。然后询问就变成了在q中位置在$[l_2,r_2]$的元素有多少个大小在$[l_1,r_1]$之间。

对于这个询问，我们对q中每个1~i中的元素建立一棵权值线段树，然后询问就是$O(\log N)$的了。用可持久化线段树可以保证空间是$O(N\log N)$的。

时间复杂度$O(N\log N + M\log N)$，空间复杂度$O(N\log N)$。

1.59 CF325D Reclamation

题目大意

将一个$R \times C$的地图的左边界和右边界粘起来形成一个圆柱。

有N次操作，每一次挖掉一个格子，要求从上边界到下边界仍然有一条四连通的路径。若不满足则不进行操作。

问最后有多少次操作是成功的。

$R,C \leq 3000$，$N \leq 300000$，时间限制 700ms。

简要题解

对偶地想，如果从上边界到下边界没有一条四连通的路径，那么已经挖掉的格子一定有一条从左到右环绕圆柱一周的八连通路径。

我们将圆柱展开并复制一份放在右边。每次要挖去(x,y)时，我们就考虑(x,y)和$(x,y+C)$之间是否是连通的。由于每次只向其中添加点，只要用并查集就能维护了。

要注意一点：由于它是一个圆柱，在处理左右边界时要注意（回到地图的另一侧）。

时间复杂度$O(RC + N\alpha(N))$，空间复杂度$O(RC)$。

1.60 CF325E The Red Button

题目大意

给定一个由N个点（分别是$0,1,\ldots,N-1$）组成的有向图。每个顶点i只向顶点$2i \bmod N$和顶点$(2i+1) \bmod N$连两条边。判断此图是否存在 Hamilton 回路，存在则求出来。

$N \leq 10^5$，时间限制 1s。

简要题解

首先证明N为奇数时无解。证明：0 只能由$\frac{N-1}{2}$走到，$N-1$只能由$\frac{N-1}{2}$走到。矛盾！

然后再考虑N为偶数时怎么构造出解的问题。（令$N=2d$，以下所有数均模N）

首先可以发现$x,x+d$都只能向$2x,2x+1$连边。于是把它们看成一组。那么比如说你连出$x \to 2x, x+d \to 2x+1$（$0 \leq x < d$），就会形成若干个环。

但是我们的要求是只有一个环。怎么办呢？有引理来救场。

引理：如果图中有不止一个环，那么一定存在x满足$x,x+d$不在同一个环中。

证明：如果对于所有x，$x,x+d$均在同一个环中，那么它们一定和$2x,2x+1$在同一个环

中。那么对于所有的x，$x,2x,2x+1$均在同一个环中。那么考虑一棵完全二叉树，这样就会发现N个点全部在一个环中。

如果存在x满足$x,x+d$不在同一个环中，假设$x\to a,x+d\to b$，那么就交换一下，变成$x\to b,x+d\to a$。这样就可以把两个环合并成一个了。

实际操作又怎么做呢？从 0 开始先搞出一个环来，然后看这个环上的每个x是否有$x+d$不在这个环上。如果有就立刻将$x+d$的环插进来，然后继续向后检查（前面查过的结点就不用再查了）。这样每个点只会被查询 1 次，总的时间复杂度就是$O(N)$了。

时间复杂度$O(N)$，空间复杂度$O(N)$。

1.61 CF329D The Evil Temple and the Moving Rocks

题目大意

你在一个$N\times N$（N为偶数，$N=2k$）的房间里。你需要在方格中放置石块（每个格子最多放一块）。一共有四种石块（每种石块无穷多），每种石块激活以后分别会向上、下、左、右移动。

被激活的石块会一直沿它的方向移动，直到撞上其他石块或者房间四周的围墙（如果在它的方向上紧挨着就有其他石块或者围墙，它将不会有任何移动）。之后这块石头将停止运动，并且解除激活状态。如果它撞到了围墙，则结束；否则，它撞到的石块将被激活。这一过程将会持续发生。

如果一个激活的石块在撞上其他石块或者围墙之前至少移动了一格，那么这次撞击就会有“咣”的一声。你的任务就是让发声次数$\geq x$。你在摆好石头之后只能手动激活一块石头。

$N\leq 300$，$x\leq k^3-k^2$，激活石块的总次数达到10^7就会自动停止。

简要题解

这道题实在是好好玩啊！

因为x在$O(N^3)$的级别，平均起来就是每一格都要遍历$O(N)$次。

首先就应该考虑至少能够周而复始的遍历$N\times N$个格子（即找出一条 Hamilton 回路），这个还是很容易的一件事情。

然后有一种神奇的结构（>> … >>>.>.> … .>.>.），如果左边有连续a个>，右边有b个点，那么激活最左边的石块一次，就可以进行b次撞击（包括一次撞击最右边），而且最左边的石块激活a次以后才会失效（此时最左边的石块会发生移动）。

在有限的长度内可以通过计算得到最优的a和b来进行排布。如果你将 Hamilton 回路的每一条长边都进行这样的排布，就可以保证在一次遍历的过程中撞击次数在$O(N^2)$的级别，并且周而复始，一共可以进行$O(N)$次遍历。

以上都是比较粗略的估计，经过精细的计算以后可以发现能够满足x的要求（如果你实在很懒，你可以在 Codeforces 上试验，出题人写的 Judge 棒棒的）。

时间复杂度$O(N^2)$，空间复杂度$O(N^2)$。

1.62 CF331C The Great Julya Calendar

题目大意

对一个非负整数N，每次减去一个在N的数位中出现过的 0~9 的数字。问最少要多少次

就能减到 0。

$N \leq 10^{18}$，时间限制 1s。

简要题解

这道题是一道非常好的数位 DP 题。

第一步是证明每一次一定是减去N的最大的数位。

设f(N)表示N减到 0 的最小次数，现证明$f(N-1) \leq f(N)$。其中$N = 1$时显然。

对于N含有的任意一个数位x，$N-1$都含有一个$\geq x-1$的数位y，则有下式：

$f(N-1) \leq f(N-1-y)+1 \leq f(N-1-(x-1))+1 = f(N-x)+1$，则$f(N-1) \leq f(N)$。

$f(N-1) \leq f(N)$成立，则原命题必然成立。

现在开始解决这道题。

当$N \leq 10^6$（原题的第一档数据范围）时，可以直接去减。

当$N \leq 10^{12}$（原题的第二档数据范围）时，又该怎么办呢？

你可以把N分成两部分，前 6 位为高位，后 6 位为低位。如果你预处理出10^6以内的数减到 0 需要多少步，你就可以一次性减掉10^6（因为你已经预处理出来怎么减了）。然后就只需要减10^6就行了。

以上的想法存在纰漏：因为“高位”的存在，你不一定会像$N \leq 10^6$那样去减。如果高位中有一个数字 9，你每一次就要减 9！而且你最后不应该减到 0，而是应该减到一个负数！（因为那样高位才会变化）

于是预处理变成了这样：f[i][j]表示在前面的高位有数字j时，把i减到负数的最小步数；g[i][j]表示对应的那个负数。然后你根据高位决定j，用f[i][j]和g[i][j]将N减掉10^6，此时高位发生了变化；再根据新的高位来做……

这样，$N \leq 10^{12}$的情况就解决了。

对N更大的情况，我们要设计一个通用的算法。我的方法是这样的：

可以发现，当高位的数字发生变化的瞬间，低位的数字一定是 99999x 的形式，而每一次高位变化的过程，实际上就是低位从 99999x→99999y 的过程。据此，我设计了如下的 DP 方程（以下我用10取代了10^6）：

f[i][j][k]表示在前面的高位有数字k时，从$10^i - j$开始减，一直减到负数的最小步数；g[i][j][k]表示对应的那个负数（$1 \leq j \leq 9, 0 \leq k \leq 9$）。这个 DP 是可以用$i-1$的情况递推到i的情况的。

现在再来说如何将 f 数组和 g 数组应用到N上去。我是这样做的：

先把N的后一位减到$10^1 - j (1 \leq j \leq 9)$，再把N的后两位减到$10^2 - j (1 \leq j \leq 9)$，再把N的后三位减到$10^3 - j (1 \leq j \leq 9)$……直至N被减完为止。

这个方法可以解决N任意大的情况。

需要注意的一点是，在f[i][j][k]和g[i][j][k]中，如果$k = 0$（相当于没有高位），此时没法减到负数，则它应该存储减到 0 的情况。

时间复杂度$O(10^3 \log N)$，空间复杂度$O(10^2 \log N)$。

1.63 CF332D Theft of Blueprints

题目大意

给定一个N个点的带权无向图，满足任意取k个结点的顶点集合S，恰好有一个点与S中的每一个点都有边。令这个点为v(S)，t(S)为v(S)向S中每一个结点连边的边权之和。

求所有t(S)的平均数。

$N \leq 2000$，时间限制 1s。

简要题解

可以证明k只能取1,2和$N-1$。现证明如下：

1. 任意两个顶点恰有$k-1$个公共的相邻顶点。证明：
   考虑两个顶点s和t，设它们的公共相邻顶点集合为$S=\{v_1,v_2,\dots,v_l\}$。
   若$l \geq k$，则与题意矛盾；
   若$l \leq k-2$，我们考虑集合$T=\{s,t,v_1,v_2,\dots,v_l\}$，然后将T补足至大小为k。
   则考虑$u=v(T)$，显然$u \notin T$，并且与s和t相邻。与S和T的定义矛盾！
   至此结论成立。
2. 图中存在一个大小为$k+1$的完全子图。证明：
   令$S=\{v_1,v_2,\dots,v_k,v_{k+1}\}$，其中$v_{k+1}=v(\{v_1,v_2,\dots,v_k\})$。然后枚举$i=1\sim k-1$：
   对于v_i，如果v_i与$v_{i+1},\dots,v_{k+1}$均相邻，则不管它；
   否则令$T=\{v_1,\dots,v_{i-1},v_{i+1},\dots,v_{k+1}\}$，用$v(T)$替代S中的$v_i$，然后继续。
   枚举完成以后得到的S，显然满足S中的顶点两两相邻。
3. 如果$k \geq 3$，那么$k=N-1$必成立。证明：
   由 2 可知存在一个完全子图，设其顶点集为$S=\{v_1,v_2,\dots,v_{k+1}\}$。
   由 1 可知S中的任意两个顶点v_i和v_j恰有$k-1$个公共的相邻顶点，就是S中剩余的那$k-1$个顶点。
   若$k<N-1$，则存在$u \notin S$。那么u不可能与S中的两个顶点均相邻。
   考虑$T=\{u,v_1,v_2\}$，因为$k \geq 3$，所以必有一个顶点v与u,v_1,v_2均相邻，那么$v \in S$。
   考虑$T=\{u,v,v_1\}$，又会有一个顶点w与u,v,v_1均相邻，那么$w \in S$。
   然后发现u与S中的两个顶点v和w均相邻。矛盾!!!
4. 综上所述，k的取值只能是1,2和$N-1$。

然后就好办了，只需要用求出每条边在几个$t(S)$中出现即可。

时间复杂度$O(N^2)$，空间复杂度$O(N^2)$。

1.64 CF332E Binary Key

题目大意

设p为字符串，q为长度为L的 01 串。下面给出从p中提取出信息串s的方法：

for $i=0$ to $|p|-1$ do if（$q[i \bmod L]=1$）$s=s+p[i]$（下标从 0 开始）

给定原串p，你需要构造出字典序最小的长度为k的钥匙串，使得得到的信息串等于给定的串s。

$|p| \leq 10^6$，$k \leq 2000$，$|s| \leq 200$，时间限制 1s。

简要题解

对于钥匙串中的每一个 1，它是否合法取决于三个条件：它在钥匙串中的位置、它前面有多少个 1 以及整个钥匙串中有多少个 1。

第一步就是要知道整个钥匙串中有多少个 1。易知钥匙串中 1 的个数$s_1 \leq \min\{k,|s|\}$。于是就进行枚举。确定了钥匙串中 1 的个数，就可以 DP 了。

设f[i][j]表示钥匙串的前i位中有j个 1 是否可行，若可行则记录上一个 1 的位置，然后转移时枚举第i位是 0 还是 1。如果这一位取 1，要判断是否合法。怎么判断是否合法并不困难，直接暴力看原串和信息串中对应位置的字符是否全部相等即可。

接下来是字典序的问题。为了保证字典序的条件，应该从后到前进行 DP，更新f[i][j]时应当使得上一个 1 的位置尽量靠后。

时间复杂度O(k|s|^{2}log|s| + |p|)，空间复杂度O(|p| + k|s|)。

1.65 CF333C Lucky Tickets

题目大意

对于一个数字串，如果加上“+”“×”“(”“)”组成算式算出来的结果是N，那么就称这个数字串是N的 Lucky Ticket。

对给定的k，从00000000~99999999的所有八位数字串中取出m个不同的k的 Lucky Tickets。

$0 \le k \le 10^4$，$m \le 3 \times 10^5$。

简要题解

这道题还是相当好玩的，而且解题思路很新颖。

解题的关键是用前四位运算出结果，然后用后四位去凑答案。从0000~9999枚举前四位，再去枚举计算的方法。对于每一个算式的结果看看能否用后四位弄到k。

你需要尝试多种计算方法，不过不必找全所有的方法。平均每个数可以弄出六十多个不同的数就已经够了。

吐槽一句：这题真难为写 SPJ 的人了，SPJ 跑得比程序慢多了。

时间复杂度O(10^4 × 一个四位串能组成的算式数)（我不知道这个数是多少），空间复杂度O(10^4)。

1.66 CF335D Rectangle and Square

题目大意

在3000 × 3000的平面上有N个边均为水平或竖直，且相互之间不重叠的矩形。

判断是否有若干个矩形正好拼成了一个正方形。

$N \le 10^5$，时间限制 1s。

简要题解

第一步：对每一个矩形确定，从它的左下角出发，向上、向右分别能延伸多长；从它的右上角出发，向下、向左分别能延伸多长。这个是为了确定一个矩形作为候选正方形的左下角和右上角时，正方形边长的范围。这个可以通过四遍排序解决。

第二步：确定候选正方形的左下角和右上角的坐标。既然是正方形，它的左下角和右上角坐标满足x − y的值相等。于是我们将长方形的左下角和右上角顶点按照x − y排序(在x − y相等时按照x从小到大排序)。只有在同一条对角线上的才有可能。

接下来是判断同一条对角线上的点是不是正好能画出一个正方形。

我们从小到大遍历这条对角线上的点：如果是左下角就加入堆中，如果是右上角就判断能否出现正方形。将所有超出边长范围的点去掉，然后就只需要去找距离当前点最近的点。（因为如果小正方形不成，包含它的大正方形也一定不成。这一点很重要哦!）

不过有一个很囧的情况：如果一个点既是某个矩形的左下角，又是某个矩形的右上角，请注意避免出现选出0 × 0正方形的情况。

第三步：判断候选正方形是否被正好覆盖满了，也就是求这个正方形内被覆盖的面积。由于这个正方形已经被画了出来，那么所有矩形要么完全在其内部，要么就完全在其外部。

这里你可以用二维线段树来搞，或者将询问排序后用一维线段树来搞。

我的方法更加暴力：对于一个矩形$(x1,y1)-(x2,y2)$，你在$(x2-1,y2-1)-(x2,y2)$这一格里记录下该矩形的面积。预处理这个3000×3000大表的二维前缀和，求覆盖面积时就直接查询正方形区域内所有数的和。

时间复杂度$O(NlogN+3000^2)$，空间复杂度$O(N+3000^2)$。

1.67 CF335E Counting Skyscapers

题目大意

有一排若干座建筑物，每座建筑物有i层的概率为2^{-i}（第$0\sim i-1$层）。两个建筑物在第k层有滑索连接当且仅当$k\le h$，两个建筑物都有第k层，并且两者之间没有其它建筑物有第k层。现在有两个人要来数一共有多少座建筑。

Alice 就直接一座一座地走过去，一边走一边数。

Bob 喜欢走滑索，他从建筑 1 开始，计数器初始值为 1。它每一次都尽量走当前建筑物有的最高的滑索。每走一个第k层的滑索就将计数器的值加上2^k。最后计数器的值就是结果。

现在给定 Alice 和 Bob 中某一个人的结果N，求另一个人读数的期望值。

$N\le30000$，$h\le30$，时间限制 1s。

简要题解

已知 Bob 的情况比较简单。因为一座建筑物有第k层的概率为2^{-k}，每当 Bob 走上一根第k层的滑索，他期望走过的建筑数就是2^k。于是原样输出N就好了。

已知 Alice 的情况，时间限制允许用一个$O(Nh^2)$的 DP 弄出来。

另外一种比较神的思路就是将这N座建筑物从第 0 层到第h层，一层一层垒起来。比如说现在要垒第k层，我们假设第i座建筑物和第j座建筑物之间有一根第k层的滑索：

前提是它们都要有第$k-1$层，概率均为$2^{-(k-1)}$；在此基础上它们各有$\frac{1}{2}$的概率要搭建第k层；然后它们中间的所有建筑不能有第k层，再乘上$\left(1-2^{-k}\right)^{j-i-1}$，就是这根滑索存在的概率。

然后再考虑这根滑索给计数器带来的变化：首先肯定是加了2^k，但是它也使得一些之前存在的滑索消失了。由于i和j本来的高度就是$k-1$，就只要考虑$i\sim j$之间的第$k-1$层的滑索。这些滑索的期望数量自然就是中间$j-i-1$幢建筑中存在第$k-1$层的建筑的期望个数加 1，也就是$1+(j-i-1)\times2^{-(k-1)}$。再乘上$2^{k-1}$得到需要减去的期望。

总而言之就得到了下面的一个式子：

$$Ans=N+\sum_{k=1}^{h}\sum_{L=1}^{N-1}(N-L)\times2^{-2k}\times\left(1-2^{-k}\right)^{L-1}\times\left(2^k-2^{k-1}\left(1+(L-1)\times2^{-(k-1)}\right)\right)$$

在上式中，N就是只有第 0 层时的期望，L是滑索的长度（相当于上文中的$j-i$）。

稍微搞一搞就能够在$O(Nh)$的时间内求出来了。

时间复杂度$O(Nh)$，空间复杂度$O(1)$。

1.68 CF338D GCD Table

题目大意

有一个N行M列的表格G，满足$G[i][j]=\gcd(i,j)$。

给定一个正整数序列$A_1,A_2,\dots,A_k$，问这个序列是否在G的某一行连续地出现。

$N,M\le 10^{12}$，$k\le 10000$，时间限制 1s。

简要题解

首先考虑$I=\operatorname{lcm}(A_1,A_2,\dots,A_k)$，这个序列如果真的出现，出现的行数显然一定是I的倍数。我们要来证明就会出现第I行：

如果这个序列出现在第qI行，对于第J列满足$\gcd(qI,J)=A_j\neq\gcd(I,J)$，那么就可以得出I不是$A_j$的倍数，与前面I的定义矛盾！

好了，现在已经得到了行数。如果$I>N$直接无解，否则继续求第一个数出现的列数J。

显然有$J\equiv 0(\bmod A_1)$，$J+1\equiv 0(\bmod A_2)$，…，$J+k-1\equiv 0(\bmod A_k)$。

这实际上就是解模方程组了，我们用中国剩余定理即可。可以证明在$0\sim I-1$内最多只有一解。若有解再去检验M列是否能放下。（$J=0$时要知道取$J=I$）

最后不要忘记检查一遍是否真的就是 gcd（因为前面只判断了是不是倍数）。

时间复杂度$O(k\log N+k\log M)$，空间复杂度$O(k)$。

1.69 CF339E Three Swaps

题目大意

将1~N按顺序排成一行，再经过不超过 3 次区间翻转得到给定的排列。

要求你用不超过 3 次区间翻转将其复原。

$N\le 1000$，时间限制 1s。

简要题解

这道题直接搜索呀，然后注意到一次区间翻转最多会新产生两组相邻的差不为 1 的数，以此为估价函数。在复原的过程一次至少应当让一对差为 1 的数回到一起。这样就可以了。

时间复杂度$O(kN^2)$（k的值我不清楚），空间复杂度$O(N)$。

1.70 CF341E Candies Game

题目大意

有N个箱子，每个箱子里有A_i颗糖果。现在要将所有糖果放到恰好两个箱子里。

一次操作要选择两个箱子i和j（要求$A_i\le A_j$），然后从箱子j中拿出A_i颗糖果放进箱子i里。

请你给出操作步骤使得最后满足条件。

$N\le 1000$，$\sum_{i=1}^{N}A_i\le 10^5$，操作步骤总数$Tot\le 10^6$，时间限制 1s。

简要题解

这道题是苏淳的数学竞赛课的例题。现证明，只要初始时有不少于两个箱子有糖果，就一定能满足条件（一个箱子显然不行）。

我们只需证明三个箱子的糖果（$0<a\le b\le c$）可以并到两个箱子里。用数学归纳法证：

当$a = 1$时，我们操作时只向第一个箱子里放糖果，那么第一个箱子里的糖果数就会如1,2,4,8,...变化，可以按照b的二进制将b最后变成 0（当b的二进制位为 1 时从第二个箱子拿，二进制位为 0 时从第三个箱子拿）。

当$a > 1$时，我们类似上述方法进行操作后可以让第二个箱子的糖果数变成$b \bmod a$，由归纳假设可以满足。

然后具体操作就是每一次选择当前糖果最少的三个箱子，按照上述方法去做。

粗略估计一下，一次合并类似于辗转相除法，辗转相除法的每一步又至多有$\log 100000$次操作，故$Tot \leq N\log^2 100000$（这个上界是我乱估的），可以满足要求。在 Codeforces 上实际测试时，这个算法在大部分大数据上操作数仅为标程的 20—35%。足够好了。

时间复杂度$O(N\log^2 100000)$，空间复杂度$O(N)$。

1.71 CF342D Xenia and Dominoes

题目大意

用多米诺骨牌覆盖$3 \times N$的棋盘，要满足每一个非障碍格都要被覆盖一次，障碍格不能被覆盖。还有一个特殊的格子，它不能被覆盖，但是要有一个骨牌能够一步滑到这个格子。求方案数。

$N \leq 10000$，时间限制 1s。

简要题解

这显然是状压 DP 了。用f[i][x1][x2][x3][0/1]表示当前要覆盖第i行，三个格子的状态为x1,x2,x3，已经放置的骨牌能否覆盖特殊格子的方案数。然后每次转移时考虑障碍和特殊格子的问题即可。

后四维状态都是 0/1 的，可以直接用位运算表示。

时间复杂度$O(16^2N)$，空间复杂度$O(16N)$。

1.72 CF346E Doodle Jump

题目大意

有n个平台，第x个平台的高度是$ax \bmod p$（a,p互质）。

有一个 Doodle 要从高度 0 开始跳，它一次最高可以向上跳h的高度。问它能否到达最高的平台。

$n, a, p, h \leq 10^9$，数据组数$T \leq 10^4$，时间限制 1s。

简要题解

这道题的初步想法很显然，就是找出高度相邻的两个平台的最大距离，判断是否超过h。

先假设$a = 5, p = 23$来看一下：

0 5 10 15 20

2 7 12 17 22

4 9 14 19

1 6 11 16 21

3 8 13 18

首先可以看出在[15,20]中出现的数字在[0,5][5,10][10,15]中一定也出现了，那么它们的

结果不会比[15,20]更大。所以只要考虑[15,20]与[20,23]这两个区间。

1. 实际上在任何情况下我们都没有必要考虑[20,23]这个区间。证明：
   什么情况下才最有可能考虑[20,23]呢？那就是 20 和 22 均出现，但是 21 没有出现的前一个。因为此时对应的[15,17]中有 16。
   但是在这种情况下出现了 19，它没有对应的数（$19+5=24>23$），于是[17,19]没有对应的区间，应当考虑的是它。
   一般化，考虑最后一列的数出现的规律：从 0 开始，公差为$a-p \bmod a$的等差数列（模a）。同时，如果一个数$x \geq p \bmod a$，它就不会出现。
   假设最后一列本应该出现z，但是由于n不够大，只出现了$z-a$。设此时包含z的最小区间为$[x,y]$（这是唯一可能需要考虑的情况），那么之前必然有一行没有最后一列（否则最后一列的数会不断增大，z就不应该出现在x,y之后）。显然有$|x-y| \leq a-p \bmod a$。
   同时再考虑最大的没有出现的数w，它与所有出现的数的距离至少为$a-p \bmod a$。那么与w对应的倒数第二列的数的结果自然会比最后一列的结果更大。

这样我们就只要考虑[15,20]，它几乎可以和[0,5]对应起来：

15 20 17 19 16 18

0 5 2 4 1 3

为什么说“几乎”呢？如果$n=21$（数字只出到$h_{21}=13$），那么 18 是不存在的，在这种情况下要将 3 去掉。于是我们又得到了一个新的表：

0 2 4

1 3

看上去很不错。我们成功转化成了$a'=2,p'=5$的情况。但是与刚才不同，现在高度为 5 的平台出现了（就是 20），还要考虑最高的平台与 5 的距离。

如果只有第一行的数（也就是说$a'n'<p'$），那么自然要考虑 5。否则呢？

2. 如果数字不止一行，就不需要考虑 5。证明：
   假设第一行的最后一个数是x，第二行的第一个数$y=x+a'-p'$，那么$[x,p']$和$[y,a']$长度相等。并且如果在$[y,a']$中出现了一个数t，它前面的那个数$s=t-a'+p'$一定出现在$[x,p']$中的相同位置。于是这两个区间是等价的。

这样我们每一次转化时将(a,p)转化为$(a-p \bmod a,a)$。现在我们分析一下时间复杂度。大囧！如果$a=p-1$，那么转化以后就变成了$(a-1,a)$，实际上是$O(p)$的。

好在补救也不难。如果出现这种情况，我们就令$a'=p \bmod a$。这样等价于我们把区间翻转一下，答案自然不会变化。这样$a'=\min\{a-p \bmod a,p \bmod a\} \leq \frac{a}{2}$，时间复杂度降为$O(\log p)$！

好了，时间复杂度已经得到了保证。然后就结束了？才不是呢！

以上情况有一个不容易发现的疏漏。如果最开始就只有一行（也就是说$an<p$），即使转化以后也不会出现高度为 5 的平台（因为 20 原本就不存在）。这时就直接特判。

还有，如果一开始$a>p$……只要想到了就知道怎么办了吧。

结束了！所有情况都考虑完了。

时间复杂度$O(T\log p)$，空间复杂度$O(1)$。

1.73 CF351D Jeff and Removing Periods

题目大意

Jeff 能对一个长度为N的序列A_i做以下操作：

第一步：选择一个子序列$A_v, A_{v+t}, A_{v+2t}, \ldots, A_{v+kt}$（$1 \le v, t \le N$，$k \ge 0$，$v + kt \le N$），满足选择的数全部相同；

第二步：把上面选择的序列删除，剩下来的重标号为$A_1, A_2, \ldots, A_{N-k-1}$；

第三步：将剩余的序列重新排列。

以上三步算一次操作。定义一个数列的“美好度”为将这个数列中的所有数字全部删除所需要的最小的操作次数。

现在Jeff有一个长度为M的序列。有Q个询问，每次询问[l,r]之间的子序列的“美好度”。

$M, Q \le 100000$，时间限制 1s。

简要题解

分析一下“美好度”的定义：

由于我们重新排列的时候一定会将相同的数字放到一起去，所以操作次数为这个数列中不同数字的个数。实际上还有一点需要注意：如果第一步不能将某一个数字全部删除，就需要多操作一次。

然后就只剩下两个部分：询问[l,r]之间的子序列中有多少个不同的数字，以及是否有一个数字在[l,r]中呈等差数列出现。

离线算法比较简单：我们将所有询问按照r从小到大排序，按顺序处理询问。

第一个部分很简单，对每一个A_i存储它前面离它最近的相同的数的位置，然后只需要考虑[l,r]中有多少个数$< l$；

第二个部分，对于每一个A_i（令$X = A_i$），存储它前面离它最近的不符合等差数列的X的位置，然后只需要考虑[l,r]中是否有数$< l$。（但是要注意，对于同一个X，只能保存最右边的$A_i = X$的数据）

其实，如果用可持久化数据结构，可以做到在线。

时间复杂度$O(M\log M + Q\log M)$，空间复杂度$O(M + Q)$。

1.74 CF360D Levko and Sets

题目大意

Levko 有两个集合$\{A_1, A_2, \ldots, A_N\}$和$\{B_1, B_2, \ldots, B_M\}$和一个素数p。他又要生成N个集合，第i个集合的生成方式如下：

1. 这个集合一开始只有一个元素 1；
2. 我们从这个集合中任意选出一个元素c。对于所有的j，如果$\left(c \times A_i^{B_j}\right) \bmod p$不存在于当前的集合中，则将它加入当前集合；
3. 重复步骤 2，直到无法将任意元素加入集合。

求所有集合的并的大小。

$N \le 10^4$，$p \le 10^9$，时间限制 3s。

简要题解

对于第i个集合，我们可以得到的数是$A_i^{\sum_{j=1}^{M} k_j B_j} \bmod p$，由费马小定理和裴蜀定理可得：

若$r = \gcd(B_1, B_2, \ldots, B_M, p-1)$，那么集合中的数就是$A_i{}^r, A_i{}^{2r}, \ldots, A_i{}^{p-1}$。

令p的原根为g，设$A_i{}^r = g^{C_i}$，那么集合中的数就是$g^{C_i}, g^{2C_i}, \ldots, A_i{}^{\frac{p-1}{r}C_i}$。

令$e_i = \gcd(p-1, C_i)$，那么集合中的数就是$g^{e_i}, g^{2e_i}, \ldots, A_i^{\frac{p-1}{r}e_i}$。

于是问题就变成了在$1 \sim p-1$中有多少个数是某一个e_i的倍数。

由于所有的e_i都是$p-1$的约数，这个可以用容斥原理在$O(1500^2)$的时间内完成（小于10^9的数的约数个数不会超过 1500）：只要用 DP 的方法求出在$\frac{p}{e_i}$前面的系数是多少就行了。

那么如何求出e_i呢？只要枚举出最小的q_i使得$A_i^{rq_i} \equiv 1 (\bmod p)$，$e_i = \frac{p-1}{q_i}$显然成立（$q_i$也一定是$p-1$的约数啦）。

时间复杂度$O(1500N + 1500^2 + M)$，空间复杂度$O(N + 1500)$。

2 USACO Monthly Contest 试题泛做题解

2.1 DEC05 Cow Pattern

题目大意

有一个长度为N的串，每个元素均为$1 \sim S$的正整数。

Farmer John 有一个长度为K的“模式串”，每个元素均为$1 \sim S$的正整数。他要将模式串与原串的每个长度为K的子串进行匹配。

只要每个元素的相对大小与模式串中对应元素的相对大小全部相同，即视为匹配成功。

问原串中有几个位置与模式串可以匹配成功。

$N \le 100000$，$K \le 25000$，$S \le 25$，时间限制 1s。

简要题解

这道题在 2010 年莫涛的国家集训队作业中也有出现（不过莫涛进行了加强）。

那么怎么做呢？这显然也是一道匹配的问题，我们自然会想到 KMP 算法。

KMP 算法的流程如下(用 B 串去匹配 A 串)：

初始$j = 0$

for $i = 1$ to N do

  while $(j > 0)$ and $(A[i] \ne B[j+1])$ $j = pre[j]$

  if $A[i] = B[j+1]$ then $j = j + 1$

  if $j = M$ then 匹配数+1

可以看出，使用 KMP 要求“=”关系满足下列条件：

1. $X = Y$，$Y = Z \Rightarrow X = Z$
2. $X = Y$，$x = y \Rightarrow \overline{Xx} = \overline{Yy}$
3. $\overline{xX} = \overline{yY} \Rightarrow X = Y$（pre 数组的作用）

应用到本题中，我们可以发现这道题中的“=”关系同样适用以上的三个条件。于是可以将 KMP 进行扩展，将“字符的相等”变成“序列中当前元素的排名”。由于$S \le 25$，可以直接用$O(S)$的时间完成一次比较，否则需要用其它数据结构维护。

时间复杂度$O((N+K)S)$，空间复杂度$O(N+K+S)$。

2.2 OPEN07 Connect

题目大意

考虑坐标平面上所有满足$1 \le i \le R, 1 \le j \le C$的整点$(i,j)$。只有曼哈顿距离为 1 的点之间才能连边。给出初始的连边情况，同时进行若干次修改和询问。

修改是新建一条边或者拆毁一条已经存在的边，询问是询问两点之间的连通性（在询问连通性时纵坐标不允许超过给定两点的范围）。

$R \le 2$，$C \le 50000$，修改和询问总数$K \le 50000$，时间限制 1s。

简要题解

这道题就是 SHOI2008 堵塞的交通那道题的来源吧，典型的线段树题：对于区间$[i,j]$，维护$(1,i)(2,i)(1,j)(2,j)$之间的连通性。修改时就直接修改，然后在线段树上进行更新；询问(x_1,y_1)和(x_2,y_2)时就查询$[y1,y2]$的连通性即可。

SHOI2008 把那道题加了一点难度，将括号里的那句话去掉了。这时可以从别的地方绕过去。那么我们查询时除了查询$[y_1,y_2]$，还要查询$[1,y1][y2,C]$。

不过这个类型最难的题目还是 WC2009 最短路问题吧（$R \le 6$呢）。

时间复杂度$O(C + KlogC)$，空间复杂度$O(C)$。

2.3 MAR08 Land Acquisition

题目大意

需要购买N块土地，每块土地有长L_i和宽W_i。

如果一次购买一块土地，价钱为L_iW_i；如果一次购买几块土地，需要付的总价钱为这几块土地的$\max\{L_i\} \times \max\{W_i\}$。

问购买下所有土地所付的总价钱的最小值。

$N \le 50000$，时间限制 1s。

简要题解

首先，如果存在土地i,j使得$W_i \le W_j$且$L_i \le L_j$，则可以将i和j一起购买，这样就可以不考虑土地i了。排序之后扫一遍就可以去掉所有不需要考虑的土地。

现在如果你按照W_i降序排序，那么L_i同时也会按照升序排序。这样就可以进行 DP：

$$f_i = \min\{f_j + W_{j+1}L_i\}$$

这是非常经典的斜率优化 DP 模型，并且L_i单调递增。

将(W_{j+1},f_j)看做是坐标系上的点，那么$f_j = -L_iW_{j+1} + f_i$可以看做一条经过(W_{j+1},f_j)，斜率为$-L_i$的直线。我们需要让这条直线的纵截距最小。那么易知最优决策点在点集的下凸壳上。又由于L_i和W_i单调变化，所以最优决策点的位置也是单调变化的，只需要用一个单调队列维护即可。

时间复杂度$O(NlogN)$（DP 的复杂度为$O(N)$），空间复杂度$O(N)$。

2.4 JAN09 Safe Travel

题目大意

有一个N个点M条边的无向图。在2~N号点各有一头奶牛，它们都想到 1 号点。

但是它们不想走最短路径。准确地说，它不会走最短路径上的第一条边。对每头奶牛，求满足该条件的最短路。

保证从任何点到 1 号点的最短路径唯一。

$N \leq 10^5$，$M \leq 2 \times 10^5$，时间限制 1s。

简要题解

第一步，用 Dijkstra+堆求出 1 号点到所有点的最短路径，构造出最短路径树。那么对于每一头奶牛就不能走向它在树中的父亲顶点。那么我们对这棵树自底向上考虑：

首先是叶子结点u。我们第一步只能走非树边，设为(u,v,len)，那么实际长度就是$len+dis[1,v]$。（令dis[i,j]表示i号点到j号点的距离）我们要把所有的$(u,v,len+dis[1,v])$加入u的“某数据结构”中，并在该数据结构中查询最小值。

然后是非叶子结点u。第一步同样可以走非树边，设为(u,v,len)，所有也要把所有的$(u,v,len+dis[1,v])$加入u的“某数据结构”中。

但是u有儿子，第一步还可以走到它的儿子，再通过儿子的方式走。设它的儿子为x，如果在x的“某数据结构”中存有(x,y,len)，那么对应到u就是$(u,y,len+dis[u,x])$。我们同样要将它加入到u的“某数据结构”中。这一过程实际上就是将x的“某数据结构”中所有元素全部加上$dis[u,x]$，然后再合并到u的“某数据结构”中。最后在u的“某数据结构”中查询最小值。

以上方法存在问题。如果(x,y,len)中的y也在u的子树中，那么它实际的路径是$u \to x \to y \to u \to u$ 的父亲 $\to 1$。这是不可以的。此时就要将对应的$(u,y,len+dis[u,x])$从u的“某数据结构”中删除。实现这一点时，你可以用类似 Tarjan 算法离线求 LCA 的方法就能快速判断y是否在u的子树中。删除时可以偷懒，只有当它成为最小值时再删除。

我们现在对于u，只考虑了它沿树边走到某一个后代，然后沿非树边一下跳到u子树以外，再向 1 走的情况。那么是否有什么合法的路径我们没有考虑到？实际上是没有的。因为即使从u出发的第一步到达的是它的儿子，我们也不会沿非树边走到另一个在u子树中的点（否则直接从u出发沿树边走到它肯定更优）。

最后再来看一下这个“某数据结构”的性质：它需要支持求最小值，删除最小值，将其中所有元素加上给定值以及合并两个数据结构。如果写平衡树的启发式合并，时间复杂度就是$O(M\log^2 N)$；如果用左偏树这样的可并堆，就可以将时间复杂度降到$O(M\log N)$。

时间复杂度$O(M\log N)$，空间复杂度$O(N+M)$。

2.5 MAR09 Cleaning Up

题目大意

你要将N个数分成若干段，每一段的代价是该段中不同的数字个数的平方。求所有段代价总和的最小值。

$N \leq 40000$，时间限制 1s。

简要题解

DP 是很容易想出来的：f[i]表示将1~i分段的最小代价。

然后就有$f[i]=\min\{f[j]+(A_{j+1}\sim A_i\text{中的不同数字个数的平方})\}$，这样如果从后向前枚举j的话转移就是$O(N)$的，总时间就是$O(N^2)$。

但是有一个很好的优化：$f[i] \le i$（因为你分成i段代价才是i），所以如果A_{j+1}~A_i中的不同数字个数超过了$\sqrt{i}$，那就根本没必要考虑。

我们就只要实时维护当前离第i个位置最近的$\sqrt{N}$个不同的数最后出现的位置。这个我是用双向链表在$O(1)$的时间内完成维护的。

时间复杂度$O(N\sqrt{N})$，空间复杂度$O(N)$。

2.6 OPEN09 Tower of Hay

题目大意

奶牛们要用N垛干草搭一个塔，每垛干草有一定的宽度。N垛干草按顺序送来，后来的干草不能放在先来的干草下面。

这N垛干草必须全部用完，并且摆成的塔必须满足上窄下宽。问这座塔最高能有多高？

$N \le 100000$，时间限制 1s。

简要题解

一个容易想到的 DP 是f[i][j]表示以i~j的干草做底，用i~N的干草搭的塔最高的高度，但是这最好也只能得到$O(N^2)$的算法。我们要换一个思路。

直觉上说，塔最底层的宽度越窄，塔的高度也应该越高。实际上也可以证明，当塔最底层的宽度达到最窄时，高度也会达到最高。用归纳法证明如下：

1. 当塔高 1 层时显然正确；
2. 如果塔高不止一层，设底层最窄（并且尽量高）的塔以第1~N_1垛干草作底，实际上最高的塔以第1~N_2垛干草作底，则显然有$N_1 \le N_2$。
   由归纳假设可知这两座塔去掉最底层以后的子塔一定也是底层最窄的塔。
   由单调性可知，由第N_1+1~N垛干草搭起的最高的塔高度不会低于由第N_2+1~N垛干草搭起的最高的塔。证明完成。

于是我们换一个 DP 的思路：f[i]表示用i~N的干草搭塔，最底层的宽度的最小值。

令S[i]为i~N的干草宽度之和，则$f[i] = \min\{S[i]-S[j] | S[i] \ge S[j]+f[j]\}$。

然后由于S[i]的单调性，这个问题就可以用单调队列来优化了。

时间复杂度$O(N)$，空间复杂度$O(N)$。

2.7 OPEN10 Triangle Counting

题目大意

给定平面上的N个点，求这些点组成的三角形中有多少个包含原点（保证原点不会在任何一个三角形的边上）。

$N \le 100000$，时间限制 1s。

简要题解

正面统计比较困难，不如从反面考虑有多少个三角形不包括原点。

考虑这样的三角形，原点和三个点的连线的跨度小于 180° 。那么我们可以过原点做出一条直线，使得这三个点在这条直线的同侧。

如果按照极角排序，我们枚举三角形中极角最小的一个点，那么只需要找到极角相差 180° 的范围内有多少个点就行了（从中取两个点即可）。当枚举的这个点变化时，可行的范

围也随之单调变化。

时间复杂度O(NlogN)，空间复杂度O(N)。

2.8 DEC10 Threatening Letter

题目大意

给定字符串S和T，问最少需要多少个T的子串才能拼成S。

$|S|,|T| \leq 50000$，时间限制 1s。

简要题解

又是一个有大量“求一个串是不是另一个串的子串”操作的题目，我们依然用后缀数据结构。（这里当然是后缀自动机了）

首先我们求出g[i]表示用T的子串从S[i]出发开始匹配最远能匹配到哪里。这里我们建立T的后缀自动机。然后令 i 从 1 到|S|依次求出g[i]的值。由于g[i]显然不降，所以这样做的复杂度是O(S)的。

求出这个以后我们就开始求解，设f[i]表示最少需要多少个T的子串才能拼成S[i..|S|]。这个我们需要倒着求解，并且用线段树查询f[i]的区间最小值并实时维护。

时间复杂度$O(|S|\log|S|+|T|)$，空间限制$O(|S|+|T|)$。

2.9 JAN12 Cow Run

题目大意

奶牛们在长度为M的环形跑道上跑步，跑步的距离由 Farmer John 和 Bessie 玩牌决定。

玩牌共N轮。每一轮有 8 张牌，每张牌上有一个数字。Farmer John 决定保留前四张还是后四张。Bessie 再决定保留前两张还是后两张。这保留的两张牌从上到下为(x,y)，奶牛们之前已经跑的距离为d，那么奶牛就要再跑$dx+y$的距离。

最后奶牛离起点的距离如果不超过K则 Farmer John 赢，否则 Bessie 赢。

每次在 Bessie 操作前，Farmer John 不知道她的决策。那么给定 Bessie 每次的决策，Farmer John 每次的决策应该是什么才能保证自己赢？如果有多解输出字典序最小的那个。

$N \leq 14$，时间限制 1s。

简要题解

这很显然是一个博弈问题，博弈树一共有2N层，每个结点有 2 个儿子，表示保留前面的还是后面的。Farmer John 和 Bessie 决策时，都要考虑到自己决策后无论对方选什么自己总能赢。这样每一次都要遍历整棵博弈树，时间复杂度为$O(4^N)$。

这显然要超时，但是……好像根本没办法优化呀。博弈树的大小就是$O(4^N)$的。

然后出题人给出的解法是：每一步进行逻辑短路，并且随机枚举儿子的顺序。没了。

（为啥我看到这个想到了极大-极小搜索的$\alpha-\beta$剪枝？）

我们来分析一下这个脑洞很大的算法最坏情况下的期望时间复杂度：

设f(K),g(K)分别为当前 Farmer John 和 Bessie 决策时还剩K轮，遍历完整棵博弈树最坏情况下的期望代价。那么有以下递推式：

（下式中的下标 1 和 0 分别表示 Farmer John 赢和 Bessie 赢）

$$f_1(K)=2g_1(K), f_0(K)=\max\left\{g_0(K)+\frac{g_1(K)}{2}, g_0(K)\right\}=g_0(K)+\frac{g_1(K)}{2}$$

$$g_0(K)=2f_0(K-1), g_1(K)=\max\left\{f_1(K-1), f_1(K-1)+\frac{f_0(K-1)}{2}\right\}=f_1(K-1)+\frac{f_0(K-1)}{2}$$

边界条件$f_0(0)=f_1(0)=1$。

我们把$g_0(K)$和$g_1(K)$从上式中提掉，又得到以下递推式：

$$f_0(K)=\frac{9}{4}f_0(K-1)+\frac{1}{2}f_1(K-1), f_1(K)=f_0(K-1)+2f_1(K-1)$$

解出来$f_0(K)=O\left(\left(\frac{17+\sqrt{33}}{8}\right)^N\right)$，$f_1(K)=O\left(\left(\frac{17+\sqrt{33}}{8}\right)^N\right)$。

当$N=14$时$\left(\frac{17+\sqrt{33}}{8}\right)^N\approx 2254376.4829395$。哈哈，能过啦！

唉，不得不说，出题人脑洞太大了！

时间复杂度$O((\frac{17+\sqrt{33}}{8})^N)$，空间复杂度$O(8N)$。

2.10 DEC12 First!

题目大意

有N个字符串S_i，问有多少个串可以通过改变字母表的排列使其变成其中字典序最小的。

$N\le 30000$，$Sum=\sum_{i=1}^{N}|S_i|$，时间限制 1s。

简要题解

首先我们肯定要把这N个串的 Trie 建出来，然后在 Trie 上考虑。

首先，如果某个串的前缀也出现在这N个串中，则显然不行。否则考察从根走到它的路径上的所有点。显然每一个点表示的字母x一定都要排在它的所有兄弟结点表示的字母y的前面。这样我们连边$x\to y$表示优先级。如果最后得到的图中有环就显然不行，否则就可以。

时间复杂度$O(26^2Sum)$，空间复杂度$O(26Sum)$。

2.11 MAR13 Hill Walk

题目大意

平面上有N条互不相交的线段。Bessie 从第一条线段的左端点出发沿线段向右走，到达右边界后向下掉，直至落到另一条线段上，如果没有另一条线段则停止。问她一共走过了多少条线段。

$N\le 100000$，时间限制 1s。

简要题解

这道题的关键就是要知道从每一条线段出发，到达的下一条线段是哪一条。这个可以用扫描线的方法来解决：

一条线段会对应两个事件：在左端点处加入，在右端点处删除。将所有的事件按照出现的横坐标排序，然后按顺序在每一次删除时查询当前线段下面的第一条线段。由于线段互不相交，所以上下关系是确定的，直接用平衡树来维护即可。

知道了这个以后，剩下来的就是模拟了。

时间复杂度$O(NlogN)$，空间复杂度$O(N)$。

2.12 OPEN13 Figure Eight

题目大意

奶牛要在$N \times N$的大理石板上雕刻一个“8”。

一个“8”由上下两个水平的矩形框组成，并且上矩形的底边必须是下矩形顶边的子集。

某些格子上有瑕疵，不能在这里雕刻。雕刻一个“8”的得分为两个矩形框各自围住的面积（边框本身不算）的乘积。求奶牛可能的最大得分。

$N \leqslant 300$，时间限制 1s。

简要题解

我们分别考虑上矩形和下矩形。

令$f_1[i][j][k]$为上矩形演延伸到第i行（尚未封底），占据第j~k列时的最大得分。然后转移时有两种情况：该矩形在第i行之前已经存在，该矩形从第i行开始。下矩形同理有$f_2[i][j][k]$。

所需要的只是判断第i行第j~k列是否均无瑕疵，这个可以在$O(N^2)$预处理出每一个格子向左可以延伸而不碰到瑕疵的最大格子数，然后一次查询就是$O(1)$的了。

上下矩形合并的时候，枚举中间的那一行和下矩形的顶边在第j~k列，上矩形的底边就是它的子区间，这个也可以在每一行$O(N^2)$的时间内处理出来。

时间复杂度$O(N^3)$，空间复杂度$O(N^3)$。

3 Google Code Jam Finals 试题泛做题解

3.1 2008E The Year of Code Jam

题目大意

有一个$N \times M$的方格，每个格子可以涂成白色或蓝色，有些格子已经被涂色。

你将剩下来的格子涂色，问蓝色区域的周长最大是多少（可以有多个区域）。

$N,M \leq 50$，数据组数$T \leq 100$，时间限制 1s。

简要题解

这道题问蓝色区域的周长，换个角度去想就是蓝色和白色的最大割呀！

最小割好求，最大割又怎么求呢？2011 年国家集训队作业里李其乐的圈地计划和这道题非常类似，做法是进行棋盘二染色，并假设初始时它们都在 S 割。这样就将初始时白色与蓝色的割达到最大化。

如果某两个相邻的格子分在了 S 和 T 两个割里，那么它们的颜色就变成了相同的，答案就要减 1。这样就把二分图最大割转为了最小割！相邻的格子互相连容量为 1 的边，再根据题目要求把已经定下来颜色的格子强制加入某一割，然后求最小割即可。

最后要注意边界的问题，你需要在外面加上一圈，而且这一圈的格子都是白色。

时间复杂度$O(N^3M^3T)$，空间复杂度$O(NM)$。

3.2 2009A Year of More Code Jam

题目大意

一年有N天，有T个比赛。每一个比赛共比m_i场，分别在比赛开始后的第$d_1, d_2, \dots d_{m_i}$天（比赛开始的当天算第一天，即$d_1 = 1$），但是开始日期未知。

Little Josh 非常喜欢打比赛，如果一天打S场比赛他会获得S^2的愉悦值。他会计算他一年获得的愉悦值总和。

如果这T个比赛的开始日期都等概率地分布在这N天的每一天上，求出他这一年获得的愉悦值总和的期望（如果某一场比赛的某一次比赛在第二年则不计入）。

$N \le 10^9$，$2 \le m_i \le 50$，$T \le 50$，时间限制 1s。

简要题解

有一个很重要的公式是$n^2 = n + 2C_n^2$，然后就把S^2这个难以统计的量转为某一场比赛提供的愉悦值和某两场比赛在同一天提供的愉悦值。

对于一场在第d_i天进行的比赛，提供的愉悦值期望为$\frac{N-d_i+1}{N}$；

对于两场分别在第d_i天和在第d_j天进行的比赛（如果不是同一个比赛的不同场次），提供的愉悦值期望为$2\frac{N-\max\{d_i,d_j\}+1}{N^2}$。

剩下来就是注意一些细节了（其实根本用不着高精度）。

时间复杂度$O(Sum^2)$，空间复杂度$O(Sum)$（$Sum = \sum_{i=1}^{T} m_i$）。

3.3 2009B Mini Perimeter

题目大意

给定平面上的N个点，求周长最小的三角形（包括退化的）。

$N \le 100000$，时间限制 1s。

简要题解

这道题和最经典的最近点对问题好像啊！于是我们就按照最近点对的做法去做。

将平面上的点按照 x 坐标排序，然后分成左右两半分治。假设左右两半分治算出的最小周长为$2d$。然后再考虑跨越左右两半的情况。

首先只要考虑距离分界线不超过d的点，然后对于每一个点，也只要考虑 y 坐标比它小，但是差距不超过d的点，于是就有了一个$2d \times d$的矩形。与最近点对的情况类似，这个矩形内的点只有常数个（根据 2010 年吴翼的国家集训队作业，点数不超过 16）。而且最多是左半边 8 个，右半边 8 个，枚举量不超过$8^2 = 64$。

在清橙上这道题保证没有重点，枚举量就变成了$4^2 = 16$。

接下来就是如何找 y 坐标比它小的点。如果每次对 y 坐标排序，那么复杂度是$O(N\log^2 N)$的；但是左右两半的 y 坐标已经排好序，我们实际上只要归并，复杂度就是$O(N\log N)$了。

对于分治的边界，你可以在点数不超过 5 时直接暴力处理（为什么是 5？是为了避免出现只有一两个点的最小周长无穷大的情况）。

时间复杂度$O(64N\log N)$，空间复杂度$O(N)$。

3.4 2009C Doubly-sorted Grid

题目大意

一个$N \times M$的方格纸的每个格子填上一个小写字母。如果每一行从左到右字母单调不降，每一列从上到下字母单调不降，那么我们称其为“双排序”的。

给定一个部分方格已经填过字母的方格纸，问有多少种方法将其填成“双排序”的。

$N,M \leq 10$，数据组数$T \leq 3$，时间限制 4s。

简要题解

这是一道轮廓线型的状压 DP。于是设状态为：f[i][j]表示当前填完了字母1~i，轮廓线j以上的区域都填满的方案数。那么怎么转移？

直接在轮廓线与轮廓线之间转移。但是轮廓线最多有$C_{20}^{10} = 184756$种，代价太吓人了。

在求 Hamilton 回路数的状压 DP 中也会遇到这个问题，我们的做法是按格转移。但是按格转移就要规定一个转移的顺序，否则会出现重复计数。在求 Hamilton 回路数时，我们是直接从上到下从左到右一格一格做的。这道题里也可以这样做。

对于每一个字母和每一条轮廓线，我们从第一行开始按照从上到下从左到右填空。这样状态就要加一维：f[i][j][k]表示当前填完了字母1~i，轮廓线j以上的区域都填满，只有1~k行填了字母i（表示下一次填只能从第k行开始填）。这样每一行能转移的格子最多只有一个，时间复杂度就降到了$O(26NC_{N+M}^{N})$。

为什么不用记录列呢？因为由轮廓线可以算出来每一行到达了哪个格子。

在写的时候注意的地方：如果你用一个二进制串表示轮廓线，那么轮廓线的转移顺序就是二进制串表示的数字的大小顺序。这样写起来比较方便。你还可以对于每条轮廓线预处理出每一行应该转移哪一个格子以节省时间。

时间复杂度$O(26NTC_{N+M}^{N})$，空间复杂度$O(26NC_{N+M}^{N})$。

3.5 2009D Wi-fi Tower

题目大意

有N座无线发送塔，每个塔有坐标及发送范围。当前每座塔都使用 A 协议。

现在要将某些塔改用 B 协议。每座塔升级会有正或负的利润。而且，如果一座塔使用了 B 协议，它发送范围内的所有塔就都要使用 B 协议（反之不亦然）。求最大利润。

$N \leq 500$，数据组数$T \leq 55$，时间限制 1s。

简要题解

这题是一眼的网络流题了，而且是最小割题。

使用 A 协议视为 S 割，使用 B 协议视为 T 割。那么对于塔j和它发送范围内的塔i就连边$(i,j,+\infty)$（如果j在 T 割，i就必须在 T 割）。

先将所有正的利润加进Ans，然后正的利润表示加入 S 割的代价，连边(i,T,利润)；负的利润表示加入 T 割的代价，连边$(S,i,-$利润$)$。

最后写网络流就可以了。虽然边数是$O(N^2)$的，但仍然能跑出来。话说这道题我一开始写时狂 T 不止，就把它撂到一边不管了。然后几天以后换了一种写法重写以后立刻 AC。真是神奇！

对边数有优化：如果有连边$B \to A$和$C \to A$，并且$|AB| > |AC|$，$\angle BAC \leq 60°$，那么必有

$|AB| > |BC|$。这时根本不需要连$B \to A$。（因为连$B \to A$和$C \to A$与之等价）这样对于每个点最多只有 6 条边。边数就降到$O(N)$了。

时间复杂度$O(N^4T)$，空间复杂度$O(N^2)$。

3.6 2009E Marbles

题目大意

有N种颜色共$2N$个珠子顺次排放在$(1,0)\ (2,0)\ \ldots\ldots\ (2N,0)$，每种颜色的珠子有两个。

现在要将同色的珠子用线连起来，要求除了在开始和结尾连接两个珠子的地方之外，不能与直线$y = 0$有任何一个交点，同时要求连线上的任意部分必须水平或者垂直，并且所有的拐点都必须是整点。问最优情况下，所有连线上纵坐标最大的点的纵坐标，与纵坐标最小的点的纵坐标的差值最小是多少。

$N \leq 500$，数据组数$T \leq 50$，时间限制 1s。

简要题解

直观来看，不是在上面连线就是在下面连线，这样就可以归结为一个有关二分图的问题。

首先是判断能否完成连线。建立一个N个点的图，一种颜色对应一个点。如果有两种颜色的珠子对应的区间相交，就在它们之间连边。然后判断这个图是否是二分图就可以了。

这样连边以后，整个图会是若干个连通块。设每一个连通块的范围为$[L_i,R_i]$（L_i表示该连通块中横坐标最小的珠子的横坐标，R_i表示该连通块中横坐标最大的珠子的横坐标），可以证明：如果连通块A中有一个区间被连通块B中的某个区间包含，那么连通块A中的所有区间都必然被这个区间包含（否则就会与之相交）。

然后有了一个很强大的结论：不同的连通块的$[L_i,R_i]$不会相交！

这样我们加入区间$[0,2N+1]$，然后把所有连通块按照区间的包含关系形成树形结构，在树上进行 DP：设$f[i][j]$表示在只考虑$[L_i,R_i]$内部的区间时，向上延伸的高度不超过j时，向下延伸的最低高度。

对于一个连通块i，枚举向上延伸的高度j，然后将它的所有儿子塞进去。在这里还要进行一个小 DP，计算出每个儿子可以向上延伸的最大高度，并求出 DP 值。最后还要进行松弛：$f[i][j] = \min\{f[i][j],f[i][j+1]\}$，$f[i][f[i][j]] = \min\{f[i][f[i][j]],j\}$（上下反过来算）。

最后一步求Ans的过程比较简单就不说了。

总而言之，这道题写起来相当麻烦，有许多细节要想清楚。太佩服当场 AC 的人了！

时间复杂度$O(N^2T)$，空间复杂度$O(N^2)$。

3.7 2010A Letter Stamper

题目大意

你要输出一个长度为N的只含“A”“B”“C”的字符串。你有一个栈，每次执行下列三种操作中的一个：将一个字母加进栈中，输出栈顶的字母，删除栈顶字母。

初始时栈为空，你执行完所有操作后栈也要为空，求能输出给定串的最少操作次数。

$N \leq 2000$，时间限制 1s。

简要题解

关于栈内元素的状态有几个重要的性质：

1. 如果将 X 加入栈中，那么下一步就是输出 X；
2. 不会有两个连续的相同字母；
3. 不会出现形如 XYX 的情况。
   证明：如果出现这种情况，那么我们在将第二个 X 加入栈中时改为弹出 Y。在需要用到 Y 时再将 Y 加入，这样操作数不变。
   这样栈中元素的状态就只能是形如 XYZXYZ…的形式，一共只有 6 种可能。

然后令f[i][j][k]表示当前打印了i个字母，栈的高度为j，栈的状态为k。然后转移时注意搞一搞就可以了。

时间复杂度$O(N^2)$，空间复杂度$O(N^2)$。

3.8 2010C Candy Store

题目大意

你经营一个糖果店。你每天早上进货时只能整包整包地进，并且这些包裹不能拆开。但是你可以从供应商手中买来任意多个装有任意整数个糖果的包裹。

这天有k个人来买糖，每个人买糖果的数量均是1~C的整数，但你不知道具体的数值和来人的顺序。你需要保证无论每个人想花多少钱买糖，你总是可以给他们对应质量的糖果。问你最少要买多少包糖才能满足要求。

$k \le 1000$，$C \le 10^{12}$，数据组数$T \le 100$，时间限制 1s。

简要题解

这题属于不膜拜题解不行的题目。

下面给出一种可行的策略（怎么想出来的）：

假设我们目前有若干包总数为N的糖果，每一包的糖果数量不超过$X = \left\lfloor \frac{N}{k} \right\rfloor + 1$，并且如果k个人的买糖总数不超过N，总能够按照“贪心原则”依次满足他们的要求。（“贪心原则”指的是每一次给买家不超过其要求的尽可能大的包裹，直至他的要求满足为止）

我们又进了一包数量为X的包裹，总数就变成了$N + X$。现在每一包的糖果数量仍然不会超过$\left\lfloor \frac{N+X}{k} \right\rfloor + 1$。这时，如果k个人的买糖总数不超过$N + X$，就依然总能够按照“贪心原则”依次满足他们的要求。证明：

如果有人的要求不少于X，那么这个数量为X的包裹就归他了。去除了数量为X的包裹，问题就变回了没有X的情况。这是满足要求的。

如果所有人的要求都不超过$X - 1$，那么总量就不超过$k(X-1) = k\left\lfloor \frac{N}{k} \right\rfloor \le N$，问题等价于没有X的情况。这也是满足要求的。至此归纳完成。

那么这个归纳法的初始条件是什么呢？就是$N = 0$的情况（没有糖果，没有人买糖果）。

于是出现了一个可行的策略：

初始 $Sum = 0$

While （ $Sum < kC$ ）

买一包数量为$\left\lfloor \frac{Sum}{k} \right\rfloor + 1$的糖果

$Sum += \left\lfloor \frac{Sum}{k} \right\rfloor + 1$

下面再来证明这个策略是最优的（怎么证出来的）：

设我们的策略得到的糖果包集合为$\{a_1, a_2, \ldots, a_p\}$，将元素从小到大排序。

假设另一个可以满足要求的糖果包集合为$\{b_1, b_2, \ldots, b_q\}$，并且$q < p$。将其元素从小到大排序，并且逐一与$\{a_i\}$进行比较。

假设有一个最小的$b_m > a_m$，那么令$Sum = \sum_{i=1}^{m-1} b_i$，则$b_m > a_m = \left\lfloor \frac{Sum}{k} \right\rfloor + 1$。此时，如果k个买家都要购买$a_m$的糖果，就只能使用前$m-1$个糖果包，但总和$Sum < ka_m$。矛盾！

那么所有$b_i \le a_i$均成立，但由$q < p$可得总和$Sum = \sum_{i=1}^{q} b_i \le \sum_{i=1}^{q} a_i < kC$。矛盾！

证毕！我们的策略就是最优策略！

然后剩下来的就是编一个两百多 B 的代码了。

时间复杂度$O(kTlogC)$，空间复杂度$O(1)$。

3.9 2011A Runs

题目大意

有一个由小写字母组成的字符串。每一个极大的连续相同子序列被称为一个“run”。

问将该字符串重排以后，有多少种不同的排列与原字符串有相同数量的 run。

两个字符串不同的定义是两个字符串在某个相同位置上的字母不同。

原字符串长度L不超过 450000，run 的数量N不超过 100，时间限制 1s。

简要题解

由于两个字符串不同是指字母不同而不是指排列不同，我们应当从字母的角度考虑：

先将字母 a 放进字符串中，再把字母 b 插进去，……，最后放字母 z。

于是有了一个 DP 状态：f[i][j]表示当前放完了前i个字母，已经组成的串有j个 run 的字符串个数。那么怎么转移呢？

首先，我们事先统计出第i个字母的个数s和前$i-1$个字母的总个数T。

如果从$f[i-1][j]$向$f[i][j']$转移，那有$j+1$个位置插入字母会导致 run 数加 1（加在某个 run 的边界），有$T-j$个位置插入字母会导致 run 数加 2（加在某个 run 的内部）。

要枚举这两种情况的个数k和l，然后就有：$f[i][j+k+2l] += f[i-1][j] C_{j+1}^{k} C_{T-j}^{l} C_{s-1}^{k+l-1}$。

时间复杂度$O(NL + 26N^3)$，空间复杂度$O(NL + 26N)$。

3.10 2011C Program within a Program

题目大意

有一个图灵机的模型。你的任务是让图灵机顺利运行，读写头向右移动N格以后停止。

你可以写不超过 30 条规则来操作图灵机，规则有两种：

1. $<S><M> \rightarrow <D><NS><NM>$，表示如果此时读写头状态为S，纸带上这一格的标记为M，则图灵机将这一格的标记改为NM，读写头状态改为NS，最后向方向D走一格（方向有两种：E向右，W向左）。

2. $<S><M> \rightarrow R$，表示如果此时读写头状态为S，纸带上这一格的标记为M，则图灵机停止。

读写头初始状态为 0，纸带上的所有点初始标记为 0。

$N \le 5000$，输出规则的时间限制 1s，规则总数不超过 30，图灵机的运行步数不超过

1.5×10^5，读写头的所有状态和纸带上的标记绝对值不超过10^6。

简要题解

有一个严峻的问题：如何记录我已经走了多少格（或者我还需要走多少格）？

由于规则的数量限制太死，显然不可能用读写头的状态记录，我们只能用纸带去记录。

那么怎么记录呢？考虑二进制串。那么就可以设置出一个大致的方法：

首先将N用二进制写在纸带上。然后每一次先$N--$，如果无法减（$N=0$）则在当前位置停止，否则减过以后将整个二进制串向右移动一格。

大致的思路出来了，你就可以仔细地设计一下（这一步还是相当麻烦的）。

这是我设计的读写头状态表：

2 开头的与右移相关：20“右移 0”，21“右移 1”，22“右移 2”，25“达到右边界”；

3 开头的与减法相关：30“要借位”，31“不要借位”。

下面是$N=12$的情况（这里用 2211 表示N的二进制 1100）：

$0\ 0\to E\ 1\ 0$，$1\ 0\to E\ 2\ 2$，$2\ 0\to E\ 3\ 2$，$3\ 0\to E\ 4\ 1$，$4\ 0\to E\ 25\ 1$。

（这几个规则是把N的二进制表示写在纸带上，指针到达右边界）

$25\ 0\to W\ 30\ 0$（到达右边界后开始减法，向左借位）

$30\ 1\to W\ 30\ 2$（这一位为 0，改成 1，向左边继续借位）

$30\ 2\to W\ 31\ 1$（这一位为 1，改成 0，向左边走就不需要借位了）

$31\ 1\to W\ 31\ 1$，$31\ 2\to W\ 31\ 2$（不需要借位）

$31\ 0\to E\ 20\ 0$（到头了，就应当开始将二进制串右移）

$30\ 0\to R$（如果到头了还是要借位，就说明$N=0$，此时应当立即在这里停止）

$20\ 1\to E\ 21\ 0$，$20\ 2\to E\ 22\ 0$（右移时要将当前位改成 0）

$21\ 1\to E\ 21\ 1$，$21\ 2\to E\ 22\ 1$（右移时要将当前位改成 1）

$22\ 1\to E\ 21\ 2$，$22\ 2\to E\ 22\ 2$（右移时要将当前位改成 2）

$21\ 0\to E\ 25\ 1$，$22\ 0\to E\ 25\ 2$（到达右边界）

这样即使$N=5000$，也只要用 29 条规则，而且刚刚好满足步数要求。当然你也可以用三进制表示，更省运行步数。

时间复杂度$O(\log N)$，空间复杂度$O(\log N)$。

3.11 2013D Can't Stop

题目大意

有N个位置，每个位置有D个数。你可以指定手上的K个数的值，然后找到一段连续区间，满足每个位置都有一个数在你手上出现。

求你能找到的最长的连续区间。

$N\le 10^5$，$D\le 4$，$K\le 3$，数据组数$T\le 10$，时间限制 1s。

简要题解

这道题的数据范围比较奇怪，N很大，D和K却特小，好像没什么太好的办法。

只能枚举起始位置i，然后从位置i选一个数加入手上的数，然后向右扩展。如果某个位置不能用手上的数表示，再从这个位置选一个数加入手上的数……重复三遍。由于每一次选数时要枚举选哪一个数，总时间复杂度$O(N^2D^K)$。简直糟透了。

有一个看似很没用的小优化：如果初始位置为i，不要选择在位置$i-1$出现的数。原因很简单，否则从$i-1$开始就已经枚举过了。但是莫名其妙的是，这样就将整个时间复杂度降

了一阶！变成了$O(ND^K)$。简要的证明如下：

我们想象从位置i出发枚举到位置j，如果用同样的K个数从位置j出发向左，一定能到达初始位置i，并且不能到达位置$i-1$。但是从位置j出发向左最多只有$O(D^K)$中不同的情况，也就是说最多只能到达$O(D^K)$个不同的位置。反过来看就是说，最多有$O(D^K)$个位置出发可以枚举到位置j！然后就没有了。

还有常数级别的小优化，比如说不选相同的数。

时间复杂度$O(ND^KT)$，空间复杂度$O(ND+K)$。

3.12 2013E Let Me Tell You a Story

题目大意

N个正整数组成一个序列A_i。我们每次删除一个数，直至形成一个单调不增序列为止。问有多少种不同的删除方法（顺序不同也算不同，不同位置的相同数字也算不同）。

$N \le 2000$，$1 \le A_i \le 10000$，数据组数$T \le 5$，时间限制 5s。

简要题解

我们考虑最后形成的单调不增子序列。

假设这个子序列长度为k，设长度为k的单调不增子序列个数为F_k，那么删除的方法总数为$(N-k)!\,F_k$。但是要考虑非法情况：从长度为$k+1$的单调不增子序列得到的，于是就需要减去$(N-k-1)!\,(k+1)F_{k+1}$。

最后的结果$Ans=\sum_{k=1}^{N}(N-k)!\,F_k-(N-k-1)!\,(k+1)F_{k+1}$，问题就变成了求$F_k$。

这个很简单，设f[i][j]表示以A_i为末尾，长度为j的单调不增子序列的个数，普通的做法为$O(N^3)$，你只需要用树状数组维护就可以优化至$O(N^2\log N)$。

时间复杂度$O(N^2T\log 10000)$，空间复杂度$O(N+10000)$。

3.13 2014C Symmetric Trees

题目大意

给定一棵由N个顶点组成并且顶点被染色的树，问它能否被放到水平面上，使得整个图形关于一条竖直的线对称。

$N \le 10000$，数据组数$T \le 3$，时间限制 1s。

简要题解

题目条件是“关于一条竖直的线对称”，那么对称的两棵子树必然是全等的。我们首先一定要考虑如何判断子树的全等。我的方法是进行 Hash（这里是对有根树进行 Hash），怎么 Hash 随你便。

我们设 1 为根，在$O(N)$的时间内计算出每一棵子树以及每一棵上方子树的 Hash 值。然后就开始考虑如何满足题目条件。

1. 没有任何一个顶点在对称轴上。
   这种情况很简单，只需要依次判断每一棵子树和对应的上方子树是否全等即可。
2. 有若干个顶点组成一条链在对称轴上。
   这种情况，我们要找到这条链上在树中深度最浅的点。它需要满足的条件是：有不超过两棵子树是对称的，然后其它的子树能够两两全等配对。

我们只要穷举每个结点看能否满足这个条件即可。

接下来就要解决如何判断一棵子树是不是对称子树的问题。这个问题也很好做，它需要满足的条件是：有不超过一棵子树是对称的，然后其它的子树能够两两全等配对。

于是只要一边 DFS 一边自底向上处理就可以了。

时间复杂度O(NT)，空间复杂度O(N)。

3.14 2014D Paradox Sort

题目大意

foreseeable 有N个东西，在任意两个东西中他都有比较喜欢的一个（虽然他不一定有最喜欢的一个）。

你将这N个东西排列成一排依次给他。他每得到一个东西就会与他手上已有的那个进行比较，只保留他更喜欢的那个。

问你是否存在一种排列使得他最后手上留下的东西恰好是第K个。若存在，请求出字典序最小的排列。

$N \leq 100$，数据组数$T \leq 100$，时间限制 1s。

简要题解

首先来看排列是否存在。对于两个东西a和b，如果他更喜欢a，则从a向b连边。下面来证明：如果从K出发可以遍历所有结点，则存在；否则不存在。

如果能遍历，可以证明后序遍历生成树即可。用归纳法很容易证明后序遍历任意一棵子树，最后手上的东西一定是该子树的根。

如果不能遍历，假设不能到达结点x，那么对任意排列考虑最后手上的东西。如果是x，显然不行；如果不是x，我们考虑那个替换掉x的东西y。显然结点y也不能从K到达（否则就有$K \to y \to x$）。然后再考虑y。依此类推，那么最后手上的东西也一定不是K能到达的，更不可能是K。

解决了存在性的问题，再来考虑字典序。

保证字典序的一般方法就是对每一位枚举这一位的取值，看是否可行；若可行则继续考虑下一位。于是问题又变成了在已知一段前缀的情况下判断是否可行。

这个问题比刚才稍微复杂一些。我们来看看现在的那棵生成树与过去有什么区别。首先有一些结点已经不能再访问（因为它们已经出现在了前缀中）。然后就是现在手上有的东西X。

直观想一想，从X出发最多只能向下走一层（如果一层走不到，就说明这个东西将替换掉X）。于是就是判断剩下来的结点能否从K出发遍历到（不经过已经出现在前缀中的结点），或者是X的儿子。

如果存在一个结点以上两个条件都不符合，则可以证明不可行（与上一个证明同理）；否则先去取那些只能从X出发遍历到的结点，然后再考虑生成树的后序遍历。

注意一点，如果X本身无法从K出发遍历到，也是无解的。

算一下时间复杂度相当高，但是还是很快就能出解。

时间复杂度$O(N^4T)$，空间复杂度$O(N^2)$。

3.15 2014E Allergy Testing

题目大意

Kelly 对N种食物中的一种过敏，她要通过试验确定是哪一种。

她一次试验可以吃任意多种食物，然后等待A天之后才能知道是否过敏。如果出现过敏症状，她需要$B-A$天恢复才能继续试验（从开始吃算起一共B天）。

问她在最坏情况下最少要多少天才能得出结论。

$N \le 10^{15}$，$A \le B \le 10^{12}$，数据组数$T \le 10$，时间限制 1s。

简要题解

直接的思路是对食物 DP：f[i]表示从i种食物中分辨出过敏原的最少天数。

然后 DP 方程为：$f[1]=0$，$f[i]=\min_{1\le j\le i-1}\max\{f[j]+A, f[i-j]+B\}$。

但是这个方程时间复杂度是$O(N^2)$。即使有优化（确实存在优化：令g[i]对计算f[i]的最优决策，则有$g[i]\ge g[i-1]$，可以优化至$O(N)$）也无法应付如此吓人的N。

第二个思路是对天数 DP：f[i]表示在i天之内最多能从多少种食物中分辨过敏原。

然后 DP 方程为：$f[i]=1$（$A-B\le i<A$），$f[i]=f[i-A]+f[i-B]$。

我们只需要二分天数Ans，每一次计算出来以后比较f[Ans]与N的大小即可。问题就变成了如何在比O(Ans)更快的时间内求出f[Ans]。（如果A和B相当大，那么Ans也将很大）

我们可以用类似于记忆化搜索的方法。因为每次不是减去A就是减去B，我们在求f[Ans]的过程中只会遇到$f[Ans-KA-LB]$这样的状态。虽然这样并不足以解决本题，但是为真正的解法提供了很好的启发。

考虑记忆化搜索最终的终点$f[i]=1$（$i=Ans-KA-LB$）：

如果转移来的最后一步是A，那么$0\le i<A$；

如果转移来的最后一步是B，那么$A-B\le i<A$。

可以发现，在这两个不等式中，如果L的值定下来了，K的取值范围就都是一个区间。而且对于给定的K和L，到达f[i]的方法数分别是C_{K-1+L}^{L}（最后一步是A）和C_{K+L-1}^{L-1}（最后一步是B）。

由组合恒等式$\sum_{N=N_{min}}^{N_{max}} C_N^M = C_{N_{max}+1}^{M+1} - C_{N_{min}}^{M+1}$可知，计算一次f[Ans]的复杂度是O(L)的。

显然f[Ans]是关于L呈指数增长的量，故L的范围是O(logN)的。

至此问题完全解决！

如果担心二分答案算出的f[Ans]可能会爆掉（如果你初值设的是10^{15}之类的），我的解决方法是可以先让$Ans=1,2,4,8,\ldots$，直到$f[Ans]\ge N$再开始二分。

最后注意$N=1$时$Ans=0$（因为你的二分是从 1 开始的）。

时间复杂度$O(T\log^3 N)$，空间复杂度$O(1)$。

3.16 2014F ARAM

题目大意

在英雄联盟里有N个英雄，你使用每个英雄的胜率A_i是给定值。

现在有一个叫“ARAM”的玩法，每一盘游戏你都会等概率随机分到一个英雄。

如果分到胜率低的英雄你会不爽，但是你可以花 1 元让系统再等概率随机分一个英雄给你。如果你赢了一盘，你可以得到$\frac{1}{G}$元。但是钱数的上限为R，如果你有R元，再赢一盘仍然有R元。

初始时你有R元，询问在最优策略下玩无限多盘的期望胜率。

$N\le 1000$，$R,G\le 20$，数据组数$T\le 10$，时间限制 5s。

简要题解

这道题让人头疼的地方是玩无限多盘的期望胜率怎么表示。然后有一个很神的方法，二分胜率Q，然后令一盘的得分（使用英雄i时）为$A_i - Q$。那么只要计算出玩无限多盘的期望得分是否大于 0 就行了（期望可以加减，显然比概率好算多了）。

那么玩无限多盘的期望得分又怎么计算呢？这时期望得分就只与我们手上的钱数有关系。可以令$f[i + \frac{j}{G}]$表示当我们手上有$i + \frac{j}{G}$元时……的最大期望得分。

那么省略号里又填什么呢？如果有R元，我们自然想知道手上的钱又回到R元时的最大期望得分。（因为无限多盘就相当于手上有R元，玩了若干盘以后手上的钱回到R元……不断循环的过程）这样我们只要判断最终算出来的$f[R]$是否大于 0 就可以了。

那如果不是R元而是$i + \frac{j}{G}$元呢？那肯定不是回到这个钱数（否则$f[R]$怎么算呢？）这里应该填什么呢？实际上应该是再挣$\frac{1}{G}$元达到$i + \frac{j+1}{G}$元的最大期望得分。

这样状态转移就很好想了：不能花钱就只能硬着头皮玩一盘，能花钱就枚举我被分到哪些英雄时花钱重选（显然是胜率前几低的）。就有了如下的方程：

$$f\left[0 + \frac{j}{G}\right] = \frac{1}{N}\sum_{L=1}^{N}(A_L - Q)\ \ (0 \le j < G)$$

$$f\left[i + \frac{j}{G}\right] = \max_{0 \le k \le N}\left(\frac{k}{N}\sum_{L=0}^{G} f\left[i + \frac{j - L}{G}\right] + \frac{1}{N}\sum_{L=k+1}^{N}(A_L - Q)\right)(1 \le i < R, 0 \le j < G)$$

$$f[R] = \max_{0 \le k \le N}\left(\frac{k}{N}\sum_{L=0}^{G-1} f\left[R - \frac{L}{G}\right] + \frac{1}{N}\sum_{L=k+1}^{N}(A_L - Q)\right)$$

时间复杂度$O(60NRGT)$，空间复杂度$O(N + RG)$。