1001 Alice Game

我们将一行有n个怪物的局势记作K_n。考虑$k=2$时，一行有5个怪物的情况K_5，当一次操作之后，可以有以下这些局面：K_1+K_2, K_2+K_1。这里的加法是局势之和，亦即Nim加(xor)。

把K_n的nim值记作y_n，有：$y_n=mex(y_i+y_{n-i-k})$，其中$n>k$，i与$n-i-k$均不能等于0，且i与$n-i-k$取遍所有可能的正整数。对这个序列打表，容易发现nim序列是由$11[1]01[1]12[2]21[2]22$这一循环节构成的，其中，$[x]$代表一段长度为$k-2$的，形如$xxx\dots x$的连续段。在每个循环节中，只有一处nim值为0，即$k+1$处。除了该处，其他位置均为先手必胜。

尽管这类游戏中的很大一部分展现出算术周期性，但像题目中这样规律的周期变化仍是不多见的。下面是两个类似的有趣例子：

假如将题目中的操作一修改成严格小于长度k的连续段才可以被消灭，把操作二修改成可以取走长度等于k或是$k+1$的段，其余规则不变，那么循环节将变为$11[1]11[1]22[2]22[2]44$，但y_{k+2}与y_{k+3}是例外，它们等于0。

假如在上述修改规则的基础上，给操作一加上一条可以取走长度等于$k+1$的补充规则，那么循环节将变为$11[1]11[1]22[2]22[2]4$，但$y_{k+2}=0$是例外。

如果仔细考虑原游戏容易发现，假如我们要在纸上计算y_n，并且我们已经有序列$y_1, y_2\dots y_{n-1}$，我们可以在一条纸带上正向写出这个序列，在另一个纸带上反向写出这个序列，然后把这两个纸带放置在一块，就像这样：

正向的序列：110112212211

反向的序列：112212211011

我们将两个纸带错位对齐，用上下数字的nim和的集合求mex就能得到y_n。这两个纸带提示我们，假如我们猜测序列有一个周期p，最后一个不符合这个周期的位置是i，每次操作最多可以取走的数量是k，则我们要验证这个序列后续是否拥有周期的最后一个位置应当是$y_{2i+2p+k}$。这是一个较为普遍的结论，对类似的将一堆物体分成几堆（事实上，它们的原型应当是$Kayles$游戏的变种八码游戏）的游戏都成立。

1002 Binary Number

$n=1$时，输入只可能是1，若k为奇数则输出0，反之输出1

$n>1$时，需要从高位往低位来看，把高位连续段的0变成1是最优的。例如1001011，当前一步操作把$[2,3]$反转时最优的。若k小于连续0的段数，则从高位往低位把连续0变为1；若k大于连续0的段数，则全部变为1即可，因为多出来的次数可以被消去。若多出偶数次操作，则全部选$[1,1]$可以互相抵消，若多出奇数次，先消去偶数的部分，剩下多出一步，同样可以消去，例如$[3,6]$操作可以拆开成$[3,4]+[5,6]$，$[3,3]$操作可以变成$[2,3]+[2,2]$

还有一种特殊情况，$s=111..11$且$k=1$，此时只能翻转s_n变为$111..10$

时间复杂度和空间复杂度均为$O(n)$

1003 Counter Strike

首先考虑一个子问题：对于一个无向连通图，图上有部分点是询问点，你能否删除一个点及其所有边，使得剩余询问点两两不连通？当点集大小小于等于2显然存在。**当询问点数量>=3时，我们可以唯一确定这个被删除的点，或者无法满足要求。**

对于判断询问点数量大于3的图，这里提供几种思路

1、随便取三个点确定中心枢纽后把他删除，然后判断剩余点的连通性。

2、判断圆方树上的虚树是菊花（只有一个点度数不为1），且菊花中心点（唯一的度数不是1的点）是圆点。（出题人最初想法）

3、同样是圆方树上的虚树，统计两两询问点之间的路径经过每个点的条数。

令$Q=$总共有q个点的询问点集合，$sum[x]=(u\to v)$的路径经过x点的数量，$u,v\in Q,u\neq v$

可以发现$(u\to v),(v\to u)$的都被计算上。若$sum[x]=q*(q-1)$，即任意两个询问点之间的路径都要经过x，且x是一个圆点，则点x是一个合法的删除对象。（std的实现方法）

4、其他在圆方树上求lca的判断方法

接下来回到原问题，由于水平有限，且这题在赛前一周紧急大改，在赛前两天才写对std造完数据，出题人使用了复杂度并非最优的二分答案。

由于要求按顺序删除点，不难发现答案是单调的，可以二分答案

设$check(x)$=购买了x个useless kits，删除了$h_1,h_2,\ldots,h_x$之后，是否能够再删除一个点使得剩余的$h_{x+1},h_{x+2},\ldots,h_k$两两不连通。

显然有 $\forall x\in[0,k-1]:check(x)=true\Rightarrow check(x+1)=true$

接下来考虑如何check

删除了部分点，会得到多张连通子图，对于一张子图，我们在上面已经讲了判断方法。对于多个子图，check需要满足要求是不存在不能通过删除一个点来满足要求的子图，且需要删除一个点的子图不超过1个。

可以删除点和边之后，建圆方树森林，遍历/并查集得到连通块，对每个连通块的询问点建虚树并判断。

std的时间复杂度为$O(n\log^2 n)$，空间复杂度$O(n+m)$

二分答案一个$\log n$，check函数中，建虚树时要按dfs序排序，还要求lca，复杂度为$O((n+m)logn)$，check其他部分为$O(n+m)$，总复杂度为$O((n+m)\log^2 n)$，也可以优化为$O((n+m)\log n)$

验题人提供的题解：从后向前加点 （非常感谢验题人提供的题解和std）

从后向前加点，用并查集维护其联通性，分类讨论当前加入点和边的情况:

1.如果存在至少两个联通块里的给定点的个数大于等于2，那显然寄了，怎么删都没办法。

2.如果只存在一个联通块里的给定点的个数等于2，那显然随便删

3.如果存在一个联通块里的给定点的个数大于2，我们可以直接确定被删的点，把他删掉好了，删完之后之后再暴力往前加点判断联通性即可

(其实细节蛮多的)

找被删除的点和圆方树的建立都是$O(n+m)$的，总的复杂度是线性的。

1004 Card Game

由于牌堆（pile）内要求连续且递减，因此我们可以确定，最大的那张牌一定会从起点牌堆移动到终点牌堆，那么在移动这张牌之前，我们要尽可能多地将牌从起点牌堆移出到空牌堆，在移动这张牌之后将之前移出的牌用类似的过程移回。上述移出起点牌堆和移入终点牌堆的过程是对称的，因此我们可以将问题转化为：现有一个空的终点牌堆和若干个有牌的牌堆，你要尽可能多地将牌移动到终点牌堆。

令$f(n)$为有n个牌堆时的最大移牌数，为了方便表述，我们认为$f(1)=0$。很显然$f(2)=1$。当$n>2$时，我们先移动点数最大的牌堆，此时除了终点牌堆外其余牌堆都有牌且点数小于当前堆的任意一张牌，因此这些牌堆又不可以使用，这一步可以移动$f(1)+1$张牌，其中的加1含义是移动牌堆底部的最大牌；随后我们移动点数第二大的牌堆，由于上一步操作，我们获得了一个空牌堆，这个空牌堆是可以借用的，所以这一步可以移动$f(2)+1$张牌，以此类推。故而$f(n)=\sum_{i=1}^{n-1}f(i)+1$。

根据递推式求得$f(n)=2^{n-1}-1$，用快速幂求值即可，时间复杂度$O(\log_2 n)$。

1005 Or

我们考虑一个从右向左的扫描线，即用线段树维护右端点为 i 时候的答案，然后依次加入左端点

我们考虑加入 l 对于右端点 r 的影响，相当于或上 $a_l, a_l+b_{l+1}, \cdots, a_l+b_{l+1}+b_{l+2}+\cdots+b_r$

然后这个东西对于右端点是累加的，考虑到或的性质，我们只需要求出每一个二进制位最早出现的位置即可，对于询问，可以离线下来之后用树状数组维护，时间复杂度 $O(n\log^2 n)$，单次询问 $O(m\log n)$

每一个二进制位最早出现的位置，我们将 a_l 贡献看成 $a_l+s_r-s_l$，这个也是一个经典的问题，我们将其看做 $s_r-(s_l-a_l)$

假设我们在考虑第 k 位，我们按照 s_l-a_l 是否大于等于 0 来进行分类

如果 $s_l-a_l\geq 0$，两数相减判断二进制第 k 位是否是 1 是一个经典问题，我们首先需要 s_l-a_l 和 s_r 对 2^k-1 取模，根据第 k 位是否是 1 以及取模后的大小关系可以快速判断，具体的对于 $x-y$，如果 x 的第 k 位是 1，y 的第 k 位也是 1 且 $x<y$ 或者 y 的第 k 位是 0 且 $x\geq y$ 就一定能保证 $x-y$ 的第 k 位是 1，如果 x 的第 k 位是 0 则情况正好相反，这种情况我们可以离散化后使用两个树状数组进行维护

如果 $s_l-a_l<0$，我们将其看做 $s_r+(-s_l+a_l)$，两个数相加判断第 k 位是否是 1 也是经典问题，具体的，当对2^{k+1} 取模后，$2^k\leq x+y<2^{k+1}, 2^k+2^{k+1}\leq x+y$ 可以保证 $x+y$ 的第 k 位是 1，这种情况我们可以使用动态开点线段树进行维护

总的时间复杂度为 $O(n\log^2 n+m\log n)$

1006 Fencing the cows

题意可以化简为有n个点和m个点，从n个点中选择尽可能少的点包围所有m个点。

思考最后的答案，一定是一个凸包包围了所有m个点，观察这些点的性质，所有的点一定处于凸包的边的同侧。因此将所有n个点两两组成边，如果m个点不位于边的同侧，那么这条边将被判定为不合法边，预处理完边的合法性之后，对所有合法边用Floyd做一次最小环。

但是如果只是判断是否同侧，可能会出现圈出一个没有包围任何一个点的凸包，因此我们可以将边记录成单向边，并且强制规定所有m个点需要在边的左侧，这样就可以避免这种情况。

复杂度 $O(n^2m+n^3)$

P.S. 由于出题人数据造的并不是很强，n个点的凸包大小比较小，导致部分没有进行预处理直接进行dp最小环的选手通过了此题（其实出题人本来预计这题是medium-easy的），以及本题仍然还有优化空间可以供大家探索 XD（才不是出题人开始没想到呢）

1007 foreverlasting and fried-chicken

考虑枚举 foreverlasting 图中度为6和4的两个点，分别称作 u, v ，设 cnt 为同时与 u, v 连边的点的个数，cnt_u 为与 u 相连的点的个数，那么对于这对 u, v ，foreverlasting 图的个数是

$$ans_{u,v} = \binom{cnt}{4}\binom{cnt_u - 4}{2}$$

那么答案就是

$$\sum_{u=1}^{n}\sum_{v=1}^{n} ans_{u,v}[u \neq v]$$

注意特判 u, v 之间连边的情况

以上是朴素 n^3 的做法，瓶颈在于如何快速求 cnt

1. 可以使用 $bitset$ 优化，复杂度$O(\frac{n^3}{w})$
2. 使用循环展开等手段减少$CPE(Cycles\ Per\ Element)$从而减小常数，出题人使用 12×12 的循环展开，时间上比 $bitset$ 做法快 $200ms$ （stl 的 $bitset$ 本身常数也较大）

1008 Hello World 3 Pro Max

本题有多种解法，这里主要讲矩阵做法。

首先考虑朴素DP：对于一个字符串S，并令目标串为T，$dp_{i,j}$表示子串$S(1,i)$中匹配目标串j位的子序列数的期望。

- 当S_i确定时，$dp_{i,j} = dp_{i-1,j} + (S_i == T_j) \cdot dp_{i-1,j-1}$，其中的$S_i == T_j$就是C语言的比较运算，得到0或1。
- 当S_i不确定时，$dp_{i,j} = dp_{i-1,j} + p_{T_j} \cdot dp_{i-1,j-1}$。

由于目标串很短，我们只需要每次维护11个值即可。

但这种朴素的DP做法，单次求解时间复杂度为$O(11n)$，由于有多次询问，总时间复杂度达到$O(11nq)$，不能通过，需要优化。

考虑将DP转化为矩阵：向量dp是一个11维的列向量，含义同上，令矩阵$A_{11\times 11}$表示转移关系，$dp_i = A_i \cdot dp_{i-1}$，则无论$S_i$是否确定，$A_{j,j} = 1$。

- 当S_i确定时，$A_{j-1,j} = (S_i == T_j)$。
- 当S_i不确定时，$A_{j-1,j} = p_{T_j}$。

除了上述提到的矩阵单元外，其余都是0。

为了解决多次询问，我们用线段树维护矩阵即可。时间复杂度$O(11^3(n + q\log_2 n))$。

接下来考虑优化：

- 优化1：由于上述所有的矩阵都是上三角矩阵（根据各人写法不同，可能灵活调整为下三角矩阵），在相乘的过程中始终还是上三角矩阵，因此我们在处理矩阵乘法时，只需处理半个矩阵即可。另外，根据三角矩阵的性质，原计算式$(A \times B)_{i,j} = \sum_{k=1}^{n} A_i^k \times B_k^j$可以优化为$(A \times B)_{i,j} = \sum_{k=i}^{j} A_i^k \times B_k^j$，从而进一步加快运行速度。

- 优化2：由于矩阵乘法中取模运算非常耗时间，用一个unsigned long long型整数来存储和，并在最后一次性取模，可以大幅度减少取模次数，也可以考虑正常使用long long运算，在中途和末尾各取模一次。
- 优化3：矩阵乘法规模固定且代码量不大，考虑循环展开，此方法不作要求。

上述优化1和优化2只需要任选一个即可通过本题，内测最快运行时间为468ms。

本题还有使用线段树维护区间DP的做法，其时间复杂度与矩阵做法相近。

1009 String Problem

签到题

相当于将字符串拆成尽可能少的段，使得每段只包含一种字母。每段对答案的贡献为长度减一。

1010 Klee likes making friends

$dp[j][k]$最后两个朋友在$j,k(j>k)$位置上的最小花费，因为连续m个人至少要有2个人是可莉的朋友，因此$j-k<m$当倒数第一个朋友在j，倒数第二个朋友在k位置时，因为(k,j)之间没有朋友，因此倒数第三个朋友的位置p必须在区间$[j-m,k-1]$内，这里可以求一个后缀最小值。

转移方程中的j,k,p之差不超过m,因此可以用取模的方式，把空间优化到$m*m$，因此$dp[i][j]$表示倒数第一个朋友在$(i \bmod m)$的位置倒数第二个朋友距离倒数第一个朋友j即倒数第二个朋友的位置为$i-k$。

1011 SPY finding NPY

枚举最大IQ的位置并统计概率，最大IQ的人在每个位置的概率都是$\frac{1}{n}$

$k=0$时概率为$\frac{1}{n}$

$$
\begin{aligned}
f(n,k)&=\sum_{i=k+1}^{n}\frac{1}{n}\times(p_1\sim p_{i-1}\text{中的最大值的位置在}[1,k]\text{的概率})\\
&=\sum_{i=k+1}^{n}\frac{1}{n}\times\frac{k}{i-1}\\
&=\frac{k}{n}\sum_{i=k}^{n-1}\frac{1}{i}
\end{aligned}
$$

对$\frac{1}{i}$求前缀和后，就可以$O(1)$得出某个k的概率。经过python高精度程序检验，发现$n=290000$多开始，double类型变量的精度不足以正确解答此题。为了避免让大家怀疑精度，这题n设得比较小

结果可以近似为$\frac{k}{n}\ln\frac{n}{k}$,发现是一个先增后减函数，当然也可以凭直觉发现单调性。

可以直接暴力枚举所有的k，$O(Tn)$可以通过。可以三分$O(Tlogn)$。

也可以求导得出$k=n/e$时概率最大，再枚举左右30个得出答案，经过测试发现枚举左右一个就可以通过。

1012 Coin

考虑建立最大流模型：

源点S向每个人连1的流量。

按时间顺序拆点，每个人拆成M个点，M个点之间连a_i的流量

对每次操作给定的(A,B)，在对应时间点上连双向的1的流量（正向边流量为1，反向边流量也为1）

对于K个朋友点，它们的最后一个时间的点向T连a_{b_i}流量的边。

这样建图，点数和边数都为$O(N*M)$

现在考虑对上面的最大流进行优化。

可以发现，拆出来的很多点是用不上的。

所以没必要每个点都拆成M个点。

每个人只要把选中自己的那个时间点拓展出来就行。

换句话说，刚开始每个人没有点，对于每个操作，只要把选中的那两个人多扩展出一个点即可。

这样建图后点数为$O(M)$,边数为$O(M+K)$

时间复杂度不超过$O(N*M)$

1013 Turrent

对于一个炮塔，可以在一个封闭多边形区域内移动它的位置使得它能瞄准的出生点不减少，在边界点上时能瞄准的出生点一定更多，不妨设炮塔的位置为封闭区域的顶点位置。做从出生点到建筑物的切线，将平面分割为若干区域，此时炮塔位于区域顶点上。

所以可以通过枚举出生点然后枚举建筑物做切线得到一个射线集合，同时将边界的四条线段加入集合。

两两枚举射线集合中的射线可以得到所有可能的炮塔位置。

对于每个炮塔位置，暴力判断它能否瞄准某个出生点，可以构建一个位于$[0,2^n)$的二进制数。

可能的炮塔位置至多有$2n^4$ 个，这部分暴力求解的时间复杂度是$O(mn^5)$ 。

然后问题就转变为了集合覆盖问题(set cover problem)，可以用fft解决，此部分的时间复杂度为$O(n^2 2^n)$。

注意到炮塔去重后的个数是很少的，后面的部分也能用dp解决。